Cloud Container Engine

Best Practices

 Issue
 01

 Date
 2025-07-11





HUAWEI CLOUD COMPUTING TECHNOLOGIES CO., LTD.

Copyright © Huawei Cloud Computing Technologies Co., Ltd. 2025. All rights reserved.

No part of this document may be reproduced or transmitted in any form or by any means without prior written consent of Huawei Cloud Computing Technologies Co., Ltd.

Trademarks and Permissions

NUAWEI and other Huawei trademarks are the property of Huawei Technologies Co., Ltd. All other trademarks and trade names mentioned in this document are the property of their respective holders.

Notice

The purchased products, services and features are stipulated by the contract made between Huawei Cloud and the customer. All or part of the products, services and features described in this document may not be within the purchase scope or the usage scope. Unless otherwise specified in the contract, all statements, information, and recommendations in this document are provided "AS IS" without warranties, guarantees or representations of any kind, either express or implied.

The information in this document is subject to change without notice. Every effort has been made in the preparation of this document to ensure accuracy of the contents, but all statements, information, and recommendations in this document do not constitute a warranty of any kind, express or implied.

Contents

1 Checklist for Deploying Containerized Applications in the Cloud	1
2 Containerization	13
2.1 Containerizing an Enterprise Application (ERP)	13
2.1.1 Solution Overview	13
2.1.2 Resource and Cost Planning	17
2.1.3 Procedure	17
2.1.3.1 Containerizing an Entire Application	18
2.1.3.2 Containerization Process	19
2.1.3.3 Analyzing the Application	20
2.1.3.4 Preparing the Application Runtime	22
2.1.3.5 Compiling a Startup Script	24
2.1.3.6 Compiling the Dockerfile	24
2.1.3.7 Building and Uploading an Image	
2.1.3.8 Creating a Container Workload	27
3 Migration	32
3.1 Migrating Container Images	32
3.1.1 Solution Overview	
3.1.2 Migrating Images to SWR Using Docker Commands	
3.1.3 Migrating Images to SWR Using image-migrator	35
3.1.4 Migrating Images Across Clouds from Harbor to SWR	41
3.2 Migrating Kubernetes Clusters to CCE	47
3.2.1 Solution Overview	47
3.2.2 Resource Planning for the Target Cluster	52
3.2.3 Procedure	54
3.2.3.1 Migrating Resources Outside a Cluster	54
3.2.3.2 Installing the Migration Tool	55
3.2.3.3 Migrating Resources in a Cluster (Velero)	58
3.2.3.4 Updating Resources Accordingly	61
3.2.3.5 Performing Additional Tasks	64
3.2.3.6 Troubleshooting	65
4 DevOps	68
4.1 Installing and Deploying Jenkins on CCE	68

4.1.1 Solution Overview	
4.1.2 Resource and Cost Planning	71
4.1.3 Procedure	
4.1.3.1 Installing and Deploying Jenkins Master	72
4.1.3.2 Configuring Jenkins Agent	
4.1.3.3 Using Jenkins to Build a Pipeline	
4.1.3.4 Interconnecting Jenkins with RBAC of Kubernetes Clusters (Example)	91
4.2 Interconnecting GitLab with SWR and CCE for CI/CD	97
4.3 Continuous Delivery Using Argo CD	104
4.4 Implementing Separate DevOps Processes for Multiple Clusters Using Jenkins and GitLab	112
4.4.1 Solution Overview	112
4.4.2 Resource Planning	114
4.4.3 Procedure	115
4.4.3.1 Setting Up the Jenkins and GitLab Environments	115
4.4.3.2 Configuring Cluster Environments	119
4.4.3.3 Configuring a GitLab Project	123
4.4.3.4 Implementing Continuous Integration and Deployment	124
5 Disaster Recovery	134
5.1 Recommended Configurations for HA CCE Clusters	134
5.2 Implementing High Availability for Applications in CCE	144
5.3 Implementing High Availability for Add-ons in CCE	146
6 Security	151
6.1 Overview	151
6.2 Configuration Suggestions on CCE Cluster Security	153
6.3 Configuration Suggestions on CCE Node Security	
6.4 Configuration Suggestions on CCE Container Runtime Security	160
6.5 Configuration Suggestions on CCE Container Security	162
6.6 Configuration Suggestions on CCE Container Image Security	166
6.7 Configuration Suggestions on CCE Secret Security	168
6.8 Configuration Suggestions on CCE Workload Identity Security	170
7 Auto Scaling	176
7.1 Using HPA and CA for Auto Scaling of Workloads and Nodes	176
7.2 Elastic Scaling of CCE Pods to CCI	
7.3 Auto Scaling Based on Prometheus Metrics	189
7.4 Auto Scaling Based on ELB Monitoring Metrics	193
7.5 Auto Scaling of Multiple Applications Using Nginx Ingresses	203
8 Monitoring	210
8.1 Monitoring Multiple Clusters Using Prometheus	210
8.2 Monitoring GPU Metrics Using DCGM-Exporter	214
8.3 Reporting Prometheus Monitoring Data to a Third-Party Monitoring Platform	222
8.4 Obtaining Prometheus Data Using PromQL Statements	225

8.5 Collecting GPU Pod Monitoring Metrics and Setting Up a Grafana Dashboard	230
9 Cluster	.252
9.1 Suggestions on CCE Cluster Selection	252
9.2 Creating an IPv4/IPv6 Dual-Stack Cluster in CCE	258
9.3 Creating a Custom CCE Node Image	267
9.4 Executing the Pre- or Post-installation Commands During Node Creation	275
9.5 Using OBS Buckets to Implement Custom Script Injection During Node Creation	276
9.6 Connecting to Multiple Clusters Using kubectl	280
9.7 Selecting a Data Disk for the Node	286
9.8 Implementing Cost Visualization for a CCE Cluster	290
9.9 Creating a CCE Turbo Cluster Using a Shared VPC	294
9.10 Protecting a CCE Cluster Against Overload	295
9.11 Managing Costs for a Cluster	303
10 Networking	. 313
10.1 Planning CIDR Blocks for a Cluster	313
10.2 Selecting a Network Model	322
10.3 Enabling Cross-VPC Network Communications Between CCE Clusters	
10.4 Implementing Network Communications Between Containers and IDCs Using VPC and Direct	
Connect	337
10.5 Enabling a CCE Cluster to Resolve Domain Names on Both On-Premises IDCs and Huawei Cloud	1343
10.5.1 Solution Overview	343
10.5.2 Solution 1: Using a DNS Endpoint for Cascading Resolution	346
10.5.3 Solution 2: Changing the CoreDNS Configurations	349
10.6 Implementing Sticky Session Through Load Balancing	352
10.7 Obtaining the Client Source IP Address for a Container	360
10.8 Increasing the Listening Queue Length by Configuring Container Kernel Parameters	367
10.9 Configuring Passthrough Networking for a LoadBalancer Service	371
10.10 Accessing an External Network from a Pod	374
10.10.1 Accessing the Internet from a Pod	374
10.10.2 Accessing Cloud Services from a Pod in the Same VPC	380
10.10.3 Accessing Cloud Services from a Pod in a Different VPC	387
10.11 Deploying Nginx Ingress Controllers Using a Chart	430
10.11.1 Deploying NGINX Ingress Controller in Custom Mode	430
10.11.2 Advanced Configuration of Nginx Ingress Controller	438
10.12 CoreDNS Configuration Optimization	441
10.12.1 CoreDNS Optimization Overview	441
10.12.2 Client	442
10.12.2.1 Optimizing Domain Name Resolution Requests	442
10.12.2.2 Selecting a Proper Image	443
10.12.2.3 Avoiding Occasional DNS Resolution Timeout Caused by IPVS Defects	443
10.12.2.4 Using NodeLocal DNSCache	444
10.12.2.5 Upgrading the CoreDNS in the Cluster Timely	444

10.12.2.6 Adjusting the DNS Configuration of the VPC and VM	445
10.12.3 Server	445
10.12.3.1 Monitoring the coredns Add-on	445
10.12.3.2 Adjusting the CoreDNS Deployment Status	446
10.12.3.3 Configuring CoreDNS	448
10.13 Pre-Binding Container Elastic Network Interfaces for CCE Turbo Clusters	455
10.14 Connecting a Cluster to the Peer VPC Through an Enterprise Router	461
10.15 Accessing an IP Address Outside of a Cluster That Uses a VPC Network by Using Source Pod IP Addresses Within the Cluster	468
10.16 Using Kmesh on Huawei Cloud CCE	471
11 Storage	477
11.1 Expanding the Storage Space	477
11.2 Mounting Object Storage Across Accounts	487
11.3 Dynamically Creating an SFS Turbo Subdirectory Using StorageClass	499
11.4 Changing the Storage Class Used by a Cluster of v1.15 from FlexVolume to CSI Everest	503
11.5 Using Custom Storage Classes	515
11.6 Scheduling EVS Disks Across AZs Using csi-disk-topology	527
11.7 Automatically Collecting JVM Dump Files That Exit Unexpectedly Using a General Purpose File System (SFS 3.0 Capacity-Oriented)	532
11.8 Deploying Storage Volumes in Multiple AZs	535
12 Container	540
12.1 Recommended Configurations for Workloads	540
12.2 Properly Allocating Container Computing Resources	544
12.3 Upgrading Pods Without Interrupting Services	545
12.4 Modifying Kernel Parameters Using a Privileged Container	549
12.5 Using Init Containers to Initialize an Application	550
12.6 Setting Time Zone Synchronization	552
12.7 Configuration Suggestions on Container Network Bandwidth Limit	556
12.8 Configuring the /etc/hosts File of a Pod Using hostAliases	558
12.9 Configuring Domain Name Resolution for CCE Containers	560
12.10 Using Dual-Architecture Images (x86 and Arm) in CCE	565
12.11 Locating Container Faults Using the Core Dump File	568
12.12 Configuring Parameters to Delay the Pod Startup in a CCE Turbo Cluster	570
12.13 Automatically Updating a Workload Version Using SWR Triggers	571
12.14 Effective Troubleshooting in Kubernetes with Temporary Containers	576
12.15 Developing and Testing a Microservice Locally Using Telepresence	579
12.16 Containerizing and Using QingTian Enclave on CCE	582
13 Permission	. 590
13.1 Configuring kubeconfig for Fine-Grained Management on Cluster Resources	590
13.2 Configuring Namespace-level Permissions for an IAM User	594
13.3 Performing RBAC Authentication on a Namespace Using kubectl Commands	605
14 Release	610

14.1 Overview	610
14.2 Using Services to Implement Simple Grayscale Release and Blue-Green Deployment	613
14.3 Using Nginx Ingress to Implement Grayscale Release and Blue-Green Deployment	619
15 Batch Computing	629
15.1 Deploying and Using Kubeflow in a CCE Cluster	629
15.1.1 Deploying Kubeflow	629
15.1.2 Training a TensorFlow Model	633
15.1.3 Using Kubeflow and Volcano to Train an AI Model	636
15.2 Deploying and Using Caffe in a CCE Cluster	640
15.2.1 Prerequisites	640
15.2.2 Preparing Resources	642
15.2.3 Caffe Classification Example	643
15.3 Deploying and Using TensorFlow in a CCE Cluster	646
15.4 Deploying and Using Flink in a CCE Cluster	652
15.5 Deploying and Using ClickHouse in a CCE Cluster	660
15.6 Deploying and Using Spark in a CCE Cluster	670
15.6.1 Installing Spark	670
15.6.2 Using Spark	674

Checklist for Deploying Containerized Applications in the Cloud

Overview

Security, efficiency, stability, and availability are common requirements on all cloud services. To meet these requirements, the system availability, data reliability, and O&M stability must be coordinated. This checklist describes the check items for deploying containerized applications on the cloud to help you efficiently migrate services to CCE, reducing potential cluster or application exceptions caused by improper use.

Check Items

Categor y	Check Item	Туре	Impact	FAQ & Example
Cluster	Before creating a cluster, properly plan the node network and container network based on service requirements to allow subsequent service expansion.	Network planning	If the subnet or container CIDR block where the cluster resides is small, the number of available nodes supported by the cluster may be less than required.	 Network Planning Planning CIDR Blocks for a Cluster How Do I Set the VPC CIDR Block and Subnet CIDR Block for a CCE Cluster?

 Table 1-1
 System availability

Categor y	Check Item	Туре	Impact	FAQ & Example
	Before creating a cluster, properly plan CIDR blocks for the related Direct Connect, peering connection, container network, service network, and subnet to avoid IP address conflicts.	Network planning	If CIDR blocks are not properly set and IP address conflicts occur, service access will be affected.	 Connectivity Planning CIDR Blocks for a Cluster
	When a cluster is created, the default security group is automatically created and bound to the cluster. You can set custom security group rules based on service requirements.	Deployme nt	Security groups are key to security isolation. Improper security policy configuration may cause security risks and service connectivity problems.	 Security Groups and Security Group Rules How Do I Prevent Cluster Nodes from Being Exposed to Public Networks?

Categor y	Check Item	Туре	Impact	FAQ & Example
	Enable the multi-master node mode, and set the number of master nodes to 3 when creating a cluster.	Reliability	After the multi- master node mode is enabled, three master nodes will be created. If a master node is faulty, the cluster can still be available without affecting service functions. In commercial scenarios, it is advised to enable the multi-master node mode.	How Do I Check Whether a Cluster Is in Multi-Master Mode? Once a cluster is created, the number of master nodes cannot be changed. Exercise caution when setting the number of master nodes.
	 When creating a cluster, select a proper network model as needed. Select VPC network or Tunnel network for your CCE standard cluster. Select Cloud Native Network 2.0 for your CCE Turbo cluster. 	Deployme nt	After a cluster is created, the network model cannot be changed.	Network Model Comparison

Categor y	Check Item	Туре	Impact	FAQ & Example
Workloa d	When creating a workload, set the CPU and memory limits to improve service robustness.	Deployme nt	When multiple applications are deployed on the same node, if the upper and lower resource limits are not set for an application, resource leakage occurs. As a result, resources cannot be allocated to other applications, and the application monitoring information will be inaccurate.	None
	When creating a workload, you can set probes for container health check, including liveness probe and readiness probe .	Reliability	If the health check function is not configured, a pod cannot detect service exceptions or automatically restart the service to restore it. This results in a situation where the pod status is normal but the service in the pod is abnormal.	 Setting Health Check for a Container Enabling ICMP Security Group Rules

Categor y	Check Item	Туре	Impact	FAQ & Example
	When creating a workload, do not set the number of replicas for a single pod. Set a proper node scheduling policy based on your service requirements.	Reliability	For example, if the number of replicas of a single pod is set, the service will be abnormal when the node or pod is abnormal. To ensure that your pods can be successfully scheduled, ensure that the node has idle resources for container scheduling after you set the scheduling rule.	None

Categor y	Check Item	Туре	Impact	FAQ & Example
	Properly set affinity and anti-affinity.	Reliability	If affinity and anti-affinity are both configured for an application that provides Services externally, Services may fail to be accessed after the application is upgraded or restarted.	Scheduling Policy (Affinity/Anti- affinity) Negative example: For application A, nodes 1 and 2 are set as affinity nodes, and nodes 3 and 4 are set as anti-affinity nodes. Application A exposes a Service through the ELB, and the ELB listens to node 1 and node 2. When application A is upgraded, it may be scheduled to a node other than nodes 1, 2, 3, and 4, and it cannot be accessed through the Service.
				Cause:
				Scheduling of application A does not need to meet both affinity and anti-affinity policies. A node will be selected for application A according to either of the policies. In this example, the node selection is based on the anti- affinity scheduling policy.

Categor y	Check Item	Туре	Impact	FAQ & Example
	When creating a workload, set the pre-stop processing command (Lifecycle > Pre-Stop) to ensure that the services running in the pods can be completed in advance in the case of application upgrade or pod deletion.	Reliability	If the pre-stop processing command is not configured, the pod will be directly killed and services will be interrupted during application upgrade.	 Setting Container Lifecycle Parameters When Is Pre- stop Processing Used?
Networ king	When creating a workload, select a proper access mode (Service). The following types of Services are supported: ClusterIP, NodePort, DNAT, and LoadBalancer.	Deployme nt	Improper Service configuration may cause logic confusion for internal and external access and resource waste.	• Network
	When creating a LoadBalancer Service or ingress, configure health checks to ensure service reliability.	Reliability	If a health check is not configured, the services in a pod might not be working properly, but traffic will still be directed to that pod.	 Creating a LoadBalancer Service Creating a LoadBalancer Ingress on the Console

Category	Check Item	Туре	Impact	FAQ & Example
Container data persistenc y	Select a proper data volume type based on service requirements.	Reliability	When a node is faulty and cannot be recovered, data in the local disk cannot be recovered. Therefore, you are advised to use cloud storage volumes to ensure data reliability.	• Storage Overview
Backup	Back up application data.	Reliability	Data cannot be restored after being lost.	What Are the Differences Among CCE Storage Classes in Terms of Persistent Storage and Multi-node Mounting?

Table 1-2 Data reliability

Table 1-3 O&M reliability

Category	Check Item	Туре	Impact	FAQ & Example
Project	The quotas of ECS, VPC, subnet, EIP, and EVS resources must meet customer requirements.	Deployme nt	If the quota is insufficient, resources will fail to be created. Specifically, users who have configured auto scaling must have sufficient resource quotas.	 Which Resource Quotas Should I Pay Attention To When Using CCE? Notes and Constraints

Category	Check Item	Туре	Impact	FAQ & Example
	You are not advised to modify kernel parameters, system configurations , cluster core component versions, security groups, and ELB-related parameters on cluster nodes, or install software that has not been verified.	Deployme nt	Exceptions may occur on CCE clusters or Kubernetes components on the node, making the node unavailable for application deployment.	 For details, see High-Risk Operations and Solutions. Negative example: The container network is interrupted after the node kernel is upgraded. The container network is interrupted after an open-source Kubernetes network add-on is installed on a node. The /var/paas or /mnt/paas/kubernetes directory is deleted from a node, which causes exceptions on the node.

Category	Check Item	Туре	Impact	FAQ & Example
	Do not modify information about resources created by CCE, such as security groups and EVS disks. Resources created by CCE are labeled cce .	Deployme nt	CCE cluster functions may be abnormal.	 Negative example: 1. On the ELB console, a user changes the name of the listener created by CCE. 2. On the VPC console, a user modifies the security group created by CCE. 3. On the EVS console, a user deletes or uninstalls data disks mounted to CCE cluster nodes. 4. On the IAM console, a user deletes cce_admin_trust. All the preceding actions will cause CCE cluster functions to be
				abnormal.

Category	Check Item	Туре	Impact	FAQ & Example
Proactive O&M	CCE provides multi- dimensional monitoring and alarm reporting functions, allowing users to locate and rectify faults as soon as possible. • Application Operations Manageme nt (AOM): The default basic resource monitoring of CCE covers detailed container- related metrics and provides alarm reporting functions. • Open- source Prometheu s: A monitoring tool for cloud native application s. It integrates an independe nt alarm system to provide more	Monitorin g	If the alarms are not configured, the standard of container cluster performance cannot be established. When an exception occurs, you cannot receive alarms and will need to manually locate the fault.	 Monitoring Overview Monitoring Custom Metrics Using the Cloud Native Monitoring Add- on

Category	Check Item	Туре	Impact	FAQ & Example
	flexible monitoring and alarm reporting functions.			

2 Containerization

2.1 Containerizing an Enterprise Application (ERP)

2.1.1 Solution Overview

This chapter provides CCE best practices to walk you through the application containerization.

What Is a Container?

A container is a lightweight high-performance resource isolation mechanism implemented based on the Linux kernel. It is a built-in capability of the operating system (OS) kernel.

CCE is an enterprise-class container service based on open source Kubernetes. It is a high-performance and high-reliability service through which enterprises can manage containerized applications. CCE supports native Kubernetes applications and tools, allowing you to easily set up a container runtime in the cloud.

Why Is a Container Preferred?

• More efficient use of system resources

A container does not require extra costs such as fees for hardware virtualization and those for running a complete OS. Therefore, a container has higher resource usage. Compared with a VM with the same configurations, a container can run more applications.

• Faster startup

A container directly runs on the host kernel and does not need to start a complete OS. Therefore, a container can be started within seconds or even milliseconds, greatly saving the development, testing, and deployment time.

• Consistent runtime environment

A container image provides a complete runtime environment to ensure environment consistency. In this case, problems (for example, some code runs properly on machine A but fails to run on machine B) will not occur. • Easier application migration, maintenance, and scaling

A consistent runtime environment makes application migration easier. In addition, the in-use storage and image technologies facilitate the reuse of repeated applications and simplifies the expansion of images based on base images.

Containerization Modes

The following modes are available for containerizing applications:

- Mode 1: Containerize a single application as a whole. Application code and architecture remain unchanged.
- Mode 2: Separate the components that are frequently upgraded or have high requirements on auto scaling from an application, and then containerize these components.
- Mode 3: Transform an application to microservices and then containerize the microservices one by one.

 Table 2-1 lists the advantages and disadvantages of the three modes.

Containerization Mode	Advantage	Disadvantage
Method 1: Containerize a single application as a whole.	 Zero modification on services: The application architecture and code require no change. The deployment and upgrade efficiency is improved. Applications can be packed as container images to ensure application environment consistency and improve deployment efficiency. Reduce resource costs: Containers use system resources more efficiently. Compared with a VM with the same configurations, a container can run more applications. 	 Difficult to expand the entire architecture of an application. As the code size increases, code update and maintenance would be complicated. Difficult to launch new functions, languages, frameworks, and technologies.

Table 2-1 Containerization modes

Containerization Mode	Advantage	Disadvantage
Method 2: Containerize first the application components that are frequently updated or have high requirements on auto scaling.	 Progressive transformation: Reconstructing the entire architecture involves a heavy workload. This mode containerizes only a part of components, which is easy to accept for customers. 	Need to decouple some services.
	• Flexible scaling: Application components that have high requirements on auto scaling are containerized. When the application needs to be scaled, you only need to scale the containers, which is flexible and reduces the required system resources.	
	• Faster rollout of new features: Application components that are frequently upgraded are containerized. In subsequent upgrades, only these containers need to be upgraded. This shortens the time to market (TTM) of new features.	

Containerization Mode	Advantage	Disadvantage
Method 3: Transform an application to microservices and then containerize the microservices one by one.	 Independent scaling: After an application is split into microservices, you can independently increase or decrease the number of instances for each microservice. 	Need to transform the application to microservices, which involves a large number of changes.
	 Increased development speed: Microservices are decoupled from one another. Code development of a microservice does not affect other microservices. 	
	• Security assurance through isolation: For an overall application, if a security vulnerability exists, attackers can use this vulnerability to obtain the permission to all functions of the application. However, in a microservice architecture, if a service is attacked, attackers can only obtain the access permission to this service, but cannot intrude other services.	
	 Breakdown isolation: If one microservice breaks down, other microservices can still run properly. 	

Mode 1 is used as an example in this tutorial to illustrate how to containerize an enterprise resource planning (ERP) system.

2.1.2 Resource and Cost Planning

NOTICE

The fees listed here are estimates. The actual fees vary with regions and will be displayed on the Huawei Cloud console.

The following table lists the resources needed in this practice.

Table 2-2 Resource and cost planning

Resource	Description	Quanti ty	Estimated Fee
ECS	 Pay-per-use recommended VM type: general computing-plus Node flavor: 4 vCPUs 8 GiB OS: Ubuntu 22.04 System disk: 40 GiB General purpose SSD EIP specification: billed by bandwidth, 5 Mbit/s 	1	USD0.3123/ hour
Cloud Container Engine (CCE)	 Pay-per-use recommended Cluster type: CCE cluster Cluster version: v1.25 Cluster scale: 50 nodes HA: Yes 	1	USD0.42/hour
	 Pay-per-use recommended Node type: general computing-plus Node flavor: 4 vCPUs 8 GiB OS: EulerOS 2.9 System disk: 50 GiB General purpose SSD Data disk: 100 GiB General purpose SSD 	1	USD0.2096/ hour
Elastic Volume Service (EVS)	Pay-per-use recommendedEVS disk specification: 100 GiBEVS disk type: General purpose SSD	1	USD0.0157/ hour

2.1.3 Procedure

2.1.3.1 Containerizing an Entire Application

This tutorial describes how to containerize an ERP system by migrating it from a VM to CCE.

No recoding or re-architecting is required. You only need to pack the entire application into a container image and deploy the container image on CCE.

Introduction

In this example, the **enterprise management application** is developed by enterprise A. This application is provided for third-party enterprises for use, and enterprise A is responsible for application maintenance.

When a third-party enterprise needs to use this application, a suit of **Tomcat application** and **MongoDB database** must be deployed for the third-party enterprise. The MySQL database, used to store data of third-party enterprises, is provided by enterprise A.



As shown in **Figure 2-1**, the application is a standard Tomcat application, and its backend interconnects with MongoDB and MySQL databases. For this type of applications, there is no need to split the architecture. The entire application is built as an image, and the MongoDB database is deployed in the same image as the Tomcat application. In this way, the application can be deployed or upgraded through the image.

- Interconnecting with the MongoDB database for storing user files.
- Interconnecting with the MySQL database for storing third-party enterprise data. The MySQL database is an external cloud database.

Benefits

In this example, the application was deployed on a VM. During application deployment and upgrade, a series of problems is encountered, but application containerization has solved these problems.

By using containers, you can easily pack application code, configurations, and dependencies and convert them into easy-to-use building blocks. This achieves the environmental consistency and version management, as well as improves the development and operation efficiency. Containers ensure quick, reliable, and consistent deployment of applications and prevent applications from being affected by deployment environment.

Category	Before: Application Deployment on VM	After: Application Deployment Using Containers
Deployment	High deployment cost. A VM is required for deploying a system for a customer.	More than 50% cost reduced. Container services achieve multi- tenant isolation, which allows you to deploy systems for different enterprises on the same VM.
Upgrade	Low upgrade efficiency. During version upgrades, log in to VMs one by one and manually configure the upgrades, which is inefficient and error-prone.	Per-second level upgrade. Version upgrades can be completed within seconds by replacing the image tag. In addition, CCE provides rolling updates, ensuring zero service downtime during upgrades.
Operation and maintenanc e (O&M)	High O&M cost. As the number of applications deployed for customer grows, the number of VMs that need to be maintained increases accordingly, which requires a large sum of maintenance cost.	Automatic O&M Enterprises can focus on service development without paying attention to VM maintenance.

Table 2-3 Comparison between the two deployment modes

2.1.3.2 Containerization Process

To fully containerize an application, you must go through the entire process.

This involves analyzing the application, setting up the runtime environment for the application, compiling the startup script and Dockerfile, creating and uploading images, and creating containerized workloads.

For details about each step of the containerization, see Containerization Process.



Figure 2-2 Process of containerizing an application

2.1.3.3 Analyzing the Application

Before containerizing an application, analyze the running environment and dependencies of the application, and get familiar with the application deployment mode. For details, see **Table 2-4**.

Table 2-4	Application	environment
-----------	-------------	-------------

Category	Sub-category	Description
Runtime environmen	OS	OS that the application runs on, such as CentOS or Ubuntu.
t		In this example, the application runs on CentOS 7.1.

Category	Sub-category	Description			
	Runtime environment	The Java application requires Java Development Kit (JDK), the Go language requires GoLang, the web application requires Tomcat environment, and the corresponding version number needs to be confirmed.			
		In this example, the web application of the Tomcat type is used. This application requires the runtime environment of Tomcat 7.0, and Tomcat requires JDK 1.8.			
	Dependency package	Understand required dependency packages, such as OpenSSL and other system software, and their version numbers.			
		In this example, no dependency package is required.			
Deployment mode	Peripheral configuration s	MongoDB database: In this example, the MongoDB database and Tomcat application are deployed on the same server. Therefore, their configurations can be fixed and there is no need to extract their configurations.			
		External services with which the application needs to interconnect, such as databases and file systems.			
		These configurations need to be manually configured each time you deploy an application on a VM. However, through containerized deployment, environment variables can be injected into a container, facilitating deployment.			
		In this example, the application needs to interconnect with the MySQL database. Obtain the database configuration file. The server address, database name, database login			
		injected through environment variables.			
		connection URL username=**** #Username for logging in to the database			
		password=**** #Password for logging in to the database			

Category	Sub-category	Description
	Application configuration s	Sort out the configuration parameters, such as configurations that need to be modified frequently and those remain unchanged during the running of the application.
		In this example, no application configurations need to be extracted.
		NOTE To avoid frequent image replacement, you are advised to classify configurations of the application.
		 For the configurations (such as peripheral interconnection information and log levels) that are frequently changed, you are advised to configure them as environment variables.
		 For the configurations that remain unchanged, directly write them into images.

2.1.3.4 Preparing the Application Runtime

After application analysis, you have gained the understanding of the OS and runtime required for running the application. Make the following preparations:

- **Installing Docker**: During application containerization, build a container image. To do so, you have to prepare a PC and install Docker on it.
- **Obtaining the runtime**: Obtain the runtime of the application and the MongoDB database with which the application interconnects.

Installing Docker

Docker is compatible with almost all operating systems. Select a Docker version that best suits your needs.

NOTE

SWR uses Docker 1.11.2 or later to upload images.

It is recommended that you install Docker and build images as the user **root**. Make sure to obtain the user **root** password for the host where Docker will be installed beforehand.

- Step 1 Log in as user root to the device on which Docker is about to be installed.
- **Step 2** Quickly install Docker on the device running Linux. You can also manually install Docker. For details, see **Docker Engine installation**.

curl -fsSL get.docker.com -o get-docker.sh

sh get-docker.sh

Step 3 Run the following command to check the Docker version:

docker version Client: Version: 17.12.0-ce API Version:1.35 Version indicates the version number.

----End

Obtaining the Runtime

In this example, the web application of the Tomcat type is used. This application requires the runtime of Tomcat 7.0, and Tomcat requires JDK 1.8. In addition, the application must interconnect with the MongoDB database in advance.

NOTE

Download the environment required by the application.

- **Step 1** Download Tomcat, JDK, and MongoDB installation packages of the specific versions.
 - 1. Download JDK 1.8.

Download address: https://www.oracle.com/java/technologies/jdk8downloads.html.

- 2. Download Tomcat 7.0 from http://archive.apache.org/dist/tomcat/ tomcat-7/v7.0.82/bin/apache-tomcat-7.0.82.tar.gz.
- 3. Download MongoDB 3.2 from https://fastdl.mongodb.org/linux/mongodblinux-x86_64-rhel70-3.2.9.tgz.
- Step 2 Log in as user root to the device running Docker.
- **Step 3** Run the following commands to create the directory where the application is to be stored: For example, set the directory to **apptest**.

mkdir apptest

cd apptest

- Step 4 Use Xshell to save the downloaded dependency files to the apptest directory.
- Step 5 Run the following commands to decompress the dependency files:

tar -zxf apache-tomcat-7.0.82.tar.gz

tar -zxf jdk-8u151-linux-x64.tar.gz

tar -zxf mongodb-linux-x86_64-rhel70-3.2.9.tgz

Step 6 Save the enterprise application (for example, **apptest.war**) in the **webapps/ apptest** directory of the Tomcat runtime environment.

NOTE

apptest.war is used as an example only. Use your own application for actual configuration.

mkdir -p apache-tomcat-7.0.82/webapps/apptest

cp apptest.war apache-tomcat-7.0.82/webapps/apptest

cd apache-tomcat-7.0.82/webapps/apptest

./../../jdk1.8.0_151/bin/jar -xf apptest.war

rm -rf apptest.war

----End

2.1.3.5 Compiling a Startup Script

During application containerization, prepare a startup script. The method of compiling this script is the same as that of compiling a shell script. The startup script is used to:

- Start up the software on which the application depends.
- Set the configurations that need to be changed as the environment variables.

NOTE

Startup scripts vary according to applications. Edit the script based on your service requirements.

Procedure

- Step 1 Log in as user root to the device running Docker.
- **Step 2** Run the following command to switch to the directory where the application is to be stored:

cd apptest

Step 3 Compile a script file. The name and content of the script file vary according to applications. Edit the script file based on your application. The following example is only for your reference.

vi start_tomcat_and_mongo.sh

```
#!/bin/bash
# Load system environment variables.
source /etc/profile
# Start MongoDB. The data is stored in /usr/local/mongodb/data.
./usr/local/mongodb/bin/mongod --dbpath=/usr/local/mongodb/data --logpath=/usr/local/mongodb/logs
--port=27017 -fork
# These three script commands indicate that the contents related to the MySQL database in the
environment variables are written into the configuration file when Docker is started.
sed -i "s|mysql://.*/awcp_crmtile|mysql://$MYSQL_URL/$MYSQL_DB|g" /root/apache-tomcat-7.0.82/
webapps/awcp/WEB-INF/classes/conf/jdbc.properties
sed -i "s|username=.*|username=$MYSQL_USER|g" /root/apache-tomcat-7.0.82/webapps/awcp/WEB-INF/
classes/conf/jdbc.properties
sed -i "s|password=.*|password=$MYSQL PASSWORD|g" /root/apache-tomcat-7.0.82/webapps/awcp/WEB-
INF/classes/conf/jdbc.properties
# Start Tomcat.
bash /root/apache-tomcat-7.0.82/bin/catalina.sh run
```

----End

2.1.3.6 Compiling the Dockerfile

An image is the basis of a container. A container runs based on the content defined in the image. An image has multiple layers. Each layer includes the modifications made based on the previous layer.

Generally, Dockerfiles are used to customize images. Dockerfile is a text file and contains various instructions. Each instruction is used to build an image layer. That is, each instruction describes how to build an image layer.

This section describes how to compile a Dockerfile file.

NOTE

Dockerfiles vary according to applications. Dockerfiles need to be compiled based on actual service requirements.

For details on how to write a quality Dockerfile, see Writing a Quality Dockerfile.

Procedure

- Step 1 Log in as the root user to the device running Docker.
- **Step 2** Compile a Dockerfile.

vi Dockerfile

The content is as follows:

CentOS 7.1.1503 is used as the base image. FROM centos:7.1.1503 # Create a folder to store data and dependency files. You are advised to write multiple commands into one line to reduce the image size. RUN mkdir -p /usr/local/mongodb/data \ && mkdir -p /usr/local/mongodb/bin \ && mkdir -p /root/apache-tomcat-7.0.82 \ && mkdir -p /root/jdk1.8.0_151 # Copy the files in the apache-tomcat-7.0.82 directory to the container path. COPY ./apache-tomcat-7.0.82 /root/apache-tomcat-7.0.82 # Copy the files in the jdk1.8.0_151 directory to the container path. COPY ./jdk1.8.0_151 /root/jdk1.8.0_151

Copy the files in the mongodb-linux-x86_64-rhel70-3.2.9 directory to the container path. COPY ./mongodb-linux-x86_64-rhel70-3.2.9/bin /usr/local/mongodb/bin # Copy start_tomcat_and_mongo.sh to the /root directory of the container. COPY ./start_tomcat_and_mongo.sh /root/

```
# Enter Java environment variables.
RUN chown root:root -R /root \
  && echo "JAVA_HOME=/root/jdk1.8.0_151 " >> /etc/profile \
  && echo "PATH=\$JAVA_HOME/bin:$PATH " >> /etc/profile \
  && echo "CLASSPATH=::\$JAVA_HOME/lib/dt.jar:\$JAVA_HOME/lib/tools.jar" >> /etc/profile \
  && chmod +x /root \
  && chmod +x /root \
```

```
# When the container is started, commands in start_tomcat_and_mongo.sh are automatically run. The file can be one or more commands, or a script.
ENTRYPOINT ["/root/start_tomcat_and_mongo.sh"]
```

In the preceding information:

- FROM statement: indicates that centos:7.1.1503 is used as the base image.
- Run statement: indicates that a shell command is executed in the container.
- **COPY** statement: indicates that files in the local computer are copied to the container.
- **ENTRYPOINT** statement: indicates the commands that are run after the container is started.

----End

2.1.3.7 Building and Uploading an Image

This section describes how to build an entire application into a Docker image. After building an image, you can use the image to deploy and upgrade the application. This reduces manual configuration and improves efficiency.

NOTE

When building an image, ensure that files used to build the image are stored in the same directory.

Required Cloud Services

SoftWare Repository for Container (SWR) provides easy, secure, and reliable management over container images throughout their lifecycle, facilitating the deployment of containerized services.

Basic Concepts

- Image: A Docker image is a special file system that includes everything needed to run containers: programs, libraries, resources, settings, and so on. It also includes corresponding configuration parameters (such as anonymous volumes, environment variables, and users) required within a container runtime. An image does not contain any dynamic data, and its content remains unchanged after being built.
- Container: Images become containers at runtime, that is, containers are created from images. A container can be created, started, stopped, deleted, or suspended.

Procedure

- **Step 1** Log in as the **root** user to the device running Docker.
- Step 2 Enter the apptest directory.

cd apptest

Ensure that files used to build the image are stored in the same directory.

Step 3 Build an image.

docker build -t apptest:v1.

Step 4 Upload the image to SWR. For details, see Uploading an Image Through a Container Engine Client.

----End

2.1.3.8 Creating a Container Workload

This section describes how to deploy a workload on CCE. When using CCE for the first time, create an initial cluster and add a node into the cluster.

NOTE

Containerized workloads are deployed in a similar way. The difference lies in:

- Whether environment variables need to be set.
- Whether cloud storage is used.

Required Cloud Services

- Cloud Container Engine (CCE): a highly reliable and high-performance service that allows enterprises to manage containerized applications. With support for Kubernetes-native applications and tools, CCE makes it simple to set up an environment for running containers in the cloud.
- Elastic Cloud Server (ECS): a scalable and on-demand cloud server. It helps you to efficiently set up reliable, secure, flexible application environments, ensuring stable service running and improving O&M efficiency.
- Virtual Private Cloud (VPC): an isolated and private virtual network environment that users apply for in the cloud. You can configure the IP address ranges, subnets, and security groups, as well as assign elastic IP addresses and allocate bandwidth in a VPC.

Basic Concepts

- A cluster is a collection of computing resources, including a group of node resources. A container runs on a node. Before creating a containerized application, you must have an available cluster.
- A node is a virtual or physical machine that provides computing resources. You must have sufficient node resources to ensure successful operations such as creating applications.
- A workload indicates a group of container pods running on CCE. CCE supports third-party application hosting and provides the full lifecycle (from deployment to O&M) management for applications. This section describes how to use a container image to create a workload.

Procedure

Step 1 Prepare the environment as described in **Table 2-5**.

No.	Category	Procedure		
1	Creating a VPC	Create a VPC before you create a cluster. A VPC provides an isolated, configurable, and manageable virtual network environment for CCE clusters.		
		If you have a VPC already, skip to the next task.		
		1. Log in to the management console.		
		 In the service list, choose Networking > Virtual Private Cloud. 		
		3. On the Dashboard page, click Create VPC .		
		4. Follow the instructions to create a VPC. Retain default settings for parameters unless otherwise specified.		
2	Creating a key pair	Create a key pair before you create a containerized application. Key pairs are used for identity authenticatior during remote login to a node. If you have a key pair already, skip this task.		
		1. Log in to the management console.		
		 In the service list, choose Security & Compliance > Data Encryption Workshop. 		
		3. In the navigation pane, choose Key Pair Service . On the Private Key Pairs tab, click Create Key Pair .		
		 Enter a key pair name, select I agree to host the private key of the key pair. and I have read and agree to the Key Pair Service Disclaimer, and click OK. 		
		5. View and save the private key. For security purposes, a key pair can be downloaded only once. Keep it secure to ensure successful login.		

Table 2-5	Preparing	the	environment
-----------	-----------	-----	-------------

Step 2 Create a cluster and a node.

1. Log in to the CCE console. On the **Clusters** page, click **Buy Cluster** and select the type for the cluster to be created.

Configure cluster parameters and select the VPC created in **Step 1**. For details, see **Buying a CCE Cluster**.

- 2. Buy a node and select the key pair created in **Step 1** as the login option. For details, see **Creating a Node**.
- **Step 3** Deploy a workload on CCE.
 - 1. Log in to the CCE console and click the name of the cluster to access the cluster console. In the navigation pane, choose **Workloads** and click **Create Workload**.
 - 2. Configure the following parameters, and retain the default settings for other parameters:
 - Workload Name: Set it to apptest.

- **Pods**: Set it to **1**.

Figure 2-3 Basic settings

Basic Info									
Workload Type	Deployment	StatefulSet	Daemo	onSet	Job	Cron Job			
	A Switching workloa	d type requires reconfiguring) workload par	rameters.					
Workload Name	apptest						Cluster Name	CCE Standard	
Namespace	default		~	Q Create N	lamespace		Description	A maximum of 200 characters are supported.	
Pods	- 1 +	·							0/200

- 3. In the **Container Settings** area, select the image uploaded in **Building and Uploading an Image**.
- 4. In the **Container Settings** area, choose **Environment Variables** and add environment variables for interconnecting with the MySQL database. The environment variables are set in the **startup script**.

NOTE

In this example, interconnection with the MySQL database is implemented through configuring the environment variables. Determine whether to use environment variables based on your service requirements.

Variable Name	Variable Value/Variable Reference				
MYSQL_DB	Database name.				
MYSQL_URL	IP address and port number of the database.				
MYSQL_USER	Database username.				
MYSQL_PASSWOR D	Database user password.				

Table 2-6 Configuring environment variables

5. In the **Container Settings** area, choose **Data Storage** and configure cloud storage for persistent data storage.

NOTE

In this example, the MongoDB database is used and persistent data storage is also needed, so you need to configure cloud storage. Determine whether to use cloud storage based on your service requirements.

The mounted path must be the same as the MongoDB storage path in the Docker startup script. For details, see the **startup script**. In this example, the path is **/usr/local/mongodb/data**.
Figure 2-4 Configuring cloud storage

Container - 1							+ Add Container
Basic Info	Local Volumes PersistentVolumeClaims (PVI	(s)					
Lifecycle							
Health	PVC Name	Volume Type	Storage Class	Mount Path (?)	Subpath (?)	Permission	Operation
Check	pvc-obj	EVS	csi-disk	/user/local/mongodb/data	Enter a sub-path, for example, tmp.	Read-write •	Delete
Environment							
Canadico -				+			
Data Storage							
Security Context							
Logging							

6. In the **Service Settings** area, click + to add a service, configure workload access parameters, and click **OK**.

NOTE

In this example, the application can be accessed from the Internet through an EIP.

- **Service Name**: name of the application that can be accessed externally. In this example, this parameter is set to **apptest**.
- Service Type: Select NodePort.
- Service Affinity
 - Cluster-level: The IP addresses and access ports of all nodes in a cluster can be used to access the workload associated with the Service. Service access will cause performance loss due to route redirection, and the source IP address of the client cannot be obtained.
 - Node-level: Only the IP address and access port of the node where the workload is located can be used to access the workload associated with the Service. Service access will not cause performance loss due to route redirection, and the source IP address of the client can be obtained.
- Port
 - **Protocol**: Set it to **TCP**.
 - Service Port: port for accessing the Service.
 - **Container Port**: port that the application will listen on the container. In this example, this parameter is set to **8080**.
 - Node Port: Set it to Auto. The system automatically opens a real port on all nodes in the current cluster and then maps the port number to the container port.

Figure	2-5	Creating	а	Service
--------	-----	----------	---	---------

Create Service					>
Service Name	apptest				
Service Type	ClusterIP Expose services through the internal IP of the cluster, which can only be accessed within the cluster It is recommended to select the	NodePort Expose services via IP and static port (NodePort) on each node	LoadBalancer Provide external services through ELB load balancing, high availability, ultra-high performance, stability and security utof-cluster access	DNAT Expose cluster node access type services through NAT gateway, support multiple nodes to share and use elastic IP	
Service Affinity	Cluster-level Node-leve				
Ports	Protocol Container	Port Service Port ⑦ N	ode Port		0
	TCP v 8080	80	Auto 🗸		Delete
			+		

7. Click Create Workload.

After the workload is created, you can view the running workload in the workload list.

----End

Verifying a Workload

After a workload is created, you can access the workload to check whether the deployment is successful.

In the preceding configuration, the NodePort mode is selected to access the workload by using **IP address:Port number**. If the access is successful, the workload is successfully deployed.

You can obtain the access mode from the **Access Mode** tab on the workload details page.

3_{Migration}

3.1 Migrating Container Images

3.1.1 Solution Overview

Background

Containers are growing in popularity. Many enterprises choose to build their own Kubernetes clusters. However, the O&M workload of on-premises clusters is heavy, and O&M personnel need to configure the management systems and monitoring solutions by themselves. For enterprises, managing a large number of images requires high O&M, labor, and management costs, and the efficiency is low.

SoftWare Repository for Container (SWR) manages container images that function on multiple architectures, such as Linux and Arm. Enterprises can migrate their image repositories to SWR to reduce costs.

This section describes three ways for migrating image repositories to SWR smoothly. You can select one as required.

Migration Solutions

Solution	Application Scenario	Precautions
Migrating Images to SWR Using Docker	Small quantity of images	• Disk storage leads to the timely deletion of local images and time-cost flushing.
Commands		• Docker daemon strictly restricts the number of concurrent pull/push operations, so high- concurrency synchronization cannot be performed.
		• Scripts are complex because HTTP APIs are needed to perform the operations that cannot be implemented through Docker CLI.
Migrating Images to SWR Using	A large number of images	• Many-to-many image repository synchronization is supported.
image- migrator		• Docker Registry V2-based image repositories (such as Docker Hub, Quay, and Harbor) can be migrated to SWR.
		• Memory- and network- dependent synchronization is fast.
		 Flushing the Blob information of synchronized images avoids repetition.
		• The number of concurrent synchronization tasks can be adjusted in the configuration file.
		 Automatically retrying failed synchronization tasks can resolve most network jitter during image synchronization.
		• Docker or other programs are not required.

 Table 3-1 Comparison of migration solutions and application scenarios

Solution	Application Scenario	Precautions
Migrating Images Across Clouds from Harbor to SWR	A customer deploys services in multiple clouds and uses Harbor as their image repository.	Only Harbor v1.10.5 and later versions are supported.

3.1.2 Migrating Images to SWR Using Docker Commands

Scenarios

SWR provides easy-to-use image hosting and efficient distribution services. If small quantity of images need to be migrated, enterprises can use the **docker pull/push** command to migrate images to SWR.

Procedure

Step 1 Pull images from the source repository.

Run the **docker pull** command to pull the images.

Example: docker pull nginx:latest

Run the **docker images** command to check whether the images are successfully pulled.

# docker images				
REPOSITORY	TAG	G IMAGE ID	CREATED	SIZE
nginx	latest	22f2bf2e2b4f	5 hours ago	22.8MB

Step 2 Push the images pulled in **Step 1** to SWR.

- Log in to the VM where the destination container is located and log in to SWR. For details, see Uploading an Image Through a Container Engine Client.
- 2. Tag the images.

docker tag [Image name:Tag name] [Image repository address]/ [Organization name]/[Image name:Tag name]

Example:

docker tag nginx:v1 swr.ap-southeast-1.myhuaweicloud.com/cloud-develop/nginx:v1

3. Run the following command to push the images to the destination image repository.

docker push [Image repository address]/[Organization name]/[Image name:Tag name]

Example:

docker push swr.ap-southeast-1.myhuaweicloud.com/cloud-develop/ nginx:v1 4. Check whether the following information is returned. If yes, the push is successful.

fbce26647e70: Pushed fb04ab8effa8: Pushed 8f736d52032f: Pushed 009f1d338b57: Pushed 678bbd796838: Pushed d1279c519351: Pushed f68ef921efae: Pushed v1: digest: sha256:0cdfc7910db531bfa7726de4c19ec556bc9190aad9bd3de93787e8bce3385f8d size: 1780

To view the pushed image, refresh the My Images page.

----End

3.1.3 Migrating Images to SWR Using image-migrator

To ensure that container images can be properly pulled after cluster migration and improve container deployment efficiency, you are advised to migrate self-built image registries to Huawei Cloud SoftWare Repository for Container (SWR).

image-migrator is an image migration tool that can automatically migrate images from a Docker registry built on Docker Registry v2 to SWR.

Preparations

Before the migration, prepare a server with kubectl installed for the connection between the source and destination clusters. The server must have at least 5 GB of local disk space and at least 8 GB of memory so that image-migrator can work properly and can store related data, such as data collected from the source cluster and recommendation data of the destination cluster.

image-migrator can run on Linux (x86 and Arm) and Windows.

Before using image-migrator in Linux, run the **chmod u+x** *Tool name* command (for example, **chmod u+x image-migrator-linux-amd64**) to grant the execute permission.

image- migrator	image-migrator is an image migration tool that can automatically migrate images from a Docker registry built on Docker Registry v2 to SWR, or from a registry on a third-party cloud to SWR.	Linux x86: https://ucs- migration.obs.cn- north-4.myhuaweicloud.com/ toolkits/image-migrator-linux-amd64 Linux Arm: https://ucs- migration.obs.cn- north-4.myhuaweicloud.com/ toolkits/image-migrator-linux-arm64 Windows: https://ucs- migration.obs.cn- north-4.myhuaweicloud.com/ toolkits/image-migrator-windows- amd64.exe
--------------------	--	---

Table 3-2 Obtaining the image-migrator package

How image-migrator Works



Figure 3-1 How image-migrator works

When using image-migrator to migrate images to SWR, you need to prepare two files. One is the registry access permission file **auth.json**. The two objects in the file are the accounts and passwords of the source and destination registries. The other is the image list file **images.json**, which consists of multiple image synchronization rules. Each rule contains a source repository (key) and a destination repository (value). Place these two files in the directory where image-migrator is located and run a simple command to migrate the image. The two files are described as follows:

• auth.json

auth.json is the registry access permission file. Each object is the username and password of a registry. Generally, you must have permission to pull images from and access image tags in the source registry and permission to push images to and create repositories in the destination registry. If you access a registry anonymously, you do not need to enter the username and password. Structure of the **auth.json** file:

```
"Source registry address": { },
"Destination registry address": {
    "username": "xxxxxx",
    "password": "xxxxxx",
    "insecure": true
  }
}
```

To be more specific:

 The Source registry address and Destination registry address can be in the registry or registry/namespace format matching that in images.json. The matched URL in images.json uses the corresponding username and password for image synchronization. The registry/namespace format is preferred.

If the destination registry address is in the *registry* format, you can obtain it from the SWR console. On the **Dashboard** page, click **Generate Login Command** in the upper right corner. The domain name at the end of the login command is the SWR registry address, for example, **swr.cnnorth-4.myhuaweicloud.com**. Note that the address varies depending on the region. Switch to the corresponding region to obtain the address. If the value is in the *registry/namespace* format, replace *namespace* with the organization name of SWR.

- **username**: (Optional) username. You can set it to a specific value or use a string of the \${env} or \$env type to reference an environment variable.
- password: (Optional) password. You can set it to a specific value or use a string of the \${env} or \$env type to reference an environment variable.
- insecure: (Optional) whether *registry* is an HTTP service. If yes, the value of insecure is true. The default value is false.

D NOTE

The username of the destination SWR registry is in the following format: *Regional project name@AK*. The password is the encrypted login key based on the AK and SK. For details, see **Obtaining a Long-Term Login Command**.

Example:

{

}

```
"quay.io/coreos": { },
"swr.cn-north-4.myhuaweicloud.com": {
"username": "cn-north-4@RVHVMX******",
"password": "cab4ceab4a1545*************",
"insecure": true
}
```

images.json

This file is essentially a list of images to migrate and consists of multiple image synchronization rules. Each rule contains a source repository (key) and a destination repository (value). The specific requirements are as follows:

- a. The largest unit that can be synchronized using one rule is a repository. An entire namespace or registry cannot be synchronized using one rule.
- b. The formats of the source and destination repositories are similar to those of the image URL used by the **docker pull** or **docker push** command (*registry/namespace/repository:tag*).
- c. Both the source and destination repositories (if the destination repository is not an empty string) contain at least *registry*/*namespace*/*repository*.
- d. The source repository cannot be empty. To synchronize data from one source repository to multiple destination repositories, you need to configure multiple rules.
- e. The destination repository name can be different from the source repository name. In this case, the synchronization function is similar to **docker pull + docker tag + docker push**.
- f. If the source repository field does not contain any tag, all tags of the repository will be synchronized to the destination repository. In this case, the destination repository field cannot contain a tag.
- g. If the source repository field contains a tag, only that tag in the source repository will be synchronized to the destination repository. If the destination repository does not contain a tag, the source tag will be used at the destination by default.
- h. If the destination repository is an empty string, the source image will be synchronized to the default namespace of the default registry. The repository and tag are the same as those of the source repository. The

default registry and namespace can be configured using command line parameters and environment variables.

Example:

{

}

```
"quay.io/coreos/etcd:1.0.0": "swr.cn-north-4.myhuaweicloud.com/test/etcd:1.0.0",
"quay.io/coreos/etcd": "swr.cn-north-4.myhuaweicloud.com/test/etcd",
"quay.io/coreos/etcd:2.7.3": "swr.cn-north-4.myhuaweicloud.com/test/etcd"
```

We provide **config**, a subcommand of image-migrator, to automatically obtain the images that are being used by workloads in a cluster. For details, see **Usage of image-migrator config**. After obtaining the **images.json** file, you can modify, add, or delete its content as needed.

How to Use image-migrator

NOTE

image-migrator can run on Linux (x86 and Arm) and Windows. The usage is similar in both environments. This section uses the Linux (x86) environment as an example.

If Linux (Arm) or Windows is used, replace **image-migrator-linux-amd64** in the following command with **image-migrator-linux-arm64** or **image-migrator-windows-amd64.exe**.

Run **./image-migrator-linux-amd64 -h** in the directory where image-migrator is located to learn about its usage.

- --auth: specifies the path of auth.json. By default, auth.json is stored in the directory where image-migrator is located.
- --images: specifies the path of images.json. By default, images.json is stored in the directory where image-migrator is located.
- --log: specifies the path for storing logs generated by image-migrator. The default value is image-migrator.log in the current directory of imagemigrator.
- --namespace: specifies the default namespace of the destination repository. That is, if the namespace of the destination repository is not specified in images.json, you can specify it when running the migration command.
- --registry: specifies the default registry of the destination repository. That is, if the registry of the destination repository is not specified in images.json, you can specify it when running the migration command.
- --retries: specifies the number of retries allowed when the migration fails. The default value is **3**.
- --workers: specifies the number of concurrent workers for image migration. The default value is **7**.

```
$ ./image-migrator-linux-amd64 -h
```

A Fast and Flexible docker registry image images tool implement by Go.

```
Usage:

image-migrator [flags]

Aliases:

image-migrator, image-migrator

Flags:

--auth string auth file path. This flag need to be pair used with --images. (default "./auth.json")

-h, --help help for image-migrator

--images string images file path. This flag need to be pair used with --auth (default "./images.json")

--log string log file path (default "./image-migrator.log")
```

--namespace string default target namespace when target namespace is not given in the images config file, can also be set with DEFAULT_NAMESPACE environment value --registry string default target registry url when target registry is not given in the images config file, can also be set with DEFAULT_REGISTRY environment value -r, --retries int times to retry failed tasks (default 3) -w, --workers int numbers of working goroutines (default 7) \$./image-migrator --workers=5 --auth=./auth.json --images=./images.json --namespace=test \ --registry=swr.cn-north-4.myhuaweicloud.com --retries=2 \$./image-migrator Start to generate images tasks, please wait ...

Example:

Start to handle images tasks, please wait ...

./image-migrator --workers=5 --auth=./auth.json --images=./images.json -namespace=test --registry=swr.cn-north-4.myhuaweicloud.com --retries=2

Images(38) migration finished, 0 images tasks failed, 0 tasks generate failed

The preceding command is used to migrate images in the **images.json** file to **swr.cn-north-4.myhuaweicloud.com/test**. If the migration fails, you can retry twice. A maximum of five images can be migrated at a time.

Usage of image-migrator config

The **config** subcommand of image-migrator can be used to obtain images used in cluster applications and generate the **images.json** file in the directory where image-migrator is located. You can run **./image-migrator-linux-amd64 config -h** to learn how to use the **config** subcommand.

- -k, --kubeconfig: specifies the location of kubeconfig files of kubectl. The default value is \$HOME/.kube/config. A kubeconfig file is used to configure access to a Kubernetes cluster. The file contains the authentication credential and endpoint (address) required for accessing a Kubernetes cluster. For details, see the Kubernetes documentation.
- -n, --namespaces: specifies the namespace of the image to be obtained. Multiple namespaces are separated by commas (,), for example, ns1,ns2,ns3. The default value is "", indicating that images of all namespaces are obtained.
- -t, --repo: specifies the destination registry address (registry/namespace).

\$./image-migrator-linux-amd64 config -h generate images.json

Usage: image-migrator config [flags]

Flags:

```
-h, --help help for config
```

-k, --kubeconfig string The kubeconfig of k8s cluster's. Default is the \$HOME/.kube/config. (default "/ root/.kube/config")

-n, --namespaces string Specify a namespace for information collection. If multiple namespaces are specified, separate them with commas (,), such as ns1,ns2. default("") is all namespaces -t, --repo string target repo,such as swr.cn-north-4.myhuaweicloud.com/test

Examples:

- Specify one namespace: ./image-migrator-linux-amd64 config -n default -t swr.cn
 - north-4.myhuaweicloud.com/test
- Specify multiple namespaces:

./image-migrator-linux-amd64 config -n default,kube-system -t swr.cnnorth-4.myhuaweicloud.com/test

If no namespace is specified, images of all namespaces are obtained:

./image-migrator-linux-amd64 config -t swr.cnnorth-4.myhuaweicloud.com/test

Procedure

Step 1 Prepare a registry access permission file **auth.json**.

Create an **auth.json** file and modify it based on the required format. If a registry is accessed anonymously, you do not need to enter information such as the username and password. Place the file in the directory where image-migrator is located.

Example:

ł	
L	"quay.io/coreos": { }, "swr.cn-north-4.myhuaweicloud.com": { "username": "cn-north-4@RVHVMX*****", "password": "cab4ceab4a1545**************, "insecure": true
	}
}	

For details about the parameters, see the **auth.json**.

Step 2 Prepare an image list file images.json.

- 1. Connect to the source cluster using kubectl. For details, see **Connecting to a Cluster Using kubectl**.
- 2. Run the **config** subcommand of image-migrator to generate an **images.json** file.

You can refer to the methods and examples in **Usage of image-migrator config** to obtain the images used in the source cluster applications without specifying a namespace, or by specifying one or multiple namespaces.

- 3. Modify the **images.json** file to make it meet the requirements in **images.json**.
- Step 3 Migrate images.

You can run the default **./image-migrator-linux-amd64** command to migrate images or configure image-migrator parameters as needed.

For example, run the following command:

./image-migrator-linux-amd64 --workers=5 --auth=./auth.json --images=./ images.json --namespace=test --registry=swr.cn-north-4.myhuaweicloud.com --retries=2

Example:

\$./image-migrator-linux-amd64 Start to generate images tasks, please wait ... Start to handle images tasks, please wait ... Images(38) migration finished, 0 images tasks failed, 0 tasks generate failed

Step 4 View the results.

After the preceding command is executed, information similar to the following is displayed:

Images(38) migration finished, 0 images tasks failed, 0 tasks generate failed

The preceding information indicates that 38 images have been migrated to SWR.

----End

3.1.4 Migrating Images Across Clouds from Harbor to SWR

Scenarios

If you are using a self-built Harbor registry on other cloud, you can replicate images from Harbor to SWR.

- 1. If Harbor can access SWR over the Internet, perform the replication as instructed in **Accessing SWR over the Internet**.
- 2. If Harbor accesses SWR through a VPC endpoint by using a private line, perform the replication as instructed in Accessing SWR Through a VPC Endpoint by Using a Private Line.

Background

Harbor is an open source enterprise-class Docker Registry server developed by VMware. It extends the Docker Distribution by adding the functionalities such as role-based access control (RBAC), image scanning, and image replication. Harbor has been widely used to store and distribute container images.

Accessing SWR over the Internet

Step 1 Configure a registry endpoint on Harbor.

NOTE

Harbor 1.10.5 and later can interconnect with Huawei Cloud SWR. You only need to set **Provider** to **Huawei SWR** when configuring an endpoint on Harbor. Harbor 2.4.1 is used as an example.

1. Add an endpoint.

Harbor	Qse	earch Harbor					
	«						
🖧 Projects		Registries					
ය Administration 쫑 Users	~	+ NEW ENDPOINT	BDIT X DELETE				
🖒 Robot Accounts		Name	Status	Endpoint URL	Τ	Provider	Verify Remote Cert
Registries					7		
Replications							
<pre> % Distributions </pre>				We couldn't fin	d any e	endpoints!	
🛇 Labels							
⊘ Project Quotas							
Interrogation Service	s						
i Garbage Collection							
Configuration							

2. Configure the following parameters.

New Registry Endpoint

Provider *	Huawei SWR 🗸
Name *	test
Description	
	li.
Endpoint URL *	https://swr .myhuaweiclou
Access ID	@CCRXJUTQ7QG
Access Secret	
Verify Remote Cert (j)	
	TEST CONNECTION CANCEL OK

- Provider: Select Huawei SWR.
- Name: Enter a name for the endpoint.
- Endpoint URL: Enter the public network domain name of SWR in the format of https://{SWR image registry address}. To obtain a registry address, log in to the SWR console, choose My Images and click Upload Through Client. In the displayed dialog box, view the image registry address in the current region.
- Access ID: Enter an access ID in the format of *Regional project* name@[AK].
- Access Secret: Enter an access secret. It is generated by using AK and SK.
 Access Secret will be used together with Access ID to generate a long-term login command. For details, see Obtaining a Long-Term Login Command.
- **Verify Remote Cert**: Deselect the option (recommended).

Step 2 Configure a replication rule.

1. Create a replication rule.

Harbor	Qse	arch Harbor					
	«						
🐣 Projects		Replicati	ons				
🗄 Logs							
🍰 Administration	~	+ NEW REPLI	CATION RULE	C REPLICATE	ACTIONS V		
密 Users				<u> </u>			
🖒 Robot Accounts		Name	Status T	Source registry	Replication Mode	Destination Registry:Namespace	Destination Repository Flattening
Registries							
C Replications						Y	
<pre><c distributions<="" pre=""></c></pre>					We	couldn't find any replication r	ules!
🛇 Labels							
⊘ Project Quotas							
O Interrogation Services							
i Garbage Collection							
Configuration							

- 2. Configure the following parameters.
 - **Name**: Enter a name for the rule.
 - **Replication mode**: Select **Push-based**, indicating that images are pushed from the local Harbor to a remote registry.
 - Source resource filter: Images in Harbor will be filtered to replicate based on the configured rule.
 - **Destination registry**: Select the endpoint created in **Step 1**.
 - Destination

Namespace: Enter an organization of SWR.

If the **Namespace** value contains a slash (/), for example, **ns1/test**, Huawei SWR considers **ns1** as the organization name and **test** as the path to the image. So **ns1/test** means the image name under the **ns1** organization is **test/***.

Flattening: Select **Flatten All Levels** to remove all hierarchy from the replicated images. For example, if the hierarchy of Harbor is **library/nginx** and the destination namespace is **dev-container**, **library/nginx** replicates to **dev-container/nginx**.

- Trigger Mode: Select Manual.
- Bandwidth: Set the maximum network bandwidth for each replication task. -1 indicates unlimited bandwidth.

For more information, see the Harbor official website.

Step 3 Select the replication rule and click **REPLICATE** to replicate images.

Image: Constraint of the second se	Harbor (Q Search Harbor
 ♣ Projects ■ Logs Administration ★ Administration ★ New REPLICATION RULE ■ Control Contro	«	
	🖧 Projects	Replications
C Robot Accounts Name Status T Source registry Replication Mode Destination Registry:Namespace	ஃ Administration ∽ 쫑 Users	+ NEW REPLICATION RULE
	🖒 Robot Accounts	Name Status $ au$ Source registry Replication Mode Destination Registry:Namespace
⑦ Registries ● test ⊘ Enabled Local push-based test : dev-container	Registries	o test ⊘ Enabled Local push-based test : dev-container
G Replications	C Replications	
«° Distributions	≪° Distributions	
S Labels Executions	🛇 Labels	Executions
⊘ Project Quotas	⊘ Project Quotas	
O Interrogation Services	O Interrogation Services	STOP

----End

Accessing SWR Through a VPC Endpoint by Using a Private Line

Step 1 Configure a VPC endpoint.

- Step 2 Obtain the private network IP address and domain name of the VPC. (By default, the domain name resolution rule is automatically added to Huawei Cloud VPCs, so you only need to configure hosts for non-Huawei Cloud endpoints.) You can query the IP address and domain name in **Private Domain Name** on the VPC endpoint details page.
- **Step 3** Configure a registry endpoint on Harbor.

NOTE

Harbor 1.10.5 and later can interconnect with Huawei Cloud SWR. You only need to set **Provider** to **Huawei SWR** when configuring an endpoint on Harbor. Harbor 2.4.1 is used as an example.

- 1. Add an endpoint.
- 2. Configure the following parameters.
 - Provider: Select Huawei SWR.
 - Name: Enter a name for the endpoint.
 - Endpoint URL: Enter the private network domain name of a VPC endpoint, which must start with https. In addition, the domain name mapping must be configured in the container where Harbor is located.
 - Access ID: Enter an access ID in the format of *Regional project name*@[*AK*].
 - Access Secret: Enter an access secret (SK). The AK and SK are used to generate a long-term login command. For details, see Obtaining a Long-Term Login Command.
 - Verify Remote Cert: Deselect the option.

Step 4 Configure a replication rule.

1. Create a replication rule.

Harbor	Q Search Harbor
~	«
D Projects	Replications
🗄 Logs	
🍰 Administration	Y Y Y Y PEPLICATION RULE C→ REPLICATE ACTIONS ✓
8 Users	
🖒 Robot Accounts	Name Status _T Source registry Replication Mode Destination Registry:Namespace Destination Repository Flattening
Registries	
🕒 Replications	
≪° Distributions	We couldn't find any replication rules!
🛇 Labels	
⊘ Project Quotas	
O Interrogation Services	
🗓 Garbage Collection	
Configuration	

2. Configure the following parameters.

New Replication Rule

Name *	test			
Description				1.
Replication mode	• Push-based	i) O Pul	ll-based (j)	
Source resource filter	Name:	library/*		í
	Tag:	matching \vee		i
	Label:	matching 🗸		⊻ (i)
	Resource:	image		~ i
Destination registry *	test-https://vp	cep-b6cf5b9d-2b46	-49a9-2b46	~
Destination	Namespace:	dev-container		i
	Flattening:	Flatten All Levels		~ (i)
Trigger Mode *	Manual			~
Bandwidth *	-1		КЪр	· (j
	- overnde (j	<i>y</i>		
			CANCEL	SAVE

- **Name**: Enter a name for the rule.
- **Replication mode**: Select **Push-based**, indicating that images are pushed from the local Harbor to a remote registry.
- **Source resource filter**: Images in Harbor will be filtered to replicate based on the configured rule.
- **Destination registry**: Select the endpoint created in **Step 3**.
- Destination
 Namespace: Enter an organization of SWR.

If the **Namespace** value contains a slash (/), for example, **ns1/test**, Huawei SWR considers **ns1** as the organization name and **test** as the path to the image. So **ns1/test** means the image name under the **ns1** organization is **test/***.

Flattening: Select **Flatten All Levels** to remove all hierarchy from the replicated images. For example, if the hierarchy of Harbor is **library/nginx** and the destination namespace is **dev-container**, **library/nginx** replicates to **dev-container/nginx**.

- Trigger Mode: Select Manual.
- **Bandwidth**: Set the maximum network bandwidth for each replication task. **-1** indicates unlimited bandwidth.

For more information, see the Harbor official website.

Step 5 Select the replication rule and click **REPLICATE** to replicate images.

Harbor	Q	Search Harbor			
	«				
🖧 Projects		Replications	S		
ය Administration 帝 Users	~	+ NEW REPLICATIO	ON RULE	IONSY	
ා. Robot Accounts		Name	Status T Source registry	Replication Mode	Destination Registry:Namespace
Registries		o test	Senabled Local	push-based	test : dev-container
Replications					
< Distributions					
🛇 Labels		Executions			
Project Quotas					
Interrogation Service	es	STOP			

----End

3.2 Migrating Kubernetes Clusters to CCE

3.2.1 Solution Overview

Application Scenarios

Containers are growing in popularity and Kubernetes simplifies containerized deployment. Many companies choose to build their own Kubernetes clusters. However, the O&M workload of on-premises clusters is heavy, and O&M personnel need to configure the management systems and monitoring solutions by themselves. This increases the labor costs while decreasing the efficiency.

In terms of performance, an on-premises cluster has poor scalability due to its fixed specifications. Auto scaling cannot be implemented in case of traffic surges, which may easily result in the insufficient or waste of cluster resources. In addition, disaster recovery risks are not considered for deploying an on-premises

cluster, leading to poor reliability. Once a fault occurs, the entire cluster may fail, resulting in serious production incidents.

Now you can address the preceding challenges by using CCE, a service that allows easy cluster management and flexible scaling, integrated with application service mesh and Helm charts to simplify cluster O&M and reduce operations costs. CCE is easy to use and delivers high performance, security, reliability, openness, and compatibility. This section describes the solution and procedure for migrating on-premises clusters to CCE.

Precautions

Compared with on-premises Kubernetes clusters, CCE clusters have multiple advantages. For details, see **Product Advantages**. There are some restrictions when using CCE clusters. For details, see **Notes and Constraints**. Evaluate the restrictions before using CCE clusters.

Migration Solution

This section describes a cluster migration solution, which applies to the following types of clusters:

- Kubernetes clusters built in local IDCs
- On-premises clusters built using multiple ECSs
- Cluster services provided by other cloud service providers
- CCE clusters that are no longer maintained and cannot be upgraded in place

Before the migration, analyze all resources in the source clusters and then determine the migration solution. Resources that can be migrated include resources inside and outside the clusters, as listed in the following table.

Category	Migration Object	Remarks
Resources inside a cluster	All objects in a cluster, including pods, jobs, Services, Deployments, and ConfigMaps.	You are not advised to migrate the resources in the velero and kube-system namespaces.
		• velero : Resources in this namespace are created by the migration tool and do not need to be migrated.
		• kube-system : Resources in this namespace are system resources. If this namespace of the source cluster contains resources created by users, migrate the resources on demand.
		CAUTION If you are migrating or backing up cluster resources in CCE, for example, from a namespace to another, do not back up Secret paas.elb. It is because secret paas.elb is periodically updated. After the backup is complete, the secret may become invalid when it is restored. As a result, network storage functions are affected.
	PersistentVolumes (PVs) mounted to containers	Due to restrictions of the restic tool, migration is not supported for the hostPath storage volume. For details about how to solve the problem, see Storage Volumes of the HostPath Type Cannot Be Backed Up.
Resources outside a cluster	On-premises image repository	Resources can be migrated to SoftWare Repository for Container (SWR).
	Non-containerized database	Resources can be migrated to Relational Database Service (RDS).
	Non-local storage, such as object storage	Resources can be migrated to Object Storage Service (OBS).

Table 3-3 Resources	that can	be migrated
---------------------	----------	-------------

Figure 3-2 shows the migration process. You can migrate resources outside a cluster as required.



Migration Process



The cluster migration process is as follows:

Step 1 Plan resources for the target cluster.

For details about the differences between CCE clusters and on-premises clusters, see **Key Performance Parameter** in **Resource Planning for the Target Cluster**. Plan resources as required and ensure that the performance configuration of the target cluster is the same as that of the source cluster.

Step 2 Migrate resources outside a cluster.

To migrate resources outside the cluster, see **Migrating Resources Outside a Cluster**.

Step 3 Install the migration tool.

After resources outside a cluster are migrated, you can use a migration tool to back up and restore application configurations in the source and target clusters. For details about how to install the tool, see **Installing the Migration Tool**.

Step 4 Migrate resources in the cluster.

Use Velero to back up resources in the source cluster to OBS and restore the resources in the target cluster. For details, see **Migrating Resources in a Cluster** (Velero).

• Backing Up an Application in the Source Cluster

To back up resources, use the Velero tool to create a backup object in the original cluster, query and back up cluster data and resources, package the data, and upload the package to the object storage that is compatible with the S3 protocol. Cluster resources are stored in the JSON format.

• Restoring the Application in the Target Cluster

During restoration in the target cluster, Velero specifies the temporary object bucket that stores the backup data, downloads the backup data to the new cluster, and redeploys resources based on the JSON file.

Step 5 Update resources accordingly.

After the migration, cluster resources may fail to be deployed. Update the faulty resources. The possible adaptation problems are as follows:

- Updating Images
- Updating Services
- Updating the Storage Class
- Updating Databases

Step 6 Perform additional tasks.

After cluster resources are properly deployed, verify application functions after the migration and switch service traffic to the target cluster. After confirming that all services are running properly, bring the source cluster offline.

----End

3.2.2 Resource Planning for the Target Cluster

CCE allows you to customize cluster resources to meet various service requirements. **Table 3-4** lists the key performance parameters of a cluster and provides the planned values. You can set the parameters based on your service requirements. It is recommended that the performance configuration be the same as that of the source cluster.

NOTICE

After a cluster is created, the resource parameters marked with asterisks (*) in **Table 3-4** cannot be modified.

Resourc e	Key Performanc e Parameter	Description	Example Value
Cluster	*Cluster Type	CCE supports multiple cluster types to meet diverse needs, including highly reliable, highly secure, commercial containers.	CCE standard cluster
		You can select CCE Standard Cluster or CCE Turbo Cluster as required.	
		For more details, see Comparison Between Cluster Types.	
	*Network Model Select VPC network or Tun for your CCE standard clust For details about the differe container network models,		VPC network
	*Number of master nodes	Select the number of master nodes, also known as control plane nodes. These nodes are hosted on CCE and run Kubernetes components like kube- apiserver, kube-controller-manager, and kube-scheduler.	3 Masters
		• 3 Masters : Three master nodes will be created to ensure high availability.	
		• Single : Only one master node will be created in your cluster.	
Node	Node Type	Select a node type based on service requirements. The available node flavors will then be displayed in the Specifications area for you to choose from.	Elastic Cloud Server (VM)
	Node Specification	Select a node flavor that best fits your service needs.	4 vCPUs 8 GiB
	s (vary depending on the actual region)	For optimal cluster component performance, you are advised to choose a node with at least 4 vCPUs and 8 GiB of memory.	
	OS	Select an OS for the nodes.	EulerOS
	System Disk	System disk used by the node OS. The value ranges from 40 GiB to 1024 GiB. The default value is 50 GiB.	50 GiB

 Table 3-4 Example of CCE cluster planning

Resourc e	Key Performanc e Parameter	Description	Example Value
	Data Disk	Add a data disk for container runtime and kubelet. The disk size ranges from 20 GiB to 32768 GiB. The default value is 100 GiB.	100 GiB

3.2.3 Procedure

3.2.3.1 Migrating Resources Outside a Cluster

If your migration does not involve resources outside a cluster listed in **Table 3-3** or you do not need to use other services to update resources after the migration, skip this section.

Migrating Container Images

To ensure that container images can be properly pulled after cluster migration and improve container deployment efficiency, you are advised to migrate private images to SoftWare Repository for Container (SWR). CCE works with SWR to provide a pipeline for automated container delivery. Images are pulled in parallel, which greatly improves container delivery efficiency.

Manually migrate container images.

- **Step 1** Remotely log in to any node in the source cluster and run the **docker pull** command to pull all images to the local host.
- **Step 2** Log in to the SWR console, click **Login Command** in the upper right corner of the page, and copy the command.
- **Step 3** Run the copied login command on the node.

The message "Login Succeeded" will be displayed upon a successful login.

Step 4 Add tags to all local images.

docker tag [Image name 1:tag 1] [Image repository address] [Organization name] [Image name 2.tag 2]

- *[Image name 1:tag 1]*: name and tag of the local image to be pulled.
- *[Image repository address]*: You can obtain the image repository address on the SWR console.
- *[Organization name]*: Enter the name of the organization you created on the SWR console.
- *{Image name 2.Tag 2}*: image name and tag displayed on the SWR console.

The following is an example:

docker tag nginx:v1 swr.ap-southeast-1.myhuaweicloud.com/cloud-develop/mynginx:v1

Step 5 Run the **docker push** command to upload all local container image files to SWR. docker push [Image repository address] [Organization name] [Image name 2:tag 2]

The following is an example:

docker push swr.ap-southeast-1.myhuaweicloud.com/cloud-develop/mynginx:v1

----End

Migrating Databases and Storage (On-Demand)

You can determine whether to use **Relational Database Service (RDS)** and **Object Storage Service (OBS)** based on your production requirements. After the migration is complete, reconfigure the database and storage for applications in the target CCE cluster.

Database migration

If your database is deployed without using containers and needs to be migrated to the cloud, **Data Replication Service (DRS)** can help you do this. DRS provides multiple capabilities, including online migration, backup migration, real-time disaster recovery, data synchronization, and data subscription. Contact O&M or development personnel to migrate the database. For details, see **Migrating Databases Across Cloud Platforms**. After the migration is complete, interconnect with the database by following the procedure described in **Updating Databases**.

Storage migration

If your cluster has connected to an object storage service and needs to be migrated to the cloud, **Object Storage Migration Service (OMS)** can help you migrate data to OBS. Other storage classes are not supported by official tools.

Contact O&M or development personnel to migrate object storage data. For details, see **Creating a Migration Task**. After the migration is complete, attach the object storage to the application by referring to **Using an Existing OBS Bucket Through a Static PV**.

NOTE

Currently, you can use OMS to migrate object storage data from AWS China, Alibaba Cloud, Microsoft Azure, Baidu Cloud, Kingsoft Cloud, Ucloud, QingCloud, Qiniu Cloud, and Tencent Cloud to Huawei Cloud **OBS**.

3.2.3.2 Installing the Migration Tool

Velero is an open-source backup and migration tool for Kubernetes clusters. With restic's PV data backup capabilities, Velero can back up Kubernetes resource objects (such as Deployments, jobs, Services, and ConfigMaps) in the source cluster and data in PVs mounted to pods and uploaded them to object storage. If a disaster occurs or migration is required, the target cluster can obtain the corresponding backup data from the object storage using Velero and restore cluster resources as required.

According to **Migration Solution**, you will need to prepare temporary object storage to store backup files before the migration. Velero supports OBS as the object storage. For details about how to deploy Velero, see **Installing Velero**.

Prerequisites

• The Kubernetes version of the source on-premises cluster must be 1.10 or later, and the cluster can use DNS and Internet services properly.

- If you are using OBS to store backup files, you must obtain the AK/SK of a user who can operate OBS. For details, see Obtaining Access Keys (AK/SK).
- The target CCE cluster has been created.
- The source cluster and target cluster must each have at least one idle node. It is recommended that the node specifications be 4 vCPUs and 8 GiB memory or higher.

Installing Velero

You need to first go to the OBS console and create a bucket named **velero** for storing backup files. The bucket name can be a custom one, which must be specified when you install Velero. Otherwise, the bucket cannot be accessed and the backup will fail. For details, see **Step 5**.

NOTICE

- Velero instances need to be installed and deployed in both the **source and target clusters** and are used for backup and restoration, respectively. The installation procedures are the same.
- The master node of a CCE cluster does not support remote login ports. You can use kubectl to install Velero.
- If there are a large number of resources to back up, you are advised to adjust the CPU and memory resources of Velero and node-agent to 1 vCPU and 1 GiB of memory or higher. For details, see **Backup Tool Resources Are Insufficient**.
- The object storage bucket for storing backup files must be empty.

Then, you can download the latest, stable binary file from https://github.com/ vmware-tanzu/velero/releases. The following uses Velero 1.15.1 as an example. The process of installing Velero in the source cluster is identical to the target cluster.

- **Step 1** Log in to a VM that can access the public network and use kubectl to access the cluster where Velero is to be installed.
- **Step 2** Download the binary file of **Velero 1.15.1**. wget https://github.com/vmware-tanzu/velero/releases/download/v1.15.1/velero-v1.15.1-linux-amd64.tar.gz
- Step 3 Install the Velero client. tar -xvf velero-v1.15.1-linux-amd64.tar.gz cp ./velero-v1.15.1-linux-amd64/velero /usr/local/bin
- **Step 4** Create the access key file **credentials-velero** for the backup object storage. vim credentials-velero

Replace the AK/SK in the file as required. If OBS is used, you can obtain the AK/SK by referring to **Obtaining Access Keys (AK/SK)**. OBS_ACCESS_KEY=<Your AK> OBS_SECRET_KEY=<Your SK>

Step 5 Deploy the Velero server. Change the value of --bucket to the name of the created object storage bucket. In this example, the bucket name is velero. For more details about custom installation parameters, see Customize Velero Install.
velero install \
--provider huawei.com/huaweicloud \

Issue 01 (2025-07-11) Copyright © Huawei Cloud Computing Technologies Co., Ltd.

--plugins swr.ap-southeast-1.myhuaweicloud.com/huaweiclouddeveloper/velero-plugin-for-

- huaweicloud:v1.0.0 \ --bucket velero \
- --secret-file ./credentials-velero \
- --use-node-agent \
- --use-volume-snapshots=false \

--backup-location-config endpoint=obs.ap-southeast-1.myhuaweicloud.com

Table 3-5 Installation parameters of Velero

Parameter	Description
provider	Component provider.
plugins	Container image address of the component.
bucket	Name of the object storage bucket for storing backup files. The bucket must be created in advance.
secret-file	Secret file for accessing the object storage, which is, the credentials-velero file created in Step 4 .
use-node- agent	Whether to enable PV data backup. You are advised to enable this function. Otherwise, storage volume resources cannot be backed up.
use-volume- snapshots	Whether to create a VolumeSnapshotLocation object for PV snapshot, which requires support from the snapshot program. Set this parameter to false .
backup- location- config	 OBS bucket configuration. endpoint: API access address of the OBS bucket. The value of this parameter depends on the region where the OBS bucket is located. For example, if the region is Hong Kong, China (apsoutheast-1), the parameter value is obs.apsoutheast-1.myhuaweicloud.com.

Step 6 View the pod status. By default, a namespace named **velero** is created for the Velero instance.

kubectl get pod -n velero

Information similar to the following is displayed:

NAMEREADYSTATUSRESTARTSAGEnode-agent-rn29c1/1Running016svelero-c9ddd56-tkzpk1/1Running016s

NOTE

To prevent memory insufficiency during backup in the actual production environment, you are advised to change the CPU and memory allocated to node-agent and Velero by referring to **Backup Tool Resources Are Insufficient**.

Step 7 Add environment variables to the Velero instance to read the key file for accessing the OBS bucket.

kubectl patch deployment velero -n velero --patch \ '{"spec":{"template":{"spec":{"containers":[{"name": "velero", "env": [{"name": "HUAWEI_CLOUD_CREDENTIALS_FILE", "value": "/credentials/cloud"}]}}}}}

Step 8 Check the interconnection between Velero and OBS and ensure that the status is **Available**.

velero backup-location get

Information similar to the following is displayed:

NAMEPROVIDERBUCKET/PREFIXPHASELAST VALIDATEDACCESS MODEDEFAULTdefault***veleroAvailable2025-06-04 17:40:55 +0800 CSTReadWritetrue

----End

3.2.3.3 Migrating Resources in a Cluster (Velero)

Application Scenarios

WordPress is used as an example to describe how to migrate an application from an on-premises Kubernetes cluster to a CCE cluster. The WordPress application consists of the WordPress and MySQL components, which are containerized. The two components are bound to two local storage volumes of the local type, respectively, and provide external access through the NodePort Services.

Before the migration, you can use a browser to access the WordPress site, create a site named **Migrate to CCE**, and publish an article for verifying the integrity of PV data after the migration. The article published in WordPress will be stored in the **wp_posts** table of the MySQL database. If the migration is successful, all contents in the database will be migrated to the new cluster. You can then verify the migration results of the PV data.

Prerequisites

- Before the migration, the abnormal pod resources in the source cluster have been cleared. If a pod with PVC mounted becomes abnormal, the PVC will be in the pending state after the migration.
- The CCE cluster does not have the same resources as the source cluster because Velero does not restore the same resources by default.
- The container images have been migrated to SWR so that they can be properly pulled after the migration.
- CCE does not support EVS disks of the **ReadWriteMany** type. If resources of this type exist in the source cluster, change the storage type to **ReadWriteOnce**.
- Velero cannot back up or restore hostPath volumes. For details, see Limitations. To back up storage volumes of this type, replace the hostPath volumes with local volumes by referring to Storage Volumes of the HostPath Type Cannot Be Backed Up. If there is a hostPath volume in a backup task, the backup of this volume will be automatically skipped and a warning message will be generated. This will not cause a backup failure.

Backing Up an Application in the Source Cluster

Step 1 (Optional) To back up the data of a specified storage volume in a pod, add an annotation to the pod. The annotation template is as follows: kubectl -n <namespace> annotate <pod/pod_name> backup.velero.io/backup-volumes=<volume_name_1>,<volume_name_2>,...

- <namespace>: namespace where the pod is located
- <pod_name>: pod name
- <volume_name>: name of the PV mounted to the pod. You can run the describe statement to obtain the pod information. The Volume field indicates the names of all PVs mounted to the pod.

You can add annotations to the WordPress pod **wordpress-758fbf6fc7-s7fsr** and MySQL pod **mysql-5ffdfbc498-c45lh**. These pods are in the default namespace **default**, so the **-n <NAMESPACE>** parameter can be omitted.

kubectl annotate pod/**wordpress-758fbf6fc7-s7fsr** backup.velero.io/backup-volumes=**wp-storage** kubectl annotate pod/**mysql-5ffdfbc498-c45lh** backup.velero.io/backup-volumes=**mysql-storage**

- **Step 2** Back up the application. During the backup, you can specify resources based on parameters. If no parameter is added, the entire cluster resources are backed up by default. For details about the parameters, see **Resource filtering**.
 - --default-volumes-to-fs-backup: indicates that the PV backup tool is used to back up all storage volumes attached to a pod. hostPath volumes are not supported. If this parameter is not specified, the storage volume specified by annotation in Step 1 is backed up by default. This parameter is available only when --use-node-agent is specified during Velero installation. velero backup create <backup-name> --default-volumes-to-fs-backup
 - --include-namespaces: backs up resources in a specified namespace. velero backup create <backup-name> --include-namespaces <namespace>
 - --include-resources: backs up the specified resources. velero backup create <backup-name> --include-resources deployments
 - --selector: backs up resources that match the selector. velero backup create <backup-name> --selector <key>=<value>

In this section, resources in the namespace **default** are backed up. **wordpressbackup** is the backup name. Specify the same backup name when restoring applications. An example is as follows:

velero backup create wordpress-backup --include-namespaces default --default-volumes-to-fs-backup

If information similar to the following is displayed, the backup task has been created:

Backup request "wordpress-backup" submitted successfully. Run `velero backup describe wordpress-backup` or `velero backup logs wordpress-backup` for more details.

Step 3 Check the backup status.

velero backup get

 Information similar to the following is displayed:

 NAME
 STATUS
 ERRORS
 WARNINGS
 CREATED
 EXPIRES
 STORAGE

 LOCATION
 SELECTOR
 wordpress-backup
 Completed
 0
 2021-10-14
 15:32:07
 +0800
 CST
 29d
 default

 <none>

In addition, you can go to the object bucket to view the backup files. The **backups** path is for the application resource backup, and the other is for the PV data backup.

----End

Restoring the Application in the Target Cluster

The storage infrastructure of an on-premises cluster is different from that of a cloud cluster. After the cluster is migrated, PVs cannot be mounted to pods.

Therefore, during the migration, update the storage class of the target cluster to shield the differences of underlying storage interfaces between the two clusters when creating a workload and request storage resources of the corresponding type. For details, see **Updating the Storage Class**. If you migrate data using Object Storage Migration Service (OMS), you can mount object storage buckets to the application pods by referring to **Using an Existing OBS Bucket Through a Static PV**.

Step 1 Use kubectl to access the CCE cluster and create a storage class with the same name as that of the source cluster.

In this example, the storage class name of the source cluster is **local** and the storage type is local disk. Local disks completely depend on the node availability. The data DR performance is poor. When the node is unavailable, the existing storage data is affected. Therefore, the storage resources in the CCE cluster should be EVS volumes, and the backend storage media should be SAS disks.

NOTE

- When an application containing PV data is restored in a CCE cluster, the defined storage class dynamically creates storage resources (such as EVS volumes) based on the PVC and mounts the resources to the application.
- You can modify the storage resources of the cluster as needed, without being restricted to EVS volumes. To mount other types of storage, such as file storage and object storage, see **Updating the Storage Class**.

An example of the YAML file of the source cluster is as follows:

apiVersion: storage.k8s.io/v1 kind: StorageClass metadata: name: local provisioner: kubernetes.io/no-provisioner volumeBindingMode: WaitForFirstConsumer

An example of the YAML file of the target cluster is as follows:

- allowVolumeExpansion: true apiVersion: storage.k8s.io/v1 kind: StorageClass metadata: **name: local** selfLink: /apis/storage.k8s.io/v1/storageclasses/csi-disk parameters: csi.storage.k8s.io/csi-driver-name: disk.csi.everest.io csi.storage.k8s.io/fstype: ext4 everest.io/disk-volume-type: **SAS** everest.io/passthrough: "true" provisioner: everest-csi-provisioner reclaimPolicy: Delete volumeBindingMode: Immediate
- **Step 2** Create a Restore object using Velero, specify a backup named **wordpress-backup**, and restore the WordPress application in the CCE cluster. velero restore create --from-backup **wordpress-backup**

You can run the **velero restore get** statement to check the application restoration.

Step 3 After the restoration is complete, check whether the application is running properly. If other adaptation problems may occur, rectify the fault by following the procedure described in **Updating Resources Accordingly**.

----End

3.2.3.4 Updating Resources Accordingly

Updating Images

The WordPress and MySQL images used in this example can be pulled from SWR. Therefore, the image pull failure (ErrImagePull) will not occur. If the application to be migrated is created from a private image, perform the following operations to update the image:

- **Step 1** Migrate the image resources to SWR. For details, see **Uploading an Image Through a Container Engine Client**.
- **Step 2** Log in to the SWR console and obtain the image path used after the migration.

The image path is in the following format: 'swr.{Region}.myhuaweicloud.com/{Organization name}/{Image name}:{Tag name}'

- **Step 3** Run the following command to modify the workload and replace the **image** field in the YAML file with the image path: kubectl edit deploy wordpress
- **Step 4** Check the running status of the workload.

----End

Updating Services

After the cluster is migrated, the Service of the source cluster may fail to take effect. You can perform the following operations to update the Service. If ingresses are configured in the source cluster, connect the new cluster to ELB again after the migration. For details, see **Creating a LoadBalancer Ingress Using kubectl**.

- **Step 1** Connect to the cluster using kubectl.
- **Step 2** Edit the YAML file of the corresponding Service to change the Service type and port number.

kubectl edit svc wordpress

To update load balancer resources, connect to ELB again. Add the annotations by following the procedure described in Using kubectl to Create a Service (Using an Existing Load Balancer).

```
annotations:
kubernetes.io/elb.class: union # Shared load balancer
kubernetes.io/elb.id: 9d06a39d-xxxx-xxxx-c204397498a3 # Load balancer ID, which can be queried
on the ELB console.
kubernetes.io/elb.session-affinity-mode: SOURCE_IP # Enable the sticky session based on the source IP
address.
```

Step 3 Use a browser to check whether the Service is available.

----End

Updating the Storage Class

As the storage infrastructures of clusters may be different, storage volumes cannot be mounted to the target cluster. You can use either of the following methods to update the volumes:

NOTICE

Both update methods can be performed only before the application is restored in the target cluster. Otherwise, PV data resources may fail to be restored. In this case, use Velero to restore applications after the storage class update is complete. For details, see **Restoring the Application in the Target Cluster**.

Method 1: Creating a ConfigMap mapping

Step 1 Create a ConfigMap in the CCE cluster and map the storage class used by the source cluster to the default storage class of the CCE cluster.

```
apiVersion: v1
kind: ConfigMap
metadata:
name: change-storageclass-plugin-config
namespace: velero
labels:
app.kubernetes.io/name: velero
velero.io/plugin-config: "true"
velero.io/plugin-config: "true"
velero.io/change-storage-class: RestoreItemAction
data:
{Storage class name01 in the source cluster}: {Storage class name01 in the target cluster}
{Storage class name02 in the source cluster}: {Storage class name02 in the target cluster}
```

Step 2 Run the following command to apply the ConfigMap configuration:

\$ kubectl create -f change-storage-class.yaml configmap/change-storageclass-plugin-config created

----End

Method 2: Creating a storage class with the same name

Step 1 Run the following command to query the default storage class supported by CCE: kubectl get sc

Information similar to the following is displayed:								
	csi-disk	everest-csi-provisioner	Delete	Immediate	true		77d	
	csi-disk-dss	everest-csi-provisioner	Delete	Immediate	true		77d	
	csi-disk-topolog	gy everest-csi-provisioner	Delete	WaitForFirs	tConsumer	true		77d
	csi-local	everest-csi-provisioner	Delete	Immediate	false		19d	
	csi-local-topolo	gy everest-csi-provisioner	Delete	WaitForFirs	tConsumer	true		19d
	csi-nas	everest-csi-provisioner	Delete	Immediate	true		77d	
	csi-obs	everest-csi-provisioner	Delete	Immediate	false		77d	
	csi-sfs	everest-csi-provisioner	Delete	Immediate	false		77d	
	csi-sfsturbo	everest-csi-provisioner	Delete	Immediate	true		77d	

Table 3-6 Storage classes

Storage Class	Storage Resource
csi-disk	EVS disk
csi-disk-topology	EVS disk with delayed binding
csi-disk-dss	DSS disk
csi-local	Local PV
csi-local-topology	Local PV with delayed binding

Storage Class	Storage Resource
csi-nas	File system (SFS 1.0)
csi-sfs	General purpose file system (SFS 3.0 Capacity-Oriented)
csi-obs	OBS bucket
csi-sfsturbo	SFS Turbo file system

Step 2 Run the following command to export the required storage class details in YAML format:

kubectl get sc <storageclass-name> -o=yaml

Step 3 Copy the YAML file and create a new storage class.

Change the storage class name to the name used in the source cluster to call basic storage resources of the cloud.

The YAML file of csi-obs is used as an example. Delete the unnecessary information in italic under the metadata field and modify the information in bold. You are advised not to modify other parameters. apiVersion: storage.k8s.io/v1 kind: StorageClass metadata: creationTimestamp: "2021-10-18T06:41:36Z" name: <your_storageclass_name> # Use the name of the storage class used in the source cluster. resourceVersion: "747" selfLink: /apis/storage.k8s.io/v1/storageclasses/csi-obs uid: 4dbbe557-ddd1-4ce8-bb7b-7fa15459aac7 parameters: csi.storage.k8s.io/csi-driver-name: obs.csi.everest.io csi.storage.k8s.io/fstype: s3fs everest.io/obs-volume-type: STANDARD provisioner: everest-csi-provisioner reclaimPolicy: Delete volumeBindingMode: Immediate

NOTE

- SFS Turbo file systems cannot be directly created using StorageClass. Go to the SFS Turbo console to create SFS Turbo file systems that belong to the same VPC subnet and have inbound ports (111, 445, 2049, 2051, 2052, and 20048) enabled in the security group.
- CCE does not support EVS disks of the ReadWriteMany type. If resources of this type exist in the source cluster, change the storage type to **ReadWriteOnce**.
- **Step 4** Restore the cluster application by referring to **Restoring the Application in the Target Cluster** and check whether the PVC is successfully created.

kubectl get pvc

In the command output, the **VOLUME** column indicates the name of the PV automatically created using the storage class. NAME STATUS VOLUME CAPACITY ACCESS MODES STORAGECLASS AGE pvc Bound pvc-4c8e655a-1dbc-4897-ae6c-446b502f5e77 5Gi RWX local 13s

----End

Updating Databases

In this example, the database is a local MySQL database and does not need to be reconfigured after the migration. If you use **DRS** to migrate a local database to RDS, configure database access based on site requirements after the migration.

- If the RDS instance is in the same VPC as the CCE cluster, it can be accessed using the private IP address. Otherwise, it can only be accessed only through public networks by binding an EIP. You are advised to use the private network access mode for high security and good RDS performance.
- Ensure that the inbound rule of the security group to which RDS belongs has been enabled for the cluster. Otherwise, the connection will fail.
- **Step 1** Log in to the RDS console and obtain the private IP address and port number of the DB instance on the **Basic Information** page.
- **Step 2** Run the following command to modify the WordPress workload: kubectl edit deploy wordpress

Set the environment variables in the **env** field.

- **WORDPRESS_DB_HOST**: address and port number used for accessing the database, that is, the internal network address and port number obtained in the previous step.
- WORDPRESS_DB_USER: username for accessing the database.
- WORDPRESS_DB_PASSWORD: password for accessing the database.
- **WORDPRESS_DB_NAME**: name of the database to be connected.
- Step 3 Check whether the RDS database is properly connected.

----End

3.2.3.5 Performing Additional Tasks

Verifying Application Functions

Cluster migration involves full migration of application data, which may cause intra-application adaptation problems. In this example, after the cluster is migrated, the redirection link of the article published in WordPress is still the original domain name. If you click the article title, you will be redirected to the application in the source cluster. Therefore, search for the original domain name in WordPress and replace it with the new domain name, change the values of **site_url** and primary URL in the database. For details, see **Changing The Site URL**.

Access the new address of the WordPress application. If the article published before the migration is displayed, the data of the persistent volume is successfully restored.

🕅 🚳 Migrate to CCE 🖌 Customize 👎 0 🕂 New	,	Howdy, demo 📕 🍳
	MIGRATE TO CCE Just another WordPress site	
	Demo A demo to migrate WordPress from Kubernetes in IDC to CCE.	
	Published December 10, 2021 <u>Edit</u> Categorized as <u>Uncategorized</u>	

Switching Live Traffic to the Target Cluster

O&M personnel switch DNS to direct live traffic to the target cluster.

- DNS traffic switching: Adjust the DNS configuration to switch traffic.
- Client traffic switching: Upgrade the client code or update the configuration to switch traffic.

Bringing the Source Cluster Offline

After confirming that the service on the target cluster is normal, bring the source cluster offline and delete the backup files.

- Verify that the service on the target cluster is running properly.
- Bring the source cluster offline.
- Delete backup files.

3.2.3.6 Troubleshooting

Storage Volumes of the HostPath Type Cannot Be Backed Up

Both hostPath and local volumes are local storage volumes. However, restic, which is integrated into Velero, cannot back up hostPath PVs. It only supports the local type. So, you need to replace the storage volumes of the hostPath type with the local type in the source cluster.

NOTE

It is recommended that Local volumes be used in Kubernetes v1.10 or later and can only be statically created. For details, see **local**.

Step 1 Create a storage class for the Local volume.

Example YAML: apiVersion: storage.k8s.io/v1 kind: StorageClass metadata: name: local provisioner: kubernetes.io/no-provisioner volumeBindingMode: WaitForFirstConsumer
Step 2 Change the **hostPath** field to the **local** field, specify the original local disk path of the host machine, and add the **nodeAffinity** field.

```
Example YAML:
apiVersion: v1
kind: PersistentVolume
metadata:
 name: mysql-pv
 labels:
  app: mysql
spec:
 accessModes:
 - ReadWriteOnce
 capacity:
 storage: 5Gi
 storageClassName: local # Storage class created in the previous step
 persistentVolumeReclaimPolicy: Delete
 local:
  path: "/mnt/data"  # Path of the attached local disk
 nodeAffinity:
  required:
   nodeSelectorTerms:
    - matchExpressions:
     - key: kubernetes.io/hostname
      operator: Exists
```

Step 3 Run the following commands to verify the creation result:

kubectl get pv

Information similar to the following is displayed:

NAME	CAPACIT	Y ACCESS N	IODES RE	ECLAIM POLICY	STATUS	CLAIM	STORAGECLASS
REASON	AGE						
mysql-pv	5Gi	RWO	Delete	Available		local	3s

----End

Backup Tool Resources Are Insufficient

In a production environment with numerous resources to be backed up, using the backup tool with the default resource size may lead to insufficient resources. In such case, take the following steps to adjust the CPU and memory resources of Velero and restic:

Before installing Velero:

You can specify the size of resources used by Velero and restic when **installing Velero**.

The following is an example of installation parameters:

```
velero install \
--velero-pod-cpu-request 500m \
--velero-pod-mem-request 1Gi \
--velero-pod-cpu-limit 1000m \
--velero-pod-mem-limit 1Gi \
--use-node-agent \
--node-agent-pod-cpu-request 500m \
```

```
--node-agent-pod-cpu-request 500m --node-agent-pod-mem-request 1Gi \
```

```
--node-agent-pod-cpu-limit 1000m \
```

```
--node-agent-pod-mem-limit 1Gi
```

After Velero is installed:

Step 1 Edit the YAML files of the Velero and node-agent workloads in the **velero** namespace.

kubectl edit deploy velero -n velero kubectl edit ds node-agent -n velero

Step 2 Modify the resource size under the **resources** field. The modification is the same for the Velero and restic workloads, as shown in the following:

resources: limits: cpu: "1" memory: 1Gi requests: cpu: 500m memory: 1Gi

----End

4 DevOps

4.1 Installing and Deploying Jenkins on CCE

4.1.1 Solution Overview

What Is Jenkins?

Jenkins is an open source continuous integration (CI) tool that provides userfriendly GUIs. It originates from Hudson and is used to automate all sorts of tasks related to building, testing, and delivering or deploying software.

Jenkins is written in Java and can run in popular servlet containers such as Tomcat, or run independently. It is usually used together with the version control tools (or SCM tools) and build tools. Jenkins supports various languages and is compatible with third-party build tools, such as Maven, Ant, and Gradle. It seamlessly integrates with common version control tools, such as SVN and Git, and can directly connect to source code hosting services, such as GitHub.

Notes and Constraints

- This solution can be deployed only in CCE clusters. It is not supported in DeC.
- CCE does not provide maintenance and support for Jenkins. The maintenance is provided by the developers.

Solution Architecture

You can install Jenkins using the following methods:

- You can use a single Master to install Jenkins. The Master handles jobs and builds and releases services. However, security risks may exist.
- Another one is to use Master+Agents. Master schedules build jobs to Agents for execution, and monitors Agent status. Agents execute build jobs dispatched by the Master and return the job progress and result.

You can install the Master and Agents on VMs, containers, or combination of the two. For details, see **Table 4-1**.

Deployment Mode	Master	Agent	Advantages and Disadvantages
Single Master	VMs	-	 Advantage: Localized construction is easy to operate. Disadvantage: Job management and execution are performed on the same VM and the security risk is high.
Single Master	Containers	-	 Advantage: Kubernetes containers support self- healing. Disadvantage: Job management and execution are not isolated. Security risks exist.
Master+Agents	VMs	VMs	 Advantage: Job management and execution are isolated and the security risk is low. Disadvantage: Agents are fixed. Resources cannot be scheduled and the resource utilization is low and the maintenance cost is high.
		Containers (Kubernetes cluster)	 Advantage: Containerized Agents can be fixed or dynamic. Kubernetes schedules the dynamic Agents, improving the resource utilization. Jobs can be evenly allocated based on the scheduling policy, which is easy to maintain. Disadvantage: The Master may break down and the recovery cost is high.

Deployment Mode	Master	Agent	Advantages and Disadvantages
Master+Agents	Containers (Kubernete s cluster)	Containers (Kubernetes cluster)	 Advantage: Containerized Agents can be fixed or dynamic. Kubernetes schedules the dynamic Agents, improving the resource utilization. The Master is self-healing and the maintenance cost is low. Agents and the Master can be deployed in the same cluster or in different clusters. Disadvantage: The system is complex and the environment is difficult to set up.

In this section, Jenkins is installed with the containerized Master and Agents. Kubernetes schedules the dynamic Agents. For details about the architecture, see **Figure 4-1**.

- The Master handles jobs. Install Kubernetes add-ons on the Master to use the Kubernetes platform resources.
- The Kubernetes platform generates pods for Agents to execute jobs. When a job is scheduled on the Master, the Master sends a request to the Kubernetes platform using the Kubernetes add-on. After receiving the request, Kubernetes builds a pod using the pod template to send requests to the Master. After the Master is successfully connected, you can execute the job on the pod.



Figure 4-1 Installing Jenkins on Kubernetes

Procedure

Step 1 Installing and Deploying Jenkins Master

Jenkins Master is deployed in the CCE cluster using container images.

Step 2 Configuring Jenkins Agent

Jenkins can fix Agents in the cluster or use the pipeline to interconnect with CCE to provide pods for Agents to execute jobs. The dynamic Agents use Kubernetes add-ons to configure cluster authentication and user permissions.

Step 3 Using Jenkins to Build a Pipeline

The Jenkins pipeline interconnects with SWR and calls **docker build/login/push** commands in Agents to package and push images automatically.

You can also use pipelines to deploy and upgrade Kubernetes resources (such as Deployments, Services, ingresses, and jobs).

----End

4.1.2 Resource and Cost Planning

NOTICE

The fees listed here are estimates. The actual fees will be displayed on the Huawei Cloud console.

The required resources are as follows:

Resource	Description	Quanti ty	Estimated Fee
Cloud Container Engine (CCE)	 Pay-per-use recommended Cluster type: CCE cluster CCE cluster version: v1.25 Cluster scale: 50 nodes HA: Yes 	1	USD0.42/hour
VM	 Pay-per-use recommended VM type: General computing-plus Node flavor: 4 vCPUs 8 GiB OS: EulerOS 2.9 System disk: 50 GiB General purpose SSD Data disk: 100 GiB General purpose SSD 	1	USD0.2096/ hour
Elastic Volume Service (EVS)	 Pay-per-use recommended EVS disk specification: 100 GiB EVS disk type: General purpose SSD 	1	USD0.0157/ hour
Load Balancer (ELB)	 Pay-per-use recommended Type: Shared Billed By: Traffic Bandwidth: 5 Mbit/s 	1	USD0.053/hour + USD0.114/GB (The traffic fee is charged based on the actual outbound traffic.)

Table 4-2 Resource	and cost	planning
--------------------	----------	----------

4.1.3 Procedure

4.1.3.1 Installing and Deploying Jenkins Master

NOTE

On the Jenkins page, the UI strings in Chinese and English are different. The screenshots in this section are for your reference only.

Selecting an Image

Select a relatively new, stable image from Docker Hub. For this test, select **jenkinsci/blueocean**, which is bound with all Blue Ocean add-ons and functions. There is no need to install Blue Ocean add-ons separately. For details, see **Installing Jenkins**.

Preparations

• Before creating a containerized workload, buy a cluster (the cluster must contain at least one node with four vCPUs and 8 GiB memory). For details, see **Buying a CCE Cluster**.

The Docker in Docker scenario is required, which is, running the Docker commands in the container. Select the Docker container engine for the node.

 To enable external networks to access the workload, ensure that an elastic IP address (EIP) has been bound to or a load balancer has been configured for at least one node in the cluster.

Installing and Deploying Jenkins on CCE

- Step 1 Log in to the CCE console, choose Workloads > Deployments and click Create
 Workload on the upper right corner.
- Step 2 Configure basic workload parameters.
 - Workload Name: jenkins (customizable)
 - **Namespace**: Select the namespace where Jenkins will be deployed. You can create a namespace.
 - Pods: Set it to 1.

Basic Info								
Workload Type	, Deployment	StatefulSet	DaemonSet	Job	Cron Job			
	A Switching workload type r	equires reconfiguring workload para	meters.					
Workload Name	jenkins					Cluster Name	CCE Standard	
Namespace	cicd	~ C	Create Namespace			Description	A maximum of 200 characters are supported.	
Pods	- 1 +						0/200	

Step 3 Configure basic container parameters.

- Image Name: Enter jenkinsci/blueocean. Select an image version as required. If no version is selected, the latest version will be used by default.
- CPU Quota: Set Limit to 2 cores.
- Memory Quota: Set Limit to 2048 MiB.
- **Privileged Container**: If Jenkins with a single Master is used, the privileged container must be enabled so that the container can perform operations on the node. Otherwise, you cannot run the Docker commands in the Jenkins Master container.

Retain the default values for other parameters.

Figure 4-2 Basic container parameters

container settings				
Container Information	Container - 1			+ Add Container
	Basic Info	Container container-1	Pull Policy	🗌 Alwaya 💿
	Health Check Environment Variables Data	Image Name (Innthrocoblaeocean Replace Image) CPU Quela Request 0.25 core, Unit (2.00 core ©	Image Tag Memory Quota	Select V Request 512.00 MB; Lime 2.048.00 MB ()
	Storage Security Context Logging	GPU Quela The function is unavailable because add on GPU is not installed, install now Prolegie Container	NPU Quota Init Container	This function is unavailable because add on NPU is not installed, install now

Step 4 Click Data Storage and add a persistent storage volume.

Click **Add Volume**, select **PVC** from the drop-down list, and add a storage volume. Enter **/var/jenkins_home** under **Mount Path** to mount the storage volume to **/var/jenkins_home** of the Jenkins container for persistent data storage.

NOTE

The cloud storage type can be **EVS** or **SFS**. If no cloud storage is available, click **Create PVC**. If you select **EVS**, the AZ of the EVS disk must be the same as that of the node.

Figure 4-3 Adding a cloud volume

Container Information	Outline 4						
	Container - 1						
	Basic Info						
		Ū					
	refresh.						
	Environment	Mount Path ③	Subpath 💿	Permission Operation			
	Variables	/var/jenkins_home	Enter a sub-path, for example, tmp.	Read-write V Delete			
	Data Storage		+				
	Security Context Add Volume Learn more about storage volumes.						
	Logging						

- **Step 5** Add permissions to the Jenkins container so that related commands can be executed in it.
 - 1. Ensure that **Privileged Container** is enabled in **3**.
 - 2. Click **Data Storage**, click **Add Volume**, select **hostPath**, and mount the host path to the corresponding container path.

Table 4-3 Mounting path

Storage Type	Host Path	Mounting Path
hostPath	/var/run/docker.sock	/var/run/docker.sock
hostPath	/usr/bin/docker	/usr/bin/docker
hostPath	/usr/lib64/libltdl.so.7	/usr/lib/x86_64-linux-gnu/ libltdl.so.7
hostPath	/usr/bin/kubectl	/usr/local/bin/kubectl

After the mounting is complete, the page shown in **Figure 4-4** is displayed.

Container Settings								
Container Information	Container - 1	+ Add Container						
	Basic Info Lifecycle Health Check	PVC projentins EVS > Q.	You can Create PVC and then click refresh.	Ū				
	Environment Variables	Mount Path ③	Subpath () Entire a sub-path, for example, trip,	Permission Operation				
	Data Storage		+					
	Security Context Logging	HostPath Instrumenteder sock	x Subpath (0) Etter a sub-path, for example, trap. +	Ĝ Perminsten Operation Read-write ♥ Duble				
		HostPath HostPath Asrbinidocker	x	Û				
		Mount Path ③	Subpath ③	Permission Operation				
		Ausribin/docker	Enter a sub-path, for example, http://www.http://wwwww.http://www.http://www.http://www.http://www.	Ressivence V Delete				

Figure 4-4 Mounting the host paths to the corresponding container paths

3. In Security Context, set User ID to 0 (user root).

Figure 4-5 Configuring the user

ontainer Settings								
ontainer Information	ution Container - 1							
	Basic Info Lifecycle Health Check Environment Variables Data Storage Security Context	O You o	en secolor the case white note the processes is a contensor.					
	Logging							

Step 6 Specify the access mode in Service Configuration.

The Jenkins container image has two ports: 8080 and 50000. Configure them separately. Port 8080 is used for web login, and port 50000 is used for the connection between Master and Agent.

In this example, two Services are created:

• LoadBalancer: provides external web access using port 8080. You can also select NodePort to provide external access.

Set the Service name to **jenkins** (customizable), the container port to **8080**, the access port to **8080**, and retain the default values for other parameters.

• **ClusterIP**: used by the Agent to connect to the Master. The IP addresses of **jenkins-web** and **jenkins-agent** need to be the same. Therefore, port 8080 for web access and port 50000 for agent access are included.

Set the Service name to **agent** (customizable), the container port 1 to **8080**, the access port 1 to **8080**, the container port 2 to **50000**, the access port 2 to **50000**, and retain the default values for other parameters.

NOTE

In this example, Agents and the Master are deployed in the same cluster. Therefore, the Agents can use the ClusterIP Service to connect to the Master.

If Agents need to connect to the Master across clusters or through the public network, select a proper Service type. Note that the IP addresses of **Jenkins-web** and **Jenkins-agent** need to be the same. Therefore, **ports 8080 and 50000 must be enabled for the IP address connected to jenkins-agent**. For addresses used only for web access, enable only the port 8080.

Figure 4-6 Adding a Service

Service Settings			
Service	Access Mode	Access Port:Container Port/Protocol	Operation
jenkins	LoadBalancer	8080 -> 8080 / TCP	Delete Edit
agent	ClusterIP	8080 -> 8080 / TCP 50000 -> 50000 / TCP	Delete Edit
	+		

- **Step 7** Retain the default settings for **Advanced Settings** and click **Create Workload**.
- **Step 8** Click **Back to Deployment List** to view the Deployment status. If the workload is in the **Running** status, the Jenkins application is accessible.

----End

Logging In and Initializing Jenkins

Step 1 On the CCE console, click the name of the target cluster to access the cluster console. Choose Services & Ingresses in the navigation pane. On the Services tab, view the Jenkins access mode.

Figure 4-7 Access mode corresponding to port 8080

rvices Ingresses						Best Practices Quick Links	Create from YAML Create Service
Delete Export ~	Quota (remaining/total): Load Balancer (197/	200) Listener (294 /300) Ba	ackend ECS (994/1,000)				
Q. Select a property or enter a keywor	rd.						0
Service 🕀	Selector 🕀	Namespace 🕀	Service Type 😂	Access Address	Access Port:Container Port/Prote	ccol 🕀 Created 🕀	Operation
O agent	app jenkins version v1	cicd	ClusterIP	10.247.180.25 (Cluster IP)	8080 -> 8080 / TCP 50000 -> 50000 / TCP	51 seconds ago	Manage Pod - View Events - More - Y
o jenkins	app jenkins version v1	cicd	LoadBalancer	10.247.195.227 (Cluster IP) 192.168.0.171 (Load Balancer	8080 -> 8080 / TCP	51 seconds ago	Manage Pod View Events More ~
Total Records: 2 10 V < 1	>						

Step 2 Enter **EIP:8080** of the load balancer in the browser address box to visit the Jenkins configuration page.

When you visit the page for the first time, you are prompted to obtain the initial administrator password. You can obtain the password from the Jenkins pod. Before running the following commands, connect to the cluster using kubectl. For details, see **Connecting to a Cluster Using kubectl**.

kubectl get pod -n cicd NAME READY STATUS RESTARTS AGE jenkins-7c69b6947c-5gvlm 1/1 Running 0 17m # kubectl exec -it jenkins-7c69b6947c-5gvlm -n cicd -- /bin/sh # cat /var/jenkins_home/secrets/initialAdminPassword b10eabe29a9f427c9b54c01a9c3383ae **Step 3** The system prompts you to select the default recommended add-on and create an administrator upon the first login. After the initial configuration is complete, the Jenkins page is displayed.

lenkins	
board >	
+ New Item	
22 People	
 Build History Manage Jenkins 	
20 My Views	
Blue Ocean	
Build Queue	~
a builds in the queue.	
Suild Executor Status	~
Idle Idle	

----End

Modifying the Number of Concurrent Build Jobs

Step 1 On the Jenkins dashboard page, click Manage Jenkins on the left, choose System Configuration > Manage nodes and clouds, and select Configure from the dropdown list of the target node.

Dashboard > Nodes >								
↑ Back to Dashboard ③ Manage Jenkins		Manage nodes and clouds						Refresh status
+ New Node		S Name i	Architecture	Clock Difference	Free Disk Space	Free Swap Space	Free Temp Space	Response Time
Configure Clouds		Built-In Node	Linux (amd64)	In sync	9.32 GB	© 0 B	80.07 GB	0ms 🔞
Node Monitoring		Configure	23 min	23 min	23 min	23 min	23 min	23 min
Build Queue	~	🖍 Load Statistics						
No builds in the queue.		Script Console						
Build Executor Status	~							

NOTE

- You can modify the number of concurrent build jobs on both Master and Agent. The following uses Master as an example.
- If the Master is used with Agents, you are advised to set the number of concurrent build jobs of Master to **0**. That is, all build jobs are performed using Agents. If a single Master is used, you do not need to change the value to **0**.
- **Step 2** Modify the maximum number of concurrent build jobs. In this example, the value is changed to **2**. You can change the value as required.

Number of executors ?	
2	
The maximum number of concurrent builds that Jenkins may perform on this node. A good value to start with would be the number of CPU cores on the machine. Setting a higher value would cause each build to take longer, but could increase the overall throughput. For example, one build might be CPU-bound, while a second build running at the same time might be I/O-bound — so the second build could take advantage of the spare I/O capacity at that moment.	9
Agents (nodes that are not the built-in node) must have at least one executor. To temporarily prevent any builds from being executed on an agent, use the Mark this node temporarily offline button on the agent's page.	
For the built-in node, set the number of executors to zero to prevent it from executing builds locally on the controller. Note: The built-in node will always be able to run flyweight tasks including Pipeline's top-level task.	

```
----End
```

4.1.3.2 Configuring Jenkins Agent

After Jenkins is installed, the following information may display, indicating that Jenkins uses a Master for local build and Agents are not configured.

Q Search	?	1 2	2	💄 adn	nin ~	→ log out
Building on the built-in node can be a security issue. You should set up distributed builds. See <u>the documentation</u> .	Se	et up agent	Set up o	loud	Dismiss	

If you install Jenkins using a Master, you can build a pipeline after performing operations in **Installing and Deploying Jenkins Master**. For details, see **Using Jenkins to Build a Pipeline**.

If you install Jenkins using a Master and Agents, you can select either of the following solutions to configure Agents.

- **Fixed Agent**: The Agent container keeps running and occupying cluster resources after a job is built. This configuration is simple.
- **Dynamic Agent**: An Agent container is dynamically created during job build and is killed after the job is built. In this way, resources can be dynamically allocated and the resource utilization is high. This configuration is complex.

In this section, the Agent is containerized using the **jenkins/inbound-agent:4.13.3-1** image.

Adding a Fixed Agent to Jenkins

- Step 1 Log in to the Jenkins dashboard, click Manage Jenkins on the left, and choose System Configuration > Manage nodes and clouds.
- **Step 2** Click **New Node** on the left, enter the node name **fixed-agent** (which can be customized), and select **Permanent Agent** for **Type**.

Dashboard \rightarrow Nodes \rightarrow		
Back to Dashboard Back to Dashboard Manage Jenkins H New Node Configure Clouds Node Monitoring		New node Node name fixed-agent
Build Queue No builds in the queue.	~	Adds a plain, permanent agent to Jenkins. This is called "permanent" because Jenkins doesn't provide higher level of integration with these agents, such as dynamic provisioning. Select this type if no other agent types apply — for example such as when you are adding a physical computer, virtual machines managed outside Jenkins, etc.
Build Executor Status	~	Create
2 Idle		

Step 3 Specify the following node information:

- **Number of executors**: The default value is **1**. Set this parameter as required.
- Remote root directory: Enter /home/jenkins/agent.
- Launch method: Select Launch agent by connecting it to the controller.

Retain the values for other parameters and click **Save**.

Number of executors ?	
1	
Remote root directory ?	
/home/jenkins/agent	
Labels ?	
Usage ?	
Use this node as much as possible	~
launch method	
Launch agent by connecting it to the controller	~

Step 4 In the **Nodes** page, click the new node. The Agent status is disconnected, and the method for connecting the node to Jenkins is provided. This command applies to VM installation. In this example, container-based installation is used. Therefore, you only need to copy the secret, as shown in the following figure.

Dashboard > Nodes > fixed-agent	
↑ Back to List	Agent fixed-agent
Q Status	Connart sharet to lankins one of these uses:
Delete Agent	Run from agent command line:
② Configure	java -jar agent.jar -jnlpUrl http:// 💼 📕 🗰 🗰 18080/computer/fixed%2Dagent/jenkins-agent.jnlp -secret
Build History	4dc38891954b66dee8a47b1826460e7ca9f8438a88794249c2861e1132e59c475 •workDir "/home/jenkins/agent"
✓ Load Statistics	Run from agent command line, with the secret stored in a file:
🗍 Log	echo 4dc38891954b5dee8a47b1826466e7ca9f8438a88794249c2861e1132e59c475 > secret-file
Blue Ocean	java -jar agent.jar -jnlpUrl http:// 🛄 📲 📲 8000/computer/fixed%2Dagent/jenkins-agent.jnlp -secret @secret-file -workDir "/home/jenkins/agent"
Build Executor Status V	Projects tied to fixed-agent

Step 5 Log in to the CCE console, click the target cluster. Choose Workloads > Deployments and click Create Workload on the right.

Step 6 Configure basic workload parameters.

- Workload Name: agent (user-defined)
- **Namespace**: Select the namespace where Jenkins will be deployed. You can create a namespace.
- Pods: Set it to 1.

Workload Type	. Deployment	StatefulSet	DaemonSet	Job	Cron Job	
	A Switching workload type r	equires reconfiguring workload paran	neters.			
Workload Name	agent				Cluste	er Name CCE Standard
Namespace	cicd	~ Q	Create Namespace		Descr	iption A maximum of 200 characters are supported.
Pods	- 1 +					0/200

Step 7 Configure basic container parameters.

- **Image Name**: Enter **jenkins/inbound-agent:4.13.3-1**. The image version may change with time. Select an image version as required or use the latest version.
- CPU Quota: In this example, set Limit to 2 cores.
- Memory Quota: Set Limit to 2048 MiB.
- **Privileged Container**: must be enabled so that the container can obtain permissions on the host. Otherwise, Docker commands cannot be executed in the container.

Retain the default values for other parameters.

Container Settings				
Container Information	Container - 1			+ Add Container
	Basic Info	Container Container 1	Pull Policy	🗌 Ahrays 💿
	Health Check Environment	Image Name jenkins/inbound-agent-4.13.3-1 Replace Image	Image Tag	Seled
	Variables Data Storage	CPU Guota Request 0.25 cores; Limit 2.00 cores ①	Memory Quota	Request 612.00 MiB; Limit 2.048.00 MiB ③
	Security	GPU Quota This function is unavailable because add-on GPU is not installed. Install now	NPU Quota	This function is unavailable because add-on NPU is not installed. Install now
	Logging	Privileged ③ Container	Init Container	•

Step 8 Run the following commands to configure the environment variables:

- JENKINS_URL: Jenkins access path, which should be the address of port 8080 configured in Step 6. (Both ports 8080 and 50000 must be enabled.) In this example, the cluster IP address and port 8080 and cluster IP address and port 50000 are used, for example, http://10.247.222.254:8080.
- JENKINS_AGENT_NAME: name of the Agent set in Step 2. In this example, the value is fixed-agent.
- JENKINS_SECRET: secret copied from Step 4.
- JENKINS_AGENT_WORKDIR: remote work directory configured in Step 3, that is, /home/jenkins/agent.

Container Settings										
Container Information	Container - 1	*1 *								
	Basic Info Lifecycle Health	Environment vantables aflect the way a running container will behave. You can modily created vantables as required after deploying the workload. How Do 1Add Environment Vantables? Adding a Vantable Editing Custom Vantables in Batches								
	Check Environment Variables	Туре		Variable Name	Va	riable Value/Reference		Operation		
		Custom	v	JENKINS_URL				Delete		
	Data Storage	Custom	×)	JENKINS_AGENT_NAME		fixed-agent		Delete		
	Security Context	Custom	×)	JENKINS_SECRET		100000000		Delete		
	erelike di	Custom	~	JENKINS_AGENT_WORKDIR		'horne/jenkins/agent]	Delete		

- **Step 9** Add permissions to the Jenkins container so that Docker commands can be executed in the Jenkins container.
 - 1. Ensure that **Privileged Container** is enabled in **3**.
 - 2. Click **Data Storage**, click **Add Volume**, select **hostPath**, and mount the host path to the corresponding container path.

 Table 4-4 Mounting path

Storage Type	Host Path	Mounting Path
hostPath	/var/run/docker.sock	/var/run/docker.sock
hostPath	/usr/bin/docker	/usr/bin/docker
hostPath	/usr/lib64/libltdl.so.7	/usr/lib/x86_64-linux-gnu/ libltdl.so.7
hostPath	/usr/bin/kubectl	/usr/local/bin/kubectl

After the mounting is complete, the page shown in **Figure 4-8** is displayed.

Figure 4-8 Mounting the host paths to the corresponding container paths

ontainer Settings				
nane momaco	Container - 1			+ Add Container
	Basic Info Lifecycle Health	PVC PVC pvc.jenkins EVS v	Q: You can Create FMC and then click refresh.	Ū
	Environment Variables	Mount Path	Subpath 🕥 Enter a sub-path, for example, tmp.	Permission Operation Read-write Delete
	Security Context Logging	HostPath HostPath Varituridocker.sock	+	Ŭ
		Mount Path ③ Avanhunddoker sock	Subpath () Errier a sub-path, for example, tmp.	Permission Operation Read-write Celete
			+	
		HostPath HostPath Assrbin/docker	x	Ü
		Mount Path ③	Subpath ①	Permission Operation
		Ausribin/docker	Enter a sub-path, for example, trap.	Read-write V Delete

3. In Security Context, set User ID to 0 (user root).

Figure 4-9 Configuring the user

Container Settings				
Container Information	Container - 1		+	Add Container
	Basic Info Lifecycle Health Check Environment Variables Data Storage Security Context	User ID	an specify the user when notes the processors is a contense.	
	r.000=0			

Step 10 Retain the default settings for **Advanced Settings** and click **Create Workload**.

Step 11 Go to the Jenkins page and refresh the node status to **In sync**.

$Dashboard \rightarrow Nodes \rightarrow$								
↑ Back to Dashboard ③ Manage Jenkins		Manage nodes and c	louds					Refresh status
+ New Node		S Name I	Architecture	Clock Difference	Free Disk Space	Free Swap Space	Free Temp Space	Response Time
Configure Clouds		Built-In Node	Linux (amd64)	In sync	9.39 GB	🖨 0 В	80.84 GB	0ms 💿
Node Monitoring		fixed-agent	Linux (amd64)	In sync	80.84 GB	🖨 0 В	80.84 GB	79ms 🔞
Build Queue	~	Data obtained	4 min 50 sec	4 min 50 sec	4 min 50 sec	4 min 50 sec	4 min 50 sec	4 min 50 sec

NOTE

After the Agent is configured, you are advised to set the number of concurrent build jobs of the Master to **0**. That is, you use the Agent for build. For details, see **Modifying the Number of Concurrent Build Jobs**.

----End

Setting a Dynamic Agent for Jenkins

Step 1 Install the plug-in.

On the Jenkins dashboard page, click **Manage Jenkins** on the left and choose **System Configuration** > **Manage Plugins**. On the **Available** tab, search for **Kubernetes** and install **Kubernetes CLI** and **Kubernetes**.

Dashboard > Plugin Manager		
↑ Back to Dashboard ③ Manage Jenkins	Plugin Manager Updates Available Installed Advanced	
	Install Name Kubernetes 3734.v562b; b, a, 627ea, c Could Providers Cutater Management Indemnetes Agent Management This plugin integrates Jenkins with Kubernetes	Released : 1 mo 11 days ago
	Kubernetes CLI 1.10.3 Kubernetes Configure kubecil for Kubernetes	1 yr 5 mo ago
	Kubernetes Contentials kubernetes Provides a read only credentials store backed by Kubernetes.	1 mo 13 days ago
	Kubernetes :: Pipeline :: DevOps Steps 1.6	3 yr 10 mo ago

The plug-in version may change with time. Select a plug-in version as required.

• Kubernetes Plugin: 3734.v562b_b_a_627ea_c

It is used to run dynamic Agents in the Kubernetes cluster, create a Kubernetes pod for each started Agent, and stop the pod after each build is complete.

• Kubernetes CLI Plugin: 1.10.3

kubectl can be configured for jobs to interact with Kubernetes clusters.

NOTE

The Jenkins plugins are provided by the plugin maintainer and may be iterated due to security risks.

Step 2 Add cluster access credentials to Jenkins.

Add cluster access credentials to Jenkins in advance. For details, see **Setting Cluster Access Credentials**.

Step 3 Configure basic cluster information.

On the Jenkins dashboard page, click **Manage Jenkins** on the left and choose **System Configuration** > **Manage nodes and clouds**. Click **Configure Clouds** on the left to configure the cluster. Click **Add a new cloud** and select **Kubernetes**. The cluster name can be customized.

Step 4 Enter Kubernetes Cloud details.

Set the following cluster parameters and retain the values for other parameters, as shown in **Figure 4-10**.

- Kubernetes URL: cluster API server address. You can enter https:// kubernetes.default.svc.cluster.local:443.
- **Credentials**: Select the cluster credential added in **Step 2**. You can click **Test Connection** to check whether the cluster is connected.
- Jenkins URL: Jenkins access path. Enter the IP address of port 8080 set in Step 6 (ports 8080 and 50000 must be enabled for the IP address, that is, the intra-cluster access address), for example, http://10.247.222.254:8080.

Figure	4-10	Examp	le
--------	------	-------	----

Kubernetes URL ?	
https://kubernetes.default.svc.cluster.local/443	
Use Jenkins Proxy ?	
Kubernetes server certificate key ?	
Disable https certificate check ?	
Kubernetes Namespace	
JNLP Docker Registry ?	
Credentials	
k8s-token	~
+ Add	
Connected to Kubernetes v1.21.4-r0-CCE22.5.1	Test Connection
WebSocket ?	
Direct Connection ?	
Jenkins URL ?	
http://10.247.222.254:8080	

Step 5 Pod Template: Click **Add Pod Template > Pod Template details** and set pod template parameters.

- Set the basic parameters of the pod template, as shown in Figure 4-11.
 - Name: jenkins-agent
 - Namespace: cicd
 - Labels: jenkins-agent
 - Usage: Select Use this node as much as possible.

Figure 4-11 Basic parameters of the pod template

Pod Template	×
Name ?	
jenkins-agent	
Namespace ?	
cicd	
Labels ?	
jenkins-agent	
Usage ?	
Use this node as much as possible	~
Pod template to inherit from ?	
Containers ?	
List of container in the agent pod	
Add Container *	

- Add a container. Click **Add Container > Container Template**. **Figure 4-12** shows the parameters.
 - Name: The value must be jnlp.
 - Docker image: jenkins/inbound-agent:4.13.3-1. The image version may change with time. Select an image version as required or use the latest version.
 - Working directory: /home/jenkins/agent is selected by default.
 - Command to run/Arguments to pass to the command: Delete the existing default value and leave these two parameters empty.
 - Allocate pseudo-TTY: Select this parameter.
 - Select Run in privileged mode and set Run As User ID to 0 (root user).

Figure 4-12 Container template parameters

Containers ?

List of container in the agent pod	
= Container Template	×
Name 🕄	
inp	
Docker image 🕐	
jenkins/inbound-agent:4.13.3-1	
Always pull image 🕐	
Working directory ?	
/home/jenkins/agent	
Command to run 🕐	
Arguments to pass to the command 👔	
Allocate pseudo-TTY (?)	
Environment Variables 🔞	
List of environment variables to set in agent pod	
Add Environment Variable *	
Run in privileged mode 👔	
Run As User ID ?	

• Add a volume: Choose **Add Volume > Host Path Volume** to mount the host path in **Table 4-5** to the corresponding path of the container.

Table 4-5 Mounting path

Storage Type	Host Path	Mounting Path
hostPath	/var/run/docker.sock	/var/run/docker.sock
hostPath	/usr/bin/docker	/usr/bin/docker
hostPath	/usr/lib64/libltdl.so.7	/usr/lib/x86_64-linux-gnu/ libltdl.so.7
hostPath	/usr/bin/kubectl	/usr/local/bin/kubectl

After the mounting is complete, the page shown in Figure 4-13 is displayed.

Figure 4-13	Mounting the	host paths to	the corresponding	container paths
-------------	--------------	---------------	-------------------	-----------------

olumes ?	
st of volumes to mount in agent pod	
≡ Host Path Volume	(
Host path ?	
/var/run/docker.sock	
Mount path ?	
/var/run/docker.sock	
≡ Host Path Volume	
Host path ?	
/usr/bin/docker	
Mount path ?	
/usr/bin/docker	
≡ Host Path Volume	
Host path ?	
/usr/lib64/libItdl.so.7	
Mount path ?	
/usr/lib/x86_64-linux-gnu/libltdl.so.7	

- Run As User ID: 0 (root user)
- Workspace Volume: working directory of the agent. Persistence is recommended. Select Host Path Workspace Volume and set Host path to / home/jenkins/agent.

Vorkspace Volume ? ?		
Host Path Workspace Volume		
Host path ?		
/home/jenkins/agent		

Step 6 Click Save.

NOTE

After the Agent is configured, you are advised to set the number of concurrent build jobs of the Master to **0**. That is, you use the Agent for build. For details, see **Modifying the Number of Concurrent Build Jobs**.

----End

Setting Cluster Access Credentials

The certificate file that can be identified in Jenkins is in PKCS#12 format. Therefore, convert the cluster certificate to a PFX certificate file in PKCS#12 format.

Step 1 Log in to the CCE console and click the cluster name to access the cluster console. Choose Overview in the navigation pane. On the page displayed, locate the Connection Information area, and download the cluster certificate. The certificate contains ca.crt, client.crt, and client.key files.

Connection Information		
Private IP	https://192.168.254.194:5443	
EIP	Bind	
Custom SAN	- 2	
kubectl	Configure	
Certificate Authentication	X.509 certificate Download Revoke	
continuato / talicinication		

- **Step 2** Log in to a Linux host, place the three certificate files in the same directory, and use OpenSSL to convert the certificate into a **cert.pfx** certificate. After the certificate is generated, the system prompts you to enter a custom password. openssl pkcs12 -export -out cert.pfx -inkey client.key -in client.crt -certfile ca.crt
- **Step 3** On the Jenkins console, choose **Manage Jenkins** > **Manage Credentials** and click **Global**. You can also create a domain.

Dashboard > Credentials					
+ New Item		Credentials			
Build History		T P Store I	Domain	ID	Name
 Manage Jenkins My Views 		Stores scoped to Jenkins			
Open Blue Ocean		P Store 1	Domains		
New View		Jenkins	(global)		
Build Queue	~	lcon: S M L			
No builds in the queue.					

Step 4 Click Add Credential.

- Kind: Select Certificate.
- Scope: Select Global.
- **Certificate**: Select **Upload PKCS#12 certificate** and upload the **cert.pfx** file generated in **Step 2**.
- **Password**: The password customized during **cert.pfx** conversion.
- **ID**: Set this parameter to **k8s-test-cert**, which can be customized.

New credentials

ſ

ind	
Certificate	~
Scope ?	
Global (Jenkins, nodes, items, all child items, etc)	~
Certificate ? Upload PKCS#12 certificate CN=90dd374846814d50ba02772b08c0bfd7, O=system.masters + O=6becd28f7bc0489297999e349b869ff5 + O=b3fdc3bff29f49d1abc8341005e1d236 Choose File cert.pfx	
Password ?	
ID ? k8s-test-cert	

----End

4.1.3.3 Using Jenkins to Build a Pipeline

Obtaining a Long-Term Valid Login Command

During Jenkins installation and deployment, the Docker commands have been configured in the container (see 9). Therefore, no additional configuration is required for interconnecting Jenkins with SWR. You can directly run the Docker commands. You only need to obtain a long-term valid SWR login command. For details, see **Obtaining a Long-Term Valid Login Command**.

For example, the command of this account is as follows:

docker login -u ap-southeast-1@xxxxx -p xxxxx swr.ap-southeast-1.myhuaweicloud.com

Creating a Pipeline to Build and Push Images

In this example, Jenkins is used to build a pipeline to pull code from the code repository, package the code into an image, and push the image to SWR.

The pipeline creation procedure is as follows:

- **Step 1** Click **New Item** on the Jenkins page.
- Step 2 Enter a task name and select Create Pipeline.

Enter	an item name
test-pi » Require	pe ed field
1	Freestyle project This is the central feature of Jenkins. Jenkins will build your project, combining any SCM with any build system, and this can be even used for something other than software build.
	Pipeline Orchestrates long-running activities that can span multiple build agents. Suitable for building pipelines (formerly known as workflows) and/or organizing complex activities that do not easily fit in free-style job type.
	Multi-configuration project Suitable for projects that need a large number of different configurations, such as testing on multiple environments, platform-specific builds, etc.
	Folder Creates a container that stores nested items in it. Useful for grouping things together. Unlike view, which is just a filter, a folder creates a separate namespace, so you can have multiple things of the same name as long as they are in different folders.
T	Multibranch Pipeline Creates a set of Pipeline projects according to detected branches in one SCM repository.

Step 3 Configure only the pipeline script.

General	Build Triggers	Advanced Project Options	Pipeline		
Pipelin	e				
Definitio	n				
Dinalina					
Fipeline	e script				
Scri	pt ?				
	1 def git_url = 2 def swr_logir 3 def swr_regic 4 def organizat 5 def build_nam 6 def credentiz 7 def apiserver 8 9 * pipeline {	<pre>'https://github.com/lookforst = 'docker login -u on = ' ion = 'container' e = 'jenkins.demo' al = 'k8s-token' = 'k8s-token'</pre>	ar/jenkins-	demo.git' try sample Pipeline	~
	11 ▼ stages { 12 ▼ stage 13 ▼ s 14 15 16 ▼ 17	e('Clone') { steps{ echo "1.Clone Stage" git url: git_url script { build_tag = sh(returnSi	:dout: true,	, script: 'git rev-parseshort HEAD').trim()	
Pipe	Use Groovy San	dbox (?)			ž
Sa	ive A	hpply			

The following pipeline scripts are for reference only. You can customize the script. For details about the syntax, see **Pipeline**.

Some parameters in the example need to be modified:

- **git_url**: Address of your code repository. Replace it with the actual address.
- swr_login: The login command obtained in Obtaining a Long-Term Valid Login Command.

- **swr_region**: SWR region.
- organization: The actual organization name in SWR.
- **build_name**: Name of the created image.
- credential: The cluster credential added to Jenkins. Enter the credential ID. If you want to deploy the service in another cluster, add the access credential of the cluster to Jenkins again. For details, see Setting Cluster Access Credentials.
- **apiserver**: IP address of the API server where the application cluster is deployed. Ensure that the IP address can be accessed from the Jenkins cluster.

```
// Define the code repository address.
def git_url = 'https://github.com/lookforstar/jenkins-demo.git'
// Define the SWR login command.
// Define the SWR region.
def swr_region = 'ap-southeast-1'
// Specify the name of an SWR organization to which images are pushed.
def organization = 'container'
// Define the image name.
def build_name = 'jenkins-demo'
// Certificate ID of the cluster to be deployed
def credential = 'k8s-token'
// API server address of the cluster. Ensure that the address can be accessed from the Jenkins cluster.
def apiserver = 'https://192.168.0.100:6443'
pipeline {
  agent any
  stages {
     stage('Clone') {
        steps{
          echo "1.Clone Stage"
          git url: git_url
          script {
             build_tag = sh(returnStdout: true, script: 'git rev-parse --short HEAD').trim()
          }
       }
     }
     stage('Test') {
        steps{
          echo "2.Test Stage"
        }
     }
     stage('Build') {
        steps{
          echo "3.Build Docker Image Stage"
          sh "docker build -t swr.${swr_region}.myhuaweicloud.com/${organization}/${build_name}:$
{build_tag} ."
          //{{build_tag} indicates that the build_tag variable is obtained as the image tag. It is the return
value of the git rev-parse --short HEAD command, that is, commit ID.
        }
     }
     stage('Push') {
        steps{
          echo "4.Push Docker Image Stage"
          sh swr login
          sh "docker push swr.${swr_region}.myhuaweicloud.com/${organization}/${build_name}:$
{build_tag}"
       }
     }
     stage('Deploy') {
        steps{
          echo "5. Deploy Stage"
          echo "This is a deploy step to test"
          script {
          sh "cat k8s.yaml"
          echo "begin to config kubenetes"
```





----End

4.1.3.4 Interconnecting Jenkins with RBAC of Kubernetes Clusters (Example)

Prerequisites

RBAC must be enabled for the cluster.

Scenario 1: Namespace-based Permissions Control

Create a service account and a Role, define a RoleBinding, and bind the service account to the Role.

```
# kubectl create ns dev
# kubectl -n dev create sa dev
# cat <<EOF > dev-user-role.yml
kind: Role
apiVersion: rbac.authorization.k8s.io/v1
metadata:
 namespace: dev
 name: dev-user-pod
rules:
- apiGroups: ["*"]
 resources: ["deployments", "pods", "pods/log"]
verbs: ["get", "watch", "list", "update", "create", "delete"]
EOF
# kubectl create -f dev-user-role.yml
# kubectl create rolebinding dev-view-pod \
   --role=dev-user-pod \
   --serviceaccount=dev:dev \
```

--namespace=dev

Generate the kubeconfig file of a specified service account.

D NOTE

 In clusters earlier than v1.21, tokens are obtained by mounting the secret of a service account to a pod. Such tokens are permanent. However, this approach is not recommended in clusters of v1.21 or later. Starting from v1.25, Kubernetes no longer automatically creates secrets for service accounts as part of its community iteration policy.

Instead, in clusters of v1.21 and later, the recommended approach is to use the **TokenRequest** API to **obtain tokens** and mount them via a projected volume to pods. These tokens remain valid only for a fixed period and become invalid once the pods are deleted. For details, see **Service Account Token Security Improvement**.

 If you need a token that never expires, you can manually manage secrets for service accounts. Although a permanent service account token can be created manually, you are advised to use a short-lived token by calling the TokenRequest API for better security.

Because the cluster version used in this case is v1.25, the ServiceAccount will not have a secret created automatically. This example shows how to manually create a secret named **dev-secret** and associate it with the ServiceAccount named **dev**.

NOTICE

The manually created secret must be in the same namespace as the ServiceAccount to be associated with. Otherwise, the creation may fail.

kubectl apply -f - <<EOF
apiVersion: v1
kind: Secret
metadata:
namespace: dev # Namespace
name: dev-secret
annotations:
kubernetes.io/service-account.name: dev
type: kubernetes.io/service-account-token
EOF</pre>

Check whether **dev-secret** has been created. If **dev-secret** is present in secrets of the **dev** namespace, then it has been created.

# kubectl get s	ecrets -n dev			
NAME	ТҮРЕ	DATA	AG	E
default-secret	kubernetes.io/dockercor	nfigjson	1	2d22h
dev-secret	kubernetes.io/service-acc	ount-token	3	4h14m
paas.elb	cfe/secure-opaque	1	2d2	2h

Generate a kubeconfig file using **dev-secret**.

```
# API_SERVER="https://172.22.132.51:6443"
# CA_CERT=$(kubectl -n dev get secret dev-secret -o yaml | awk '/ca.crt:/{print $2}')
# cat <<EOF > dev.conf
apiVersion: v1
kind: Config
clusters:
    - cluster:
        certificate-authority-data: $CA_CERT
        server: $API_SERVER
        name: cluster
EOF
# TOKEN=$(kubectl -n dev get secret dev-secret -o go-template='{{.data.token}}')
# kubectl config set-credentials dev-user \
        --token=`echo ${TOKEN} | base64 -d` \
```

--kubeconfig=dev.conf

- # kubectl config set-context default \
 - --cluster=cluster \
 - --user=dev-user \ --kubeconfig=dev.conf
- # kubectl config use-context default \
 - --kubeconfig=dev.conf

Verify the configuration.

kubectlkubeconfig=dev.conf get po
Error from server (Forbidden): pods is forbidden: User "system:serviceaccount:dev:dev" cannot list pods in the namespace "default"
kubectl -n devkubeconfig=dev.conf run nginximage nginxport 80restart=Never

```
    # kubectt -n dev --kubeconfig=dev.conf run nginx --image nginx --port 80 --restart=Never
    # kubectl -n dev --kubeconfig=dev.conf get po
    NAME READY STATUS RESTARTS AGE
    nginx 1/1 Running 0 39s
```

Verify whether the permissions meet the expectation in Jenkins.

Step 1 On the Jenkins dashboard page, click Manage Jenkins, choose Security > Manage Credentials, and click the default global credential storage domain of Jenkins. You can also create a domain.

Dashboard > Credentials					
+ New Item		Credentials			
 People Build History 		T P Store I	Domain	ID	Name
Manage Jenkins		Stores scoped to Jenkins			
Open Blue Ocean		P Store⊥	Domains		
New View		Jenkins	(global)		
Build Queue	~	lcon: S M L			

- **Step 2** Click **Add Credentials**. Set **Kind** to **Secret file**, **Scope** to **Global (Jenkins, nodes, items, all child items, etc)**, and **File** to the downloaded kubeconfig configuration file, retain the default values for other parameters, and click **Create**.
- **Step 3** Start the Jenkins job. In this example, Jenkins fails to be deployed in namespace **default** but is successfully deployed in namespace **dev**.

+ cat k8s.yanl
apiVersion: extensions/w1betal
kind: Deployment
netudata:
nane: jenking-domo
namespace: default
spec:
template:
netadata:
labels:
app: jenkins-demo
spect
containers:
- image: cnych/jenkins-demo:716a813
imagePullPolicy: IfNotPresent
name: jenkins-demo
60.4.
- name: branch
value: <branch_name></branch_name>
[Pipeline] echo
begin to config kubenetes
[Pipeline] kubernetezDeploy
Starting Kubernetes deployment
Loading configuration: /var/jenkins_home/workspace/liuyi=swr=cce=pipe01/k8s.yaml
ERNOR: ERNOR: io.fabric8.kubernetes.client.KubernetesClientException: Failure executing: GET at: https://192.168.0.153:5443/apis/extensions/vlbetal/nanespaces/default/deployments/jenking=deng. Message: Forbidden/
User dew-user doesn't have permission. deployments.extensions jenkins-demo is forbidden: User system:serviceaccount:dev:sa-dev cannot get deployments.extensions in the namespace "default".
hudson.reaoting.ProxyException: io.fabric8.kubernetes.client.KubernetesClientException: Failure executing: GET at: https://132.168.0.153:5443/apis/extensions/vibetal/namespaces/default/deployments/ienkins-demo.
- Message: Forbidden! User dev-user doesn't have permission, deployments.extensions jenkins-demo is forbidden: User "system:serviceaccount:dev:sa-dev" cannot get deployments.extensions in the namespace "default".
at io.fabric8.kubernetes.client.dsl.base.OperationSupport.requestFailure(O <u>perationSupport.java:472)</u>
at io.fabric8.kubernetes.client.dsl.base.OperationSupport.assertResponseCode(OperationSupport.java:409)
at io.fabric8.kubernetes.client.dsl.base.OperationSupport.handleResponse(OperationSupport.java:381)
at io.fabric8.kubernetes.client.dsl.base.OperationSupport.handleResponse(OperationSupport.java:344)
at io.fabric8.kubernetes.client.dsl.base.OperationSupport.handleGet(OperationSupport.java:313)
at io.fabric8.kubernetes.client.dsl.base.OperationSupport.handleGet(OperationSupport.java:296)
at io. fabric8. kubernetes. client. dsl. base. BaseOperation. handleGet(BaseOperation. java: 770)
at io.fabric8.kubernetes.client.dsl.base.BaseOperation.getMandatory(BaseOperation.java:195)
at io.fabric8.kubernetes.client.ds1.base.BaseOperation.get(BaseOperation.java:162)



hooray, success

----End

Scenario 2: Resource-based Permissions Control

```
Step 1 Generate the service account, role, and binding.
```

kubectl -n dev create sa sa-test0304

```
# cat <<EOF > test0304-role.yml
kind: Role
apiVersion: rbac.authorization.k8s.io/v1
metadata:
 namespace: dev
 name: role-test0304
rules:
- apiGroups: ["*"]
 resources: ["deployments"]
 resourceNames: ["tomcat03", "tomcat04"]
 verbs: ["get", "update", "patch"]
EOF
# kubectl create -f test0304-role.yml
# kubectl create rolebinding test0304-bind \
```

--namespace=dev **Step 2** Generate a kubeconfig file.

--role=role-test0304 \

--serviceaccount=dev:sa-test0304\

In clusters earlier than v1.21, tokens are obtained by mounting the secret of a service account to a pod. Such tokens are permanent. However, this approach is not recommended in clusters of v1.21 or later. Starting from v1.25, Kubernetes no longer automatically creates secrets for service accounts as part of its community iteration policy.

Instead, in clusters of v1.21 and later, the recommended approach is to use the TokenRequest API to obtain tokens and mount them via a projected volume to pods. These tokens remain valid only for a fixed period and become invalid once the pods are deleted. For details, see Service Account Token Security Improvement.

If you need a token that never expires, you can manually manage secrets for service • accounts. Although a permanent service account token can be created manually, you are advised to use a short-lived token by calling the TokenRequest API for better security.

Because the cluster version used in this case is v1.25, the ServiceAccount will not have a secret created automatically. This example shows how to manually create a secret named **test-secret** and associate it with the ServiceAccount named **sa-test0304**.

kubectl apply -f - <<EOF
apiVersion: v1
kind: Secret
metadata:
namespace: dev
name: test-secret
annotations:
kubernetes.io/service-account.name: sa-test0304
type: kubernetes.io/service-account-token
EOF</pre>

Check whether **test-secret** has been created. If **test-secret** is present in secrets of the **dev** namespace, then it has been created.

# kubectl get s	ecrets -n dev				
NAME	TYPE	DATA	AG	E	
default-secret	kubernetes.io/dockercor	nfigjson	1	2d22h	
dev-secret	kubernetes.io/service-acc	ount-token	3	4h14m	
paas.elb	cfe/secure-opaque	1	2d2	2h	
test-secret	kubernetes.io/service-acc	ount-token	3	25m	

Generate a kubeconfig file using **test-secret**.

```
# API_SERVER=" https://192.168.0.153:5443"
# CA_CERT=$(kubectl -n dev get secret test-secret -o yaml | awk '/ca.crt:/{print $2}')
# cat <<EOF > test0304.conf
apiVersion: v1
kind: Config
clusters:
- cluster:
  certificate-authority-data: $CA_CERT
  server: $API_SERVER
 name: cluster
FOF
# TOKEN=$(kubectl -n dev get secret test-secret -o go-template='{{.data.token}}')
# kubectl config set-credentials test0304-user \
  --token=`echo ${TOKEN} | base64 -d` \
  --kubeconfig=test0304.conf
# kubectl config set-context default \
  --cluster=cluster \
  --user=test0304-user \
  --kubeconfig=test0304.conf
# kubectl config use-context default \
```

--kubeconfig=test0304.conf

Step 3 Verify that Jenkins is running as expected.

In the pipeline script, update the Deployments of tomcat03, tomcat04, and tomcat05 in sequence.

```
try {

kubernetesDeploy(

kubeconfigId: "test0304",

configs: "test03.yaml")

println "hooray, success"

} catch (e) {

println "oh no! Deployment failed! "

println e

}

echo "test04"
```

```
trv {
kubernetesDeploy(
   kubeconfigId: "test0304",
   configs: "test04.yaml")
 println "hooray, success"
} catch (e) {
  println "oh no! Deployment failed! "
  println e
echo "test05"
try {
kubernetesDeploy(
   kubeconfigId: "test0304",
   configs: "test05.yaml")
println "hooray, success"
} catch (e) {
  println "oh no! Deployment failed! "
  println e
```

The following shows examples of the running results.

Figure 4-14 test03

test03 [Pipeline] kubernetesDeploy Starting Kubernetes deployment Loading configuration: /var/jenkins_home/workspace/liuyi-swi Applied Deployment: Deployment(apiVersion=extensions/v1beta) deletionTimestamp=null, finalizers=[], generateName=null, ge selfLink=/apis/extensions/v1beta1/namespaces/dev/deployment: progressDeadlineSeconds=null, replicas=1, revisionHistoryLip strategy=DeploymentStrategy(rollingUpdate=RollingUpdateDeplo additionalProperties={}), additionalProperties={}), type=Rol deletionGracePeriodSeconds=null, deletionTimestamp=null, fir resourceVersion=null, selfLink=null, uid=null, additionalPro [EnvVar(name=branch, value=<BRANCH_NAME>, valueFrom=null, ac livenessProbe=null, name=tomcat03, ports=[], readinessProbe= terminationMessagePath=/dev/termination-log, terminationMess hostNetwork=null, hostPID=null, hostname=null, imagePullSeci securityContext=PodSecurityContext(fsGroup=null, runAsNonRo(terminationGracePeriodSeconds=30, tolerations=[], volumes=[] conditions=[DeploymentCondition(lastTransitionTime=2019-02-: type=Available, additionalProperties={}), DeploymentCondition reason=NewReplicaSetAvailable, status=True, type=Progressing additionalProperties={}) Finished Kubernetes deployment [Pipeline] echo

hooray, success

Figure 4-15 test04



[Pipeline] kubernetesDeploy

Starting Kubernetes deployment Loading configuration: /var/jenkins_home/workspace/liuyi-swr-cce-pipe01/test04.yaml Applied Deployment: Deployment(apiVersion=extensions/v1beta1, kind=Deployment, metadata=Obje deletionTimestamp=null, finalizers=[], generateName=null, generation=3, initializers=null, 1 selfLink=/apis/extensions/v1beta1/namespaces/dev/deployments/tomcat04, uid=06af3b14-3356-11e progressDeadlineSeconds=null, replicas=1, revisionHistoryLimit=null, rollbackTo=null, select strategy=DeploymentStrategy(rollingUpdate=RollingUpdateDeployment(maxSurge=IntOrString(IntVa additionalProperties={}), additionalProperties={}), type=RollingUpdate, additionalProperties deletionGracePeriodSeconds=null, deletionTimestamp=null, finalizers=[], generateName=null, g resourceVersion=null, selfLink=null, uid=null, additionalProperties={}), spec=PodSpec(active [EnvVar(name=branch, value=<BRANCH_NAME>, valueFrom=null, additionalProperties={})], envFrom livenessProbe=null, name=tomcat04, ports=[], readinessProbe=null, resources=ResourceRequirem

terminationMessagePath=/dev/termination-log, terminationMessagePolicy=File, tty=null, volume hostNetwork=null, hostPID=null, hostname=null, imagePullSecrets=[], initContainers=[], nodeN securityContext=PodSecurityContext(fsGroup=null, runAsNonRoot=null, runAsUser=null, seLinux0 terminationGracePeriodSeconds=30, tolerations=[], volumes=[], additionalProperties={}), addi conditions=[DeploymentCondition(lastTransitionTime=2019-02-18T08:56:55Z, lastUpdateTime=2019 type=Available, additionalProperties={}), DeploymentCondition(lastTransitionTime=2019-02-18T reason=ReplicaSetUpdated, status=True, type=Progressing, additionalProperties={})], observed additionalProperties={})

Finished Kubernetes deployment

[Pipeline] echo hooray, success

[Pipeline] echo

test05

[Pipeline] kubernetesDeploy

Starting Kubernetes deployment

Loading configuration: /var/jenkins_home/workspace/liuyi-swr-cce-pipe01/test05.yaml ERROR: ERROR: io.fabric8.kubernetes.client.KubernetesClientException: Failure executing: GET test0304-user doesn't have permission. deployments.extensions "tomcat05" is forbidden: User hudson.remoting.ProxyException: io.fabric 8. kubernetes.client.KubernetesClientException: Fail Forbidden! User test0304-user doesn't have permission. deployments.extensions "tomcat05" is at io.fabric8.kubernetes.client.dsl.base.OperationSupport.requestFailure(OperationSu at io.fabric8.kubernetes.client.dsl.base.OperationSupport.assertResponseCode(Operati at io.fabric8.kubernetes.client.dsl.base.OperationSupport.handleResponse(OperationSu at io.fabric8.kubernetes.client.dsl.base.OperationSupport.handleResponse(OperationSu

----End

4.2 Interconnecting GitLab with SWR and CCE for CI/CD

Background

GitLab is an open source version management system developed with Ruby on Rails for Git project repository management. It supports web-based access to public and private projects. Similar to GitHub, GitLab allows you to browse source code, manage bugs and comments, and control team member access to repositories. You will find it very easy to view committed versions and file history database. Team members can communicate with each other using the built-in chat program (Wall).

GitLab provides powerful CI/CD functions and is widely used in software development.

Figure 4-16 GitLab CI/CD process



This section describes how to interconnect GitLab with SWR and CCE for CI/CD.

Preparations

- 1. Create a CCE cluster and a node and bind an EIP to the node for downloading an image during GitLab Runner installation.
- 2. Download and configure kubectl to connect to the cluster.

Log in to the CCE console and click the cluster name to access the cluster console. In the navigation pane, choose **Overview**, locate the **Connection Information** area, click **Configure** next to **kubectl**, and configure kubectl as instructed.

Connection Information	on
Private IP	https://192.168.254.194:5443
EIP	Bind
Custom SAN	- 2
kubectl	Configure
Certificate Authentication	X.509 certificate Download Revoke

3. Install Helm 3.

Installing GitLab Runner

Log in to **GitLab**, choose **Settings** > **CI/CD** in the project view, click **Expand** next to **Runners**, and search for the GitLab Runner registration URL and token.

4 DevOps

@ Deployments	Duran and							
🚇 Monitor	Runners	6	niapse					
lnfrastructure	Runners are processes that pick up and execute CI/CD jobs for	GitLab. How do I configure runners?						
🖰 Packages & Registries	Register as many runners as you want. You can register runners	Register as many runners as you want. You can register runners as separate users, on separate servers, and on your local machine.						
Analytics	Runners are either:	Runners are either:						
U Wiki	active - Available to run jobs.							
👗 Snippets	 paused - Not available to run jobs. 							
Settings	Specific runners	Shared runners						
General	These runners are specific to this project.	These runners are shared across this GitLab instance.						
Integrations		characterization of the second second second second						
Webhooks	Entrin - martific manage for a main at	are powered by Google Cloud Platform. Autoscale mode and						
Repository	Install Gitlab Rupper and ensure it's rupping	reduced wait times to spin up builds, and isolated VMs for						
CI/CD	2. Register the runner with this URL:	each project, mus maximizing security.						
Monitor	https://gitlab.com/ 🎝	They're free to use for public open source projects and limited to 400 CL minutes per month per group for private						
Pages	And this registration token:	projects. Read about all GitLab.com plans.						
Packages & Registries								
Usage Quotas	Reset registration token	Enable shared runners for this project						
≪ Collapse sidebar	snow runner installation instructions	Available snared runners: 42						

Create the **values.yaml** file and fill in the following information:

Registration URL
gitlabUrl: https://gitlab.com/
Registration token
runnerRegistrationToken: "******"
rbac:
create: true
runners:
privileged: true

Create a GitLab namespace.

kubectl create namespace gitlab

Install GitLab Runner using Helm.

helm repo add gitlab https://charts.gitlab.io helm install --namespace gitlab gitlab-runner -f values.yaml gitlab/gitlab-runner --version=0.43.1

After the installation, you can obtain the gitlab-runner workload on the CCE console and view the connection information in GitLab later.

Q Learn GitLab 33%		Runners	
Repository		Runners are processes that pick up and execute CI/CD jobs for GitL	ab. How do I configure runners?
Issues		Register as many runners as you want. You can register runners as Runners are either:	separate users, on separate servers, and on your local machine.
Ø CI/CD		active - Available to run jobs.	
Φ Security & Compliance	Discover	 paused - Not available to run jobs. 	
Deployments	Audit events	Specific runners	Shared runners
폪 Monitor	Configuration	These runners are specific to this project	These runners are shared across this Gitl ab instance
lnfrastructure	0	meter annete are speene to ans project	Chand Durante of City of any on the state of a state of a
🖰 Packages & Registries		Set up a specific rupper for a project	are powered by Google Cloud Platform. Autoscaling means
🔟 Analytics		1. Install GitLab Runner and ensure it's running.	reduced wait times to spin up builds, and isolated VMs for
📮 Wiki		2. Register the runner with this URL:	each project, and maximizing security.
X Snippets		https://gitlab.com/ [0]	limited to 400 CI minutes per month per group for private
Settings		And this registration token:	projects. Read about all GitLab.com plans.
General		e ^C	Enable shared runners for this project
Integrations		Reset registration token	
Webhooks		Show runner installation instructions	Available shared runners: 42
Repository			
CI/CD			#11574084 (EuhiQzPR) 3-green shared-gitlab-org runners-manager gitlab com
Monitor		Available specific runners	(gitlab-org)
Pages		#14806619 (Nah2Vmah) A	
Packages & Registries		aitlab-runner-aitlab-runner-7d76d4fdbf-z5afr	 # 12270035 (ZXWGKJAP) 3-blue.shared.runners-manager.gitlab.com/default
Usage Quotas		и на население и се стание и се дин	docker east-c gce git-annex linux mongo mysql postgres ruby shared

Creating an Application

Place the application to be created in the GitLab project repository. This section takes Nginx modification as an example. For details, visit https://gitlab.com/c8147/cidemo/-/tree/main.

The following files are included:

- .gitlab-ci.yml: GitLab CI file, which will be described in detail in Creating a Pipeline.
- **Dockerfile**: used to build Docker images.
- **index.html**: used to replace the index page of Nginx.
- **k8s.yaml**: used to deploy the Nginx app. A Deployment named **nginx-test** and a Service named **nginx-test** will be created.

The preceding files are only examples. You can replace or modify them accordingly.

Configuring Global Variables

When using pipelines, build an image, upload it to SWR, and run kubectl commands to deploy the image in the cluster. Before performing these operations, you must log in to SWR and obtain the credential for connecting to the cluster. You can define the information as variables in GitLab.

Log in to **GitLab**, choose **Settings** > **CI/CD** in the project view, and click **Expand** next to **Variables** to add variables.

2 Security & Compliance 2 Deployments 3 Monitor 3 Infrastructure 3 Packages & Registries		Variables Variables Variables • Pr • Ma	les store information, like passwords au can be: otected: Only exposed to protect sked: Hidden in job logs. Must ma	nd secret keys, that you can use in j ed branches or tags. tch masking requirements. Learn n	ob scripts. Learn	more.		
Analytics	Value stream							
🚽 Wiki	CI/CD		Environment variab	es are configured by your administ	rator to be prote	ected by default.		
X Snippets	Repository	Type	↑ Kev	Value	Protected	Masked	Environments	
Settings			,					
General		Variable	kube_config 🚯	····· 6	~	×	All (default)	Ø
Integrations		Variable	project P	M	~	×	All (default)	A
Webhooks			project (0	L0				0
Repository		Variable	swr_ak 🔁	ß	~	×	All (default)	0
CI/CD								
Monitor		Variable	swr_sk 📴	·····	~	×	All (default)	Ø
Pages								
Packages & Registries		Add var	Reveal values					
Usage Quotas								

• kube_config

kubeconfig.json file used for kubectl command authentication. Run the following command on the host where kubectl is configured to convert the file to the Base64 format:

echo \$(cat ~/.kube/config | base64) | tr -d " "

The command output is the content of **kubeconfig.json**.

• **project**: project name.

Log in to the management console, hover the cursor on your username in the upper right corner, and choose **My Credentials**. In the **Projects** area on the **API Credentials** page, check the name of the project in your current region.

• swr_ak: access key.

Log in to the management console, hover the cursor on your username in the upper right corner, and choose **My Credentials**. In the navigation pane, choose **Access Keys**. Click **Create Access Key**, enter the description, and click **OK**. In the displayed **Information** dialog box, click **Download**. After the certificate is downloaded, obtain the AK and SK information from the **credentials** file.

• **swr_sk**: secret key for logging in to SWR.

Run the following command to obtain the key pair. Replace *\$AK* and *\$SK* with the AK and SK obtained in the preceding steps.

printf "*\$AK*" | openssl dgst -binary -sha256 -hmac "*\$SK*" | od -An -vtx1 | sed 's/[\n]//g' | sed 'N;s/\n//'

The command output displays the login key pair.

Creating a Pipeline

Log in to GitLab and add the .gitlab-ci.yml file to Repository.

(Project information		Now file				
(🛛 Learn GitLab	33%	New me	ile			
(Repository		₽ main	n / File name			
	Files		1				
	Commits		1	.gitignore			
	Branches			✓ .gitlab-ci.yml			
	Tags			.metrics-dashboard.yml			
	Contributors			Dockerfile			
	Granh			LICENSE			
	Commerce						
	Compare	-					
l	V Issues	0					
ş	 Merge requests 	1					
4	🕼 CI/CD						
(Security & Complian	nce					
ć	Deployments						
F	W Monitor						

The content is as follows:

Define pipeline stages, including package, build , and deploy .
stages:
- package
- build
- deploy
If no image is specified in each stage, the default image docker:latest is used.
mage: swr.ap-southeast-3.myhuaweicloud.com/container/docker:latest
In the package stage, only printing is performed.
backage:
stage: package
script:
- echo "package"
In the build stage, the Docker-in-Docker mode is used.
build:
stage: build
Define environment variables for the build stage.
variables:
DOCKER_HOST: tcp://docker:2375
Define the image for running Docker-in-Docker.
services:
- docker:18.09-dind
script:
- echo "build"
Log in to SWR.
- docker login -u \$project@\$swr_ak -p \$swr_sk swr.ap-southeast-3.myhuaweicloud.com
Build an image. k8s-dev is the organization name in SWR. Replace it to the actual name.
- docker build -t swr.ap-southeast-3.myhuaweicloud.com/ <i>k8s-dev</i> /nginx:\$CI_PIPELINE_ID .
Push the image to SWR.
```
- docker push swr.ap-southeast-3.myhuaweicloud.com/k8s-dev/nginx:$CI_PIPELINE_ID
deploy:
 # Use the kubectl image.
 image:
  name: swr.ap-southeast-3.myhuaweicloud.com/container/kubectl:latest
  entrypoint: [""]
 stage: deploy
 script:
  # Configure the kubeconfig file.
  - mkdir -p $HOME/.kube
  - export KUBECONFIG=$HOME/.kube/config
  - echo $kube_config |base64 -d > $KUBECONFIG
  # Replace the image in the k8s.yaml file.
  - sed -i "s/<IMAGE_NAME>/swr.ap-southeast-3.myhuaweicloud.com\/k8s-dev\/nginx:$CI_PIPELINE_ID/g"
k8s.yaml
  - cat k8s.yaml
  # Deploy an application.
  - kubectl apply -f k8s.yaml
```

After the **.gitlab-ci.yml** file is saved, the pipeline is started immediately. You can view the pipeline execution status in GitLab.

2 CI/CD										
Pipelines	११ No related merge requests f	የን No related merge requests found.								
Editor										
Jobs	Pipeline Needs Jobs 3 Tests 0									
Schedules										
Φ Security & Compliance	Package	Build	Deploy							
ව Deployments	Dackage C	build C	anlay C							
💬 Monitor	o package		U depidy							
lnfrastructure										

Verifying Deployment

After the pipeline is deployed, locate the **nginx-test** Service on the CCE console, query its access address, and run the **curl** command to access the Service.

curl *xxx.xxx.xxx.xxx*.31111 Hello Gitlab!

If the preceding information is displayed, the deployment is correct.

Common Issues

• If the following problem occurs during the deployment:

	78	Getting source from Git repository							
	79	Fetching changes with git depth set to 20							
		Initialized empty Git repository in /builds/hw65/gitlab-cce-cicd-demo/.git/							
	81	Created fresh repository.							
	82	Checking out a9c9f90b as main							
	83	Skipping Git submodules setup							
\sim		Executing "step_script" stage of the job script							
	86	<pre>\$ echo \$kube_config base64 -d > \$KUBECONFIG</pre>							
	87	/scripts-43497556-3766012849/step_script: line 143: \$KUBECONFIG: ambiguous redirect							
\sim	89	Cleaning up project directory and file based variables							
	91	ERROR: Job failed: command terminated with exit code 1							

.

76	\$ kubectl apply -f k8s.yaml
	E0215 08:03:55.297105 19 memcache.go:255] couldn't get resource list for proxy.exporter.k8s.io/v1beta1:
	Got empty response for: proxy.exporter.k8s.io/v1beta1
	Error from server (Forbidden): error when retrieving current configuration of:
	Resource: "apps/v1, Resource=deployments", GroupVersionKind: "apps/v1, Kind=Deployment"
	Name: "nginx-test", Namespace: "gitlab"
81	from server for: "k8s.yaml": deployments.apps "nginx-test" is forbidden: User "system:serviceaccount:gitlab:
	default" cannot get resource "deployments" in API group "apps" in the namespace "gitlab"
82	Error from server (Forbidden): error when retrieving current configuration of:
	Resource: "/v1, Resource=services", GroupVersionKind: "/v1, Kind=Service"
	Name: "nginx-test", Namespace: "gitlab"
	from server for: "k8s.yaml": services "nginx-test" is forbidden: User "system:serviceaccount:gitlab:default"
	cannot get resource "services" in API group "" in the namespace "gitlab"
87	Cleaning up project directory and file based variables 00:01
89	ERROR: Job failed: command terminated with exit code 1

Check whether the following commands are missing in the **.gitlab-ci.yml** file. If yes, add them to the **.gitlab-ci.yml** file.

...
script:
 # Configure the kubeconfig file.
 - mkdir -p \$HOME/.kube
 export KUBECONFIG=\$HOME/.kube/config
 echo \$kube_config |base64 -d > \$KUBECONFIG
 # Replace the image in the k8s.yaml file.
...
If Docker cannot be executed, information similar to the following will display.

30	Configure a credential helper to remove this warning. See
31	<u>https://docs.docker.com/engine/reference/commandline/login/#credentials-sto</u>
1	<u>re</u>
32	Login Succeeded
33	<pre>\$ docker build -t swr.</pre>
	IPELINE_ID .
34	Cannot connect to the Docker daemon at tcp://docker:2375. Is the docker dae
	mon running?
36	Cleaning up project directory and file based variables
38	ERROR: Job failed: command terminated with exit code 1

The **privileged: true** parameter fails to be transferred during GitLab Runner installation. As a result, you do not have the permissions to run the Docker command. To resolve this issue, find GitLab Runner in the workload list on the CCE console, add the environment variable **KUBERNETES_PRIVILEGED**, and set its value to **true**.

Container Settings									
Container Information	Container - 1	1							
	Basic Info Lifecycle Health Check	Environment variables affect the way a running container will behave. You can modify created variables as required after deploying the workload. How Do IAdd Environment Variables? Adding a Variable Edding Custom Variables in Batches							
	Environment	Туре		Variable Name		Variable Value/Reference	Operation		
	Variables Data Storage Security	Custom	 	CL_SERVER_URL CLONE_URL		https:/igitab.com/	Delete		
	Logging	Custom	~	RUNNER_EXECUTOR		kubernetes	Delete		
		Custom	~	REGISTER_LOCKED		true	Delete		
		Custom	~	RUNNER_TAG_LIST			Delete		
		Custom	~	KUBERNETES_PRIVILEGED		true	Delete		

4.3 Continuous Delivery Using Argo CD

Background

ArgoCD is a declarative, GitOps continuous delivery tool for Kubernetes. Argo CD automates the deployment of applications to Kubernetes.

Figure 4-17 Argo CD workflow



This section describes how to interconnect ArgoCD with CCE to perform continuous deployment.

Preparations

- 1. Create a CCE cluster and a node and bind an EIP to the node for downloading an image during Argo CD installation.
- 2. Create an ECS, bind an EIP to the ECS, and download and configure kubectl to connect to the cluster. For details, see **Connecting to a Cluster Using kubectl**.

3. Prepare an application in the Git repository. This section uses the Nginx sample application in the https://gitlab.com/c8147/examples.git repository.

Installing Argo CD

Step 1 Install the Argo CD server in the cluster.

kubectl create namespace argocd # kubectl apply -n argocd -f https://raw.githubusercontent.com/argoproj/argo-cd/v2.4.0/manifests/ install.yaml

Check whether the installation is successful. If all pods in the **argocd** namespace are in the **Running** status, the installation is successful.

# kubectl get pod -A		
NAMESPACE NAME	READY STATUS RESTARTS AGE	
argocd argocd-application-controller-0	1/1 Running 0 8m32s	
argocd argocd-applicationset-controller-789457b	498-6n6l5 1/1 Running 0 8m32s	
argocd argocd-dex-server-748bddb496-bxj2c	1/1 Running 0 8m32s	
argocd argocd-notifications-controller-8668ffdd7	5-q7wdb 1/1 Running 0 8m32s	
argocd argocd-redis-55d64cd8bf-g85np	1/1 Running 0 8m32s	
argocd argocd-repo-server-778d695657-skprm	1/1 Running 0 8m32s	
argocd argocd-server-59c9ccff4c-vd9ww	1/1 Running 0 8m32s	

Run the following commands to change the Service type of **argocd-server** to **NodePort**:

kubectl patch svc argocd-server -n argocd -p '{"spec": {"type": "NodePort"}}'
service/argocd-server patched

Check the result.

# kubectl -n argocd get svc					
NAME	TYPE CL	USTER-IP	EXTERNAL-IP	PORT(S)	AGE
argocd-applicationset-controller	Cluster	IP 10.247.23	37.53 <none< td=""><td>> 7000/TCP,8080/T</td><td>CP</td></none<>	> 7000/TCP,8080/T	CP
18m					
argocd-dex-server	ClusterIP	10.247.164.1	11 <none></none>	5556/TCP,5557/TCP,5	5558/TCP
18m					
argocd-metrics	ClusterIP 1	0.247.138.98	<none></none>	8082/TCP	18m
argocd-notifications-controller-r	netrics Clust	erIP 10.247.	239.85 <nor< td=""><td>ne> 9001/TCP</td><td></td></nor<>	ne> 9001/TCP	
18m					
argocd-redis	ClusterIP 10).247.220.90	<none></none>	6379/TCP	18m
argocd-repo-server	ClusterIP	10.247.1.142	<none></none>	8081/TCP,8084/TCP	18m
argocd-server	NodePort	10.247.57.16	<none></none>	80:30118/TCP,443:31	221 /TCP
18m				, .	
argocd-server-metrics	ClusterIP	10.247.206.	190 <none></none>	8083/TCP	18m
5					

To access Argo CD using the **argocd-server** Service, use *Node IP.Port number*. In this example, the port number is **31221**.



The login username is **admin**, and the password can be obtained by running the following command:

kubectl -n argocd get secret argocd-initial-admin-secret -o jsonpath="{.data.password}" | base64 -d;echo

Step 2 Install the Argo CD client on the ECS.

wget https://github.com/argoproj/argo-cd/releases/download/v2.4.0/argocd-linux-amd64
cp argocd-linux-amd64 /usr/local/bin/argocd
chmod +x /usr/local/bin/argocd

Run the following commands. If the following information is displayed, the installation is successful.

argocd version argocd: v2.4.0+91aefab BuildDate: 2022-06-10T17:44:14Z GitCommit: 91aefabc5b213a258ddcfe04b8e69bb4a2dd2566 GitTreeState: clean GoVersion: go1.18.3 Compiler: gc Platform: linux/amd64 FATA[0000] Argo CD server address unspecified

----End

Deploying an Application Using Argo CD

Step 1 Add a CCE cluster to Argo CD.

- 1. Log in to an ECS.
- 2. Check the kubectl context configuration. # kubectl config get-contexts CURRENT NAME CLUSTER AUTHINFO NAMESPACE * internal internalCluster user
- 3. Log in to the Argo CD server. The username is **admin**. You can obtain the server IP address and password from **Step 1**. If the ECS and the cluster are in the same VPC, the node IP address can be a private IP address. argocd login *<Node IP address:Port number>* --username *admin* --password *<password>*

Information similar to the following is displayed:

argocd login **192.168.0.52:31221** --username **admin** --password ********* WARNING: server certificate had error: x509: cannot validate certificate for 192.168.0.52 because it doesn't contain any IP SANs. Proceed insecurely (y/n)? y 'admin:login' logged in successfully Context '192.168.0.52:31221' updated

4. Add a CCE cluster.

argocd cluster add *internal* --kubeconfig /*root/.kube/config* --name *argocd-01*

In the preceding command, *internal* is the context name queried in **Step 1.2**, */root/.kube/config* is the path of the kubectl configuration file, and *argocd-01* is the cluster name defined in Argo CD.

Information similar to the following is displayed:

WARNING: This will create a service account `argocd-manager` on the cluster referenced by context `internal` with full cluster level privileges. Do you want to continue [y/N]? y INFO[0002] ServiceAccount "argocd-manager" already exists in namespace "kube-system" INFO[0002] ClusterRole "argocd-manager-role" updated INFO[0002] ClusterRoleBinding "argocd-manager-role-binding" updated Cluster "https://192.168.0.113:5443"" added

Log in to the Argo CD page. You can see that the connection is successful.

Settings / Clusters				
\diamond	NAME	URL	VERSION	CONNECTION STATUS
	දී argocd-01	https://192.168.0.113:5443	1.21+	Successful
	දුරිදු in-cluster	https://kubernetes.default.svc		O Unknown
9				

Step 2 Connect to the Git repository.

argocd repo add *https://gitlab.com/c8147/examples.git* --username *<username>* --password *<password>*

In the preceding command, *https://gitlab.com/c8147/examples.git* indicates the repository address, and *<username>* and *<password>* indicate the repository login username and password. Replace them with the actual values.

Information similar to the following is displayed:

Repository 'https://gitlab.com/c8147/cidemo.git' added

Log in to the Argo CD page. You can see that the cluster has been added.

V2.4.0+5	Settings / Repositories + CONNECT REPO USING SSH + CONNECT REPO USING HT	TPS	+ CONNECT REPO	USING GITHUB APP	C REFRESH LIST	
			TYPE	NAME	REPOSITORY	CONNECTION STATUS
		٨	git		https://gitlab.com/c8147/examples.git	Successful
¢?						
9						

Step 3 Add an application to Argo CD.

argocd app create nginx --repo *https://gitlab.com/c8147/examples.git* --path *nginx* --dest-server *https://192.168.0.113:5443* --dest-namespace *default*

https://gitlab.com/c8147/examples.git indicates the repository address, *nginx* indicates the repository path, *https://192.168.0.113:5443* indicates the address of the cluster to be deployed, and *default* indicates the namespace.

In this example, the nginx directory in the GitLab repository contains a YAML file of the Nginx application. The file includes a Deployment and a Service.

After the application is created, you can view its details.

# argocd	app list							
NAME C	LUSTER	NAMESPACE	PROJECT	STATUS	HEALT	H SYNC	POLICY CON	IDITIONS
REPO		PATH TARGET						
nginx htt	ps://192.168.0.113:5	443 default d	lefault C	OutOfSync	Missing	<none></none>	<none></none>	https://
gitlab.con	n/c8147/examples.g	t nginx			•			

Log in to the Argo CD page. You can see that the application has been added.

	Applications		
€ v2.4.0+9	+ NEW APP	NC APPS	C REFRESH APPS Q Search applications 7
	FILTERS		
 ~	🔲 🌟 Favorites Only	,	
₹¢₿	SYNC STATUS		Project: default
	🗌 🔿 Unknown	0	Labels:
9	🔲 < Synced	0	Reposito https://gitlab.com/c8147/examples.git
_	🔲 📀 OutOfSync	1	Target R Path: nginx
? <u>=</u>			Destinati argood-01
	HEALTH STATUS	•	Namesp default
	🔲 📀 Unknown	0	
	Progressing	0	

Step 4 Synchronize the application.

Synchronize the application and deploy it on a specified cluster. Run the following commands:

# argocd app sync nginx								
TIMESTAMP	GROUP	KIND NAME	ESPACE	NAME	STATUS	HEALTH		
HOOK MESSA	.GE							
2022-10-24T12	2:15:10+08:00	Service defa	ult	nginx OutO	fSync Miss	sing		
2022-10-24T12	2:15:10+08:00 apps	Deployment	default	nginx	OutOfSync	Missing		
2022-10-24T12	2:15:10+08:00	Service defa	ult	nginx Syne	ed Health	y		
2022-10-24T12	2:15:10+08:00	Service defa	ult	nginx Syno	ed Health	ny :	service/	
nginx created				5 5		5		
2022-10-24T12	2:15:10+08:00 apps	Deployment	default	nginx	OutOfSync	Missing		
deployment.ap	ps/nginx created			5	2	5		
2022-10-24T12	2:15:10+08:00 apps	Deployment	default	nainx	Svnced Pr	roaressina		
deployment.ap	ps/nginx created					- <u>j</u> j		
	p=,g							
Name:	nainx							
Proiect:	default							
Server:	https://192.168.0.11	3:5443						
Namespace:	default							
URI:	https://192.168.0.17	8:32459/applica	ations/nainx					
Repo:	https://gitlab.com/c	8147/examples	.ait					
Target:	······································	····, ····	.9					
Path:	nainx							
SyncWindow:	Svnc Allowed							
Sync Policy:	<none></none>							
Sync Status	Synced to (dd150	906)						
Health Status:	Progressing	,00)						
ricultin Status.	riogressing							
Operation [.]	Sync							
Sync Revision	dd1590679856bd	19288036847bd	-4a5556c1692	67				
Phase	Succeeded	15200050047.000	1055001052					
Start	2022-10-24 12:15:10	T22 0080+						
Finished	2022 10 24 12:13:10	10 +0800 CST						
Duration:	Ωs	10 10000 051						
Message:	successfully synce	d (all tasks run))					
wiessage.	successfully synce)					
	ΝΔΜΕΣΡΔΟΕ Ν	ΔΜΕ ΣΤΔΤΙΙς	μεδιτή		AGE			
Service	default noiny Syn	red Healthy	service/	nainy created	NGL .			
anns Denlovn	nent default noin	Svnced Progr	essina de	nginx created	ns/nainy cre	eated		
apps Deployin	iene derdate fight/	Cogneca riogr	cooring ut	proyment.up		catea		

You can see that an Nginx workload and a Service are deployed in a CCE cluster.

kubectl get deployment NAME READY UP-TO-DATE AVAILABLE AGE nginx 1/1 1 1 2m47s # kubectl get svc NAME TYPE CLUSTER-IP EXTERNAL-IP PORT(S) AGE kubernetes ClusterIP 10.247.0.1 <none> 443/TCP 5h12m nginx ClusterIP 10.247.177.24 <none> 80/TCP 2m52s

Log in to the Argo CD page. You can see that the application status has changed to **Synced**.

	Applications							
€ v2.4.0+9	+ NEW APP & SYN	C APPS C REFRESH APPS Q Search applications						
	T FILTERS							
	🗌 🌟 Favorites Only							
₹¢₿	SYNC STATUS	Project: default						
	🗌 🔿 Unknown	0 Labels: Status: ♥ Healthy ♥ Synced						
	🔲 < Synced	1 Reposito https://gitlab.com/c8147/examples.git						
	OutOfSync	0 Target R_ Path: nginx						
		Destinati argocd-01						
	HEALTH STATUS	A Namesp default						
	🔲 🔞 Unknown	0 SYNC CREFRESH ODELETE						
	Progressing	0						

----End

Using Argo Rollouts for Grayscale Release

Argo Rollouts is a Kubernetes controller that provides advanced deployment capabilities such as blue-green, grayscale (canary) release, and progressive delivery.

Step 1 Install an argo-rollouts server in the cluster.

kubectl create namespace argo-rollouts

kubectl apply -f https://github.com/argoproj/argo-rollouts/releases/download/v1.2.2/install.yaml -n argo-rollouts

NOTE

If the application is deployed in multiple clusters, install the argo-rollouts server in each target cluster.

Step 2 Install the kubectl add-on of argo-rollouts on the ECS.

curl -LO https://github.com/argoproj/argo-rollouts/releases/download/v1.2.2/kubectl-argo-rollouts-linuxamd64

- # chmod +x ./kubectl-argo-rollouts-linux-amd64
- # sudo mv ./kubectl-argo-rollouts-linux-amd64 /usr/local/bin/kubectl-argo-rollouts

Run the following commands to check whether the add-on has been installed:

```
# kubectl argo rollouts version
kubectl-argo-rollouts: v1.2.2+22aff27
BuildDate: 2022-07-26T17:24:43Z
GitCommit: 22aff273bf95646e0cd02555fbe7d2da0f903316
GitTreeState: clean
GoVersion: go1.17.6
Compiler: gc
Platform: linux/amd64
```

Step 3 Prepare two sample Nginx application images whose versions are v1 and v2, respectively. The welcome pages are displayed as "nginx:v1!" and "nginx:v2!", respectively.

Create a Dockerfile. The content of the Dockerfile for v1 is as follows: (For v2, replace **nginx:v1!** with **nginx:v2!**.) FROM nginx:latest RUN echo '<h1>*nginx:v1!*</h1>' > /usr/share/nginx/html/index.html

Create a v1 image.

docker build -t nginx:v1 .

Log in to SWR and push the image to SWR. For details, see **Uploading an Image Through a Container Engine Client**. During the push, **container** indicates the organization name in SWR. Set it as required.

docker login -u {region}@xxx -p xxx swr.{region}.myhuaweicloud.com docker tag nginx:v1 swr.cn-east-3.myhuaweicloud.com/*container*/nginx:v1 docker push swr.cn-east-3.myhuaweicloud.com/*container*/nginx:v1

Create a v2 image and push it to SWR in the same way.

Step 4 Deploy an Argo Rollouts controller. In this example, the controller first shifts 20% of all service traffic to the new version. Then, manually increase the traffic proportion. After that, the controller automatically and gradually increases traffic until the release is complete.

Create a file named rollout-canary.yaml:

```
apiVersion: argoproj.io/v1alpha1
.
kind: Rollout
metadata:
 name: rollout-canary # Custom Rollout name
spec:
 replicas: 5 # Five replicas
 strategy: # Upgrade policy
  canary: # Grayscale (canary) release
   steps: # Release pace (duration can be set for each phase)
    - setWeight: 20 # Traffic weight
    - pause: {} # If this field is not specified, the release is paused.
   - setWeight: 40
   - pause: {duration: 10} # Pause duration, in seconds.
    - setWeight: 60
    - pause: {duration: 10}
    - setWeight: 80
    - pause: {duration: 10}
 revisionHistoryLimit: 2
 selector:
  matchLabels:
    app: rollout-canary
 template:
  metadata:
    labels:
     app: rollout-canary
  spec:
    containers:
    - name: rollout-canary
     image: swr.cn-east-3.myhuaweicloud.com/container/nginx:v1 # The pushed image, whose version is
v1.
     ports:
     - name: http
      containerPort: 80
      protocol: TCP
     resources:
      requests:
        memory: 32Mi
        cpu: 5m
    imagePullSecrets:
     - name: default-secret
apiVersion: v1
kind: Service
metadata:
 name: rollout-canary
spec:
 type: NodePort
 ports:
 - port: 80
  targetPort: 80
  nodePort: 31270 # The custom node port number
```

selector: app: rollout-canary

Run the following command to create the preceding two resource objects:

kubectl apply -f rollout-canary.yaml

NOTE

The Argo Rollouts controller does not trigger upgrade during initial creation and the configured release policy does not take effect. Only the number of replicas increases to 100% immediately.

Step 5 Argo Rollouts visualizes the rollout process and related resource objects to display real-time changes. You can run the **get rollout --watch** command to observe the deployment process, for example:

kubectl argo rollouts get rollout *rollout-canary* --watch

In the preceding command, *rollout-canary* indicates the custom Rollout name.

NAME	KIND	STATUS	AGE	INFO
🗘 rollout-canary	Rollout	🗸 Healthy	8s	
└──# revision:1				
└───ि rollout-canary-d66dc64d	ReplicaSet	🗸 Healthy	8s	stable
├──□ rollout-canary-d66dc64d-bw6sm	Pod	🗸 Running	8s	ready:1/1
├──□ rollout-canary-d66dc64d-h89kl	Pod	🗸 Running	8s	ready:1/1
├──□ rollout-canary-d66dc64d-kwwnm	Pod	🗸 Running	8s	ready:1/1
├──□ rollout-canary-d66dc64d-r6zw4	Pod	🗸 Running	8s	ready:1/1
└──□ rollout-canary-d66dc64d-rbsr6	Pod	🗸 Running	8s	ready:1/1

- Step 6 After the creation is complete, you can access the Nginx application using Node EIP:Port number. The port number is specified in the Service resource in the rollout-canary.yaml file. In this example, the port number is 31270.
- **Step 7** Use the v2 image to update the application.

kubectl argo rollouts set image rollout-canary rollout-canary=swr.cn-east-3.myhuaweicloud.com/
container/nginx:v2

The controller will update the application according to the update policy. In this example, a 20% traffic weight is set in the first step, and the release is paused until the user cancels or continues. You can run the following command to view the detailed process. The release is paused.

kubectl argo rollouts get rollout *rollout-canary* --watch

You can view that only one of the five replicas runs the new version, that is, the weight of 20% defined in **setWeight: 20**.

NAME	KIND	STATUS	AGE	INFO
🔿 rollout-canary	Rollout	I Paused	49s	
├──# revision:2				
└──── rollout-canary-8644685965	ReplicaSet	🗸 Healthy	9s	canary
└──□ rollout-canary-8644685965-9mvvw	Pod	🗸 Running	9s	ready:1/1
└──# revision:1				
🖵 🗗 rollout-canary-d66dc64d	ReplicaSet	🗸 Healthy	49s	stable
├──□ rollout-canary-d66dc64d-h89kl	Pod	🗸 Running	49s	ready:1/1
├──□ rollout-canary-d66dc64d-kwwnm	Pod	🗸 Running	49s	ready:1/1
├──□ rollout-canary-d66dc64d-r6zw4	Pod	🗸 Running	49s	ready:1/1
_ □ rollout-canary-d66dc64d-rbsr6	Pod	🗸 Running	49s	ready:1/1
└──□ rollout-canary-d66dc64d-rbsr6	Pod	✓ Running	49s	ready:1/1

If you run the following command for multiple times, 20% of the response results are the response of v2.

for i in {1..10}; do curl *<Node EIP:Port number>*; done;

Verification result:

<h1>nginx:v2!</h1> <h1>nginx:v1!</h1> <h1>nginx:v1!</h1> <h1>nginx:v1!</h1> <h1>nginx:v1!</h1> <h1>nginx:v1!</h1> <h1>nginx:v1!</h1> <h1>nginx:v1!</h1> <h1>nginx:v2!</h1> <h1>nginx:v2!</h1>

Step 8 Manually update the version.

kubectl argo rollouts promote *rollout-canary*

In this example, the remaining steps are fully automated until the release is complete. Run the following command to view the detailed process. The controller gradually switches all traffic to the new version.

kubectl argo rollouts get rollout rollout-canary --watch

NAME	KIND	STATUS	AGE	INFO
🔿 rollout-canary	Rollout	🗸 Healthy	2m16s	
├──# revision:2				
│ └──चिrollout-canary-8644685965	ReplicaSet	🗸 Healthy	96s	stable
	Pod	🗸 Running	96s	ready:1/1
⊢ rollout-canary-8644685965-z9nbw	Pod	🗸 Running	48s	ready:1/1
	Pod	🗸 Running	37s	ready:1/1
	Pod	🗸 Running	27s	ready:1/1
∣ └──□ rollout-canary-8644685965-frblf	Pod	🗸 Running	17s	ready:1/1
└──# revision:1				
└───ि rollout-canary-d66dc64d	ReplicaSet	 ScaledDown 	2m16s	

Step 9 You can run the following command to use more Argo Rollouts functions, such as terminating or rolling back a release: kubectl argo rollouts --help

----End

4.4 Implementing Separate DevOps Processes for Multiple Clusters Using Jenkins and GitLab

4.4.1 Solution Overview

DevOps is a set of processes, methods, and systems that promote close communication, efficient collaboration, and integration between development (applications or software engineering), technical operations, and quality assurance (QA) departments. By using automatic software delivery and architecture change processes, DevOps enables faster, more frequent, and more reliable planning, development, build, tests, releases, deployment, and maintenance, resulting in stable and reliable development outcomes. As microservices and middle-end architectures continue to emerge, DevOps becomes increasingly crucial.

Solution Architecture

This solution uses GitLab and Jenkins to automate the building and continuous deployment of containerized applications. GitLab handles source code

management and versioning, while Jenkins is responsible for building and deploying the containerized applications. To maintain high isolation between the production and testing environments, two separate Kubernetes clusters are used, one for each environment.

The following shows an example of the complete process, starting from source code compilation and image building, to application testing, production, and rollout:

- 1. Create a Git repository in GitLab and associate it with Jenkins.
- 2. Create a build job in Jenkins and allow it to use the Git repository of GitLab.
- 3. In the build job, configure the compiler and build parameters so that the source code can be obtained from the Git repository and compiled.
- 4. Push the compiled image to the SWR image repository.
- 5. Deploy the image in a Kubernetes cluster in the testing environment.
- 6. After the image passes the test and is approved, deploy it in a Kubernetes cluster in the production environment.

Figure 4-18 Architecture



Production environment

Solution Highlights

- The entire process, from code submission to deployment and rollout, is automated, resulting in a significant improvement in delivery efficiency.
- Containerized applications are ready to use out-of-the-box, and they can be reused at a low cost.
- To ensure that the testing and production environments operate independently and do not interfere with each other, multiple Kubernetes clusters are isolated.

4.4.2 Resource Planning

In this example, you need to create a VPC, an ECS, a CCE cluster, and a VPC peering connection. For details about the resource planning, see **Table 4-6**.

NOTE

The following resource planning details are only examples for your reference. You need to plan resources based on actual service requirements.

Table 4-6	Resource	and	cost	planning
-----------	----------	-----	------	----------

Resource	Description
VPC	 In this example, there are three VPCs, including the VPC where the ECS resides and the VPC where the testing cluster and production cluster reside. These VPCs are located in the same region, and the subnet CIDR blocks of these VPCs do not overlap. vpc-X: 192.168.0.0/16 (VPC where the ECS with GitLab and Jenkins installed resides) vpc-A: 172.16.0.0/16 (VPC to which the testing cluster resides)
	 vpc-B: 172.17.0.0/16 (VPC where the production cluster resides)
ECS	 In this example, there is one ECS located in vpc-X (with CIDR block of 192.168.0.0/16). Node flavor: 4 vCPUs 16 GiB OS: Huawei Cloud EulerOS 2.0 EIP: An EIP is automatically assigned to the node to access the public network and pull images.
CCE cluster	 In this example, there are two CCE clusters, including the testing and production clusters. The following lists some key parameter configurations. (You can configure other parameters as required or use their default values.) Cluster type: CCE Turbo cluster VPC: The testing cluster is located in vpc-A (172.16.0.0/16), and the production cluster is located in vpc-B (172.17.0.0/16). Actual number of nodes: 1 per cluster Node configurations: Node flavor: 4 vCPUs 16 GiB OS: Huawei Cloud EulerOS 2.0 EIP: An EIP is automatically assigned to the node to access the public network and pull images.

Resource	Description
VPC peering connection	In this example, there are two VPC peering connections. The network connection requirements are as follows:
	• peer-XA: connects vpc-X to vpc-A.
	• peer-XB: connects vpc-X to vpc-B.

4.4.3 Procedure

4.4.3.1 Setting Up the Jenkins and GitLab Environments

Prerequisites

- A new VPC has been created. In this example, the VPC is named **vpc-X** and the CIDR block is **192.168.0.0/16**.
- An ECS in **vpc-X** (with CIDR block of **192.168.0.0/16**) has been created. The recommended flavor for this ECS is 4 vCPUs and 16 GiB of memory, and the ECS runs Huawei Cloud EulerOS 2.0. An EIP has been bound to the ECS to pull images from the Internet.

Installing Docker of a Specified Version

- **Step 1** Log in to the ECS.
- **Step 2** Quickly install Docker of the latest version on the device running Huawei Cloud EulerOS 2.0. You can also manually install Docker. For details, see **Docker Engine installation**.

dnf install docker

Step 3 Check whether Docker is installed. docker info



----End

Installing and Configuring GitLab

- **Step 1** Pull the GitLab image. docker pull gitlab/gitlab-ce
- Step 2 Run the container. docker run -d -p 443:443 -p 80:80 -p 222:22 --name gitlab --restart always -v /home/gitlab/config:/etc/ gitlab -v /home/gitlab/logs:/var/log/gitlab -v /home/gitlab/data:/var/opt/gitlab gitlab/gitlab-ce
- **Step 3** Check the container status.

docker ps | grep gitlab

cooth - J# docker ps | grep gillab B4e5a549e8 gillab/gillab-ce ''_assets/wrapper' 11 seconds ago Up 10 seconds (health: starting) 0.0.0.0:00->80/tcp, :::80->80/tcp, 0.0.0:043->443/tcp, :::443->443/tcp, 0.0200 0.010 u.uc00 u.uc00 u.uc01 u.uc01

- Step 4 Configure gitlab.rb (with the node path of /home/gitlab/config/gitlab.rb). GitLab generates a project URL based on the container's host name (that is, the container ID) when a project is created. However, a fixed URL is required for the GitLab server. To achieve this, you must configure gitlab.rb.
 - 1. Run the **vi** command on the node to open **/home/gitlab/config/gitlab.rb**. vi /home/gitlab/config/gitlab.rb

```
Add the following content to it:
external_url 'http://**.**.***' //External IP address of the node, for example, EIP
gitlab_rails['gitlab_ssh_host'] = '**.**.**' //External IP address of the node, for example, EIP
gitlab_rails['gitlab_shell_ssh_port'] = 222 //This port is mapped when the container is started. (222-
>22)
```

- 2. Save the changes.
- **Step 5** Restart the container. docker restart gitlab
- Step 6 Use a browser to access ECS EIP:80 and log in to the GitLab service.

The default username is **root**, and the initial login password is stored in the **/etc/ gitlab/initial_root_password** directory of the GitLab container.

You can run the following command to obtain the password: docker exec gitlab cat /etc/gitlab/initial_root_password

```
----End
```

Installing and Configuring Jenkins

- **Step 1** Pull the Jenkins image. docker pull jenkins/jenkins:lts
- **Step 2** Run the Jenkins container.

To execute commands like **docker build** within the container, the container must have the capability of docker in docker (dind). This requires mounting **docker.sock** and **docker** to the container.

docker run -d -p 8000:8080 -p 50000:50000 \ -v /var/jenkins_home:/var/jenkins_home \ -v /var/run/docker.sock:/var/run/docker.sock \ -v /usr/bin/docker:/usr/bin/docker \ --name myjenkins --privileged=true -u root jenkins/jenkins:lts

Step 3 Test whether the docker command can be executed in the Jenkins container. docker exec myjenkins docker ps

[root@ecs	iii			
[root@ecs-wine]]]	🖬 ~]# docker exec myje	enkins docker ps		
CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS
1a667946f1d8	jenkins/jenkins:lts	"/usr/bin/tini /u…"	3 hours ago	Up 3 hours
1ab337dbe510	gitlab/gitlab-ce	"/assets/wrapper"	3 hours ago	Up 3 hours (healthy)
[root@edga_land]]	iiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiii			

If the error message "docker: error while loading shared libraries: libltdl.so.7: cannot open shared object file: No such file or directory" is displayed, run the following command to address this issue:

docker exec myjenkins sh -c "apt-get update && apt-get install -y libltdl7"

Step 4 Use a browser to access ECS EIP:8000 and log in to the Jenkins service.

The default username is **root**, and the initial login password is stored in **the /var/jenkins_home/secrets/initialAdminPassword** directory of the container.

You can run the following command to obtain the password:

docker exec myjenkins cat /var/jenkins_home/secrets/initialAdminPassword

Step 5 Install the recommended plugins.

After the initial configuration is complete, the Jenkins page is displayed.

Getting Started

Customize Jenkins

Plugins extend Jenkins with additional features to support many different needs.

Install suggested plugins

Install plugins the Jenkins community finds most useful.

Select plugins to install

Select and install plugins most suitable for your needs.

kins	Q
Item	Welcome to Jenkins!
l History age Jenkins	This page is where your Jenkins Jobs will be displayed. To get started, you can s distributed builds or start building a software project.
ews	Start building your software project
~	Create a job
queue. Status	Set up a distributed build
	Set up an agent
	Configure a cloud
	Learn more about distributed builds

Step 6 Install the kubectl command line tool.

1. Download kubectl to the local PC.

It is recommended that kubectl be of the same version as the cluster to be used. For details, see **kubectl**.

wget https://dl.k8s.io/release/v*.*.*/bin/linux/amd64/kubectl // v*.*.* specifies the cluster version.

2. Copy kubectl to the Jenkins container. docker cp kubectl myjenkins:/usr/bin/ docker exec myjenkins chmod +x /usr/bin/kubectl

Step 7 Configure Jenkins to obtain GitLab code without a password.

1. Obtain the SSH public key. docker exec -it myjenkins ssh-keygen -t rsa

Press **Enter** to skip configuring other parameters.

You do not have to adjust the settings for parameters that are already set to their default values.

Check the generated public key.

docker exec myjenkins cat /root/.ssh/id_rsa.pub

2. Search for SSH in GitLab and add a new key.

⊌ □ + 🔾	User Settings / SSH Keys
D 11 C	Q Search page
User settings	SSH Keys SSH keys allow you to establish a secure connection between your computer and ORLab. SSH fingerprints verify that the client is connecting to the correct host. Check the current instance configuration.
8* Account 88 Applications	Your SSH keys 🖉 0
💭 Chat	There are no SSH keys with access to your account.
Access Tokens	
🖾 Emails	
Password	
D. Notifications	
₽ SSH Keys	
₽ GPG Keys	
2 ² Preferences	
Comment Templates	
Active Sessions	
Authentication Log	

3. Enter an SSH public key.

	Q Search page
Q Search or go to	SSH Kave
User settings	SOLINEYS
Profile	configuration.
8º Account	
SB Applications	Your SSH keys 🖉 0
Chat	Add an SSH key
 Access Tokens 	Add an SSH key for server access to Gitlab Learn more
🖾 Emails	
Password	Rey
₽ SSH Keys	
₽ GPG Keys	
≅ Preferences	
Comment Templates	Begins with 'ssh-rsa', 'ssh-dss', 'ecdsa-sha2-nistp256', 'ecdsa-sha2-nistp384', 'ecdsa-sha2-nistp521', 'ssh-ed25519', 'ssh-ed3a-sha2-nistp256@openssh.com', or 'sk-ssh- er/7551@onessh-com'
C Active Sessions	
Authentication Log	
	Example: MacBook key
	Kay utans are publicity visitore.
	Usage type
	Authentication & Signing V
	Expiration date
	2025-04-17 🚳 🛍
	Optional but recommended. If set, key becomes invalid on the specified date.
	Add two
3 Help & Admin Area	Add wey Cancer

4. Save the SSH key.

₩		+ 🌬	User Settings / SSH Key	/5						
D	n	ß		Q Search page						
Q Se	earch or go t	to								
User settings	User settings			SSH Keys						
Profile				SSH keys allow you to establish a secure connection between your computer and GitLab. SSH fingerprints verify that the client is connecting to the correct hest. Check the configuration.						, the current insta
8ª Account	nt									
88 Applicat	itions			Your SSH keys 🖉 1						Add new key
💭 Chat				Title	Key	Usage type	Created	Last used	Expires	Actions
Access	Tokens			root@	P	Authentication & Signing	just now	Never	2025-04-18	Develop 19
🖻 Emails										(Revoke)
🔂 Passwo	ord									
	ations									
🖉 SSH Ke	iys									
🖉 GPG Ke	rys									
2 Preferences										
Comment Templates										
Active Sessions										
Authent	tication Log									

----End

4.4.3.2 Configuring Cluster Environments

Prerequisites

Two new VPCs have been created, and their CIDR blocks do not overlap. In this example, the VPC is named **vpc-A** and the CIDR block is 172.16.0.0/16. The other VPC is named **vpc-B** and the CIDR block is 172.17.0.0/16.

Creating a Cluster in the Testing Environment

- **Step 1** Log in to the CCE console and click **Buy Cluster** in the upper right corner on the **Clusters** page.
- **Step 2** Configure the cluster. The following lists some key parameter configurations. You can configure other parameters as required or use their default values. For details, see **Buying a CCE Standard/Turbo Cluster**.
 - Type: CCE Turbo Cluster
 - **Cluster Version**: Select the latest version.
 - **VPC**: Select **vpc-A** with the CIDR block of 172.16.0.0/16.
- **Step 3** Configure other parameters, complete the cluster creation, and wait until the cluster is running.
- **Step 4** In the navigation pane, choose **Nodes**, click the **Nodes** tab, and click **Create Node** in the upper right corner.
- **Step 5** Configure the node. The following lists some key parameter configurations. You can configure other parameters as required or use their default values. For details, see **Creating a Node**.
 - **Specifications**: Select a flavor with 4 vCPUs and 16 GiB of memory.
 - OS: Huawei Cloud EulerOS 2.0
 - EIP: Select **Auto create** and bind an EIP to the node for the node to pull public network images and perform other operations.
- **Step 6** Confirm the specifications and create the node.

One node can meet the basic requirements in this practice.

----End

Creating a Cluster in the Production Environment

- **Step 1** Log in to the CCE console and click **Buy Cluster** in the upper right corner on the **Clusters** page.
- **Step 2** Configure the cluster. The following lists some key parameter configurations. You can configure other parameters as required or use their default values. For details, see **Buying a CCE Standard/Turbo Cluster**.
 - Type: CCE Turbo Cluster
 - **Cluster Version**: Select the latest version.
 - VPC: Select vpc-B with the CIDR block of 172.17.0.0/16.
- **Step 3** Configure other parameters, complete the cluster creation, and wait until the cluster is running.
- **Step 4** In the navigation pane, choose **Nodes**, click the **Nodes** tab, and click **Create Node** in the upper right corner.
- Step 5 Configure the node pool. The following lists some key parameter configurations. You can configure other parameters as required or use their default values. For details, see Creating a Node.
 - Specifications: Select a flavor with 4 vCPUs and 16 GiB of memory.
 - OS: Huawei Cloud EulerOS 2.0
 - EIP: Select **Auto create** and bind an EIP to the node for the node to pull public network images and perform other operations.
- **Step 6** Confirm the specifications and create the node. One node can meet the basic requirements in this practice.

----End

Creating VPC Peering Connections

To enable Jenkins to access the API servers of the testing and production clusters, VPC peering connections need to be created since they are located in different VPCs.

- Step 1 Log in to the VPC console. In the navigation pane, choose Virtual Private Cloud > VPC Peering Connections.
- **Step 2** Enable the network between the Jenkins server and the testing cluster.
 - 1. In the upper right corner, click **Create VPC Peering Connection**. Configure the parameters following instructions.
 - VPC Peering Connection Name: peering-XA
 - Local VPC: vpc-X where the Jenkins server resides
 - **Peer VPC**: **vpc-A** where the testing cluster resides

Figure 4-19 Creating a VPC peering connection

Basic Configuration	
Region	
VPC Peering Connection Name	e peering-XA
Description (Optional)	
	0/255 %
Local VPC Settings	
Local VPC	vpc-X V Q
Local VPC CIDR Block 1	92.168.0.0/16
Peer VPC Settings	
Account	My account Another account
Peer Project	· · · ·
Н	f you select My account, the project is filled in by default.
Peer VPC	vpc-A v
Peer VPC CIDR Block 1	72.16.0.0/16

2. Click Create Now.

3. Click the name of the created VPC peering connection and add routes.

Click **Add Route** and add the CIDR block of vpc-A (172.16.0.0/16) to the route table of vpc-X.

Select **Add a route for the other VPC** and add the CIDR block of vpc-X (192.168.0.0/16) to the route table of vpc-A.

Figure 4-20 Adding a route

VPC Peering Connections	s / peering-XA			×
		Add Route		
^ peering-XA		* VPC	VDC-X	
Name	peering-XA 🖉			
ID	d8100ebb-f0ae-4ffc-9f90-f3216a9a8538	* Route Table	rtb-vpc-X(Default) ~	Q. View Route Table
Local VPC Name	vpc-X	* Destination	172.16.0.0/16 ×	1
Local VPC ID	5e3864c6-e78d-4d00-b6fd-de09d432c632			4
Local VPC CIDR Block	192.168.0.0/16	* Next Hop	peering-XA(d8100ebb-f0ae-4ffc-9f90-f3216a9a8!	
Description	A	Description		
			0.075	
Associated Rout	PS		0/255 %	
		Add a route for	the other VPC	
Add Route	few route configuration examples of a VPC peering connection	To enable communi forward and return r	ications between VPCs connected by a VPC peering co routes to the route tables of the VPCs.Learn more	nnection, you need to add
Q Select a proper	ty or enter a keyword.	* VPC	VDC-A	
Destination	VPC N	ext Hop Type		
		* Route Table	rtb-vpc-A(Default) ~	Q View Route Table
		* Destination	192.168.0.0/16 ×	
		* Next Hop	peering-XA(d8100ebb-f0ae-4ffc-9f90-f3216a9a8!	
		Description		
			0/255 //	
				Cancel

- 4. Click OK.
- **Step 3** Repeat the preceding steps to enable the networking between the Jenkins server and the production cluster.
 - 1. In the upper right corner, click **Create VPC Peering Connection**. Configure the parameters following instructions.
 - VPC Peering Connection Name: peering-XB
 - Local VPC: vpc-X where the Jenkins server belongs
 - **Peer VPC**: **vpc-B** where the testing cluster resides
 - 2. Click **OK**.
 - 3. Click the name of the created VPC peering connection and add routes.

Click **Add Route** and add the CIDR block of vpc-B (172.17.0.0/16) to the route table of vpc-X.

Select **Add a route for the other VPC** and add the CIDR block of vpc-X (192.168.0.0/16) to the route table of vpc-B.

4. Click OK.

----End

Interconnecting with SWR

This solution uses SoftWare Repository for Container (SWR) to store container images built using Jenkins.

- **Step 1** Log in to the SWR console.
- **Step 2** Create an organization to manage images. For details, see **Organization Management**.
- **Step 3** Obtain the long-term valid **docker login** command. For details, see **Obtaining a Long-Term Valid Login Command**.

This login command will be used when images are uploaded through pipelines. For details, see **Implementing Continuous Integration and Deployment**.

----End

4.4.3.3 Configuring a GitLab Project

- Step 1 Obtain the source code and save it to the local. A Java example is used.
- Step 2 Create the ccedemo group on GitLab.

	Create group
(+)	Groups allow you to manage and collaborate across multiple projects. Members of a group have access to all of its projects.
	Groups can also be nested by creating subgroups.
Group namo	
ccedemo	
Must start with letter,	digit, emoji, or underscore. Can also contain periods, dashes, spaces, and parentheses.
A Your group na	me must not contain a period if you intend to use SCIM integration, as it can lead to errors.
Group URL	
http://60.204.169.3	0/ ccedemo
Checking group URL	availability
Visibility level	
Who will be able to s	ee this group? View the documentation
Private The group and its	projects can only be viewed by members.
$\odot \Phi$ Internal	
The group and an	y internal projects can be viewed by any logged in user except external users.
O Public The group and an	w sublic prelate can be viewed without any authentication

Step 3 Add the java-demo project to the ccedemo group.

🤟 🗆 + ֎	Your work / Projects / New project / Create blank project
C Search or go to	Create blank project Create a blank project to store your files, plan your work, and collaborate on code, among other things.
Projects	Project name
% Groups	java-demo
D' Issues	uturi dust start with a lowercase or uppercase letter, digit, emoji, or underscore. Can also contain dots, pluses, dashes, or spaces.
\$3 Merge requests >	Project URL Project stug
🖂 To-Do List	http://
③ Milestones	Visibility Level 🕜
X Snippets 3 Activity	
	Project Configuration
	Initialize repository with a README Allows you to immediately clone this project's repository. Skip this if you plan to push up an existing repository.
	Enable Static Application Security Testing (SAST) Analyze your source code for known security vulnerabilities. Learn more.
	Create project Cancel

Step 4 Upload the project code to the local GitLab repository.

```
cd ~/java-demo-main //Change the directory address as needed.
git init
git remote add origin http://**.**.**/ccedemo/java-demo.git // Project URL of java-demo in step 3
git config --global user.name "Administrator"
git config --global user.email "admin@example.com"
git add .
git commit -m "Initial commit"
git push -u origin main
```

Enter the user name **root** and its password. (If the default password is not changed, obtain it by referring to **Step 6**.)

----End

4.4.3.4 Implementing Continuous Integration and Deployment

To implement this solution, Jenkins must be triggered for compilation and packaging via a code push event. Once the request is approved via email, the application can be deployed in a Kubernetes cluster.

Installing Jenkins Plugins

In addition to the default plugins installed during Jenkins installation, you also need to install the GitLab, Kubernetes CLI, and Email Extension Template plugins. For details, see the Jenkins official documentation.

On the Jenkins dashboard page, click **Manage Jenkins** on the left and choose **System Configuration** > **Manage Plugins**. On the **Available** tab, search for **GitLab**, **Kubernetes CLI**, and **Email Extension Template**, and install them.

Datified ** Minege Media ** ** Miges Plugins Q. Exclassion Q. Exclasin Q. Exclassion <	🏘 Jenkins			Q Search (CTRL+K)	0 0 0 1	🕐 admin 🗸	⊖ log out
Plugins Q. text textme C (if text) C (if text) <thc (if="" text)<="" th=""> <thc (if="" text)<="" th=""></thc></thc>	Dashboard > Manage Jenkins > Plugins						
itedates instal Kane i interference interference interference Image: Second Seco	Plugins	Q, Ema	I Extension			👍 Install	ى 🗸
▲ Available plagins 0: Gatab 1.8.0 Built Higger Bu	Updates	Install	Name i		Released		
Image: Advanced settings Image: Kubernets CU 1121 Lubernets 7 mo 21 days ago Image: Kubernets CU 1121 Lubernets Compare kubert for Kubernets 7 mo 21 days ago Image: Kubernets CU 1121 Lubernets Email Edemains Template 1.5 Luber Kutefun; Image: The plaquin Alows a minimistants to create global templates for the Extended Email Publisher: The plaquin is up for adaption (W are looking for new maintainers. Wild our <u>Adopt a Plaquin Instante</u> for more information. 1 yr 6 mo ago	Available plugins Installed plugins		GitLab 18.0 Build Higgers This plugin allows GitLab to trigger Jenkins builds and display their results in the GitLab UI.		3 mo 16 days ago		
Insult Interview Invaluet: 1 yr 6 mo ago This plugin is up for adoption! We are looking for new maintainers. Visit our <u>Adopt a Proprin</u> initiative for more information. 1 yr 6 mo ago	Advanced settings		Kubernetes CU 1.12.1 Kubernetes Configure kubert! for Kubernetes		7 mo 21 days ago		
			Email Environment for particular 1.5 Relativistics: evaluati: The Juguin allows addiministators to create global templates for the Extended Email Publishee. This plaugin is up for adoption! We are looking for new maintainers. Visit our <u>Adopt a Plaugin</u> initiative for more information.		1 yr 6 mo ago		

The versions of the preceding plugins may change over time.

Configuring GitLab Hooks

Once you push your code, GitLab will notify Jenkins of the event using webhooks. To ensure this process runs smoothly, you must first configure GitLab hooks.

Step 1 Log in to Jenkins, click **New Item**, and create a pipeline.

🍨 Jenkins		Q Search (CTRL+K)	û 🗘 🛈 🚺 🏵 admin ∽ 🕞 log out
Dashboard > All >			
	Enter an item name		
	jøræ-demo » Required field		
	Freestyle project Classic, general-purpose job type that checks out from up to one SCM, executes build steps serially, followed b steps like archiving artifacts and sending email notifications.	ıy post-build	
	Pipeline Pipeline Pipeline Solidate solid agents. Suitable for building pipelines (forme workflows) and/or organizing complex activities that do not easily fit in free-style job type.	rly known as	
	Wulti-configuration project Suitable for projects that need a large number of different configurations, such as testing on multiple environm specific builds, etc.	nents, platform-	
	Organization Folder Creates a set of multibranch project subfolders by scanning for repositories.		
	Wultibranch Pipeline Creates a set of Pipeline projects according to detected branches in one SCM repository.		
	Folder Creates a container that stores nested items in it. Useful for grouping things together. Unlike view, which is just creates a separate namespace, so you can have multiple things of the same name as long as they are in differe	t a filter, a folder nt folders.	
	OK		

Step 2 Copy the URL and click **Advanced**.

Configure	Build Triggers
(h) count	Build after other projects are built ?
203 General	Build periodically ?
Advanced Project Options	Build when a change is pushed to GitLab. GitLab webhook URL: http://
Ppeline لام	Enabled Gittab triggers Push Events ? Push Events in case of branch delete ? Goneral Merge Request Events ? Build only if new commits were pushed to Merge Request ? Accepted Merge Request Events ? Gosed Merge Request Events ?
	Rebuild open Merge Requests Never
	Comment (reger) for triggering a build Comment (reger) for triggering a build Advanced Advanced G Enable (ci-skip) G (gover WIP Morge Requests C (gover WIP

Step 3 Click **Generate** to generate a token, record it, retain the default values for other parameters, and save the changes.

Configure	Advanced A
 General Advanced Project Options 	Enable (ci-skip) Toron (Circle) Ignore WIP Merge Requests
رجا Pipeline	Labels that launch a build if they are added (comma-separated) ?
	Set build description to build cause (eg. Merge request or Git Push) Build on successful pipeline events Duble on succe
	Pedding build name for pipeline (2)
	Cancel pending merge request builds on update Allowed branches Allowed branches to trigger this job
	Filter branches by name (?)
	Truet transfers by region (2) Filter merge request by label
	Secret token (?)
	Generate

Step 4 Log in to GitLab and enable webhooks.

Allow requests to local network. To enhance security, GitLab 10.6 and later versions prohibit webhook requests from being sent to the local network.

₩		+ 🚵	Admin Area / Network		
D I	3		Git SSH operations rate limit	Expand	
Q Search	or go to.		Limit the number of Git operations a user can perform per minute, per repository. Learn more.		
Admin Area					
81 Overview		>	Outbound requests	Collapse	
🥩 CI/CD		>	Allow requests to the local network from hocks and integrations. Learn more. Block all requests except for IP addresses. IP rankanian names defined in the allowlist		
h Analytics		>	O Allow requests to the local network from webhooks and integrations		
Monitoring		>	Allow requests to the local network from system hooks		
ଟ୍ସ Messages			Local IP addresses and domain names that hooks and integrations can access		
ພໍ System Hook	ks		example.com, 192.168.1.1, xnittab-j1a.com		
88 Applications					
Abuse Report	rts	0			
Deploy Keys					
Zabels			Requests can be made to these IP addresses and domains even when local requests are not allowed. IP ranges such as 1:8:8:10:80:8:0:8:0:8:124 and 127.0.8.9/28 are supported. Domain wildcards are not supported. To separate entries, use commas, semicolons, or newlines. The allowiist can have a maximum of 1000 entries. Domains		
Settings		~	must be IDNA-encoded. Learn more.		
General			Enforce DNS-rebinding attack protection Besolve IP addresses for outbound requests to prevent DNS-rebinding attacks.		
Integrations			Construction of the second of the second conference of the second of the		
Repository			Save changes		
CI/CD					
Reporting			Protected paths	Expand	
Metrics and p	profiling		Rate limit access to specified paths. Learn more.		
Network			Income Bade Linelin		
Appearance			Issues Rate Limits	Expand	
Preterences			Limit dia nomber of issues and abits ber ministe a user can create driough web and API requests. Learn more.		
Quint	0- Hui	init vited)	Notes rate unit	Expand	
U		+ 🚵	ccedemo / jive-demo / Webhook Settings		
e n	<i>.</i>	+ 🔊	ccodemo / java-demo / Webhook Settings		
D n Q Search o	D a or go to	+ 🔊	ccedema / jirei-demo / Webhook Settings Q, Sarrch page		
Q. Search o	C or go to	+	ccedeme / jiva-demo / Wethnesk Settings Q. Search page Webhooks		
C Search o Project J java-demo	or go to	+ 🔊	cooleme / jive-deno / Webhook Settings Q. Search page Webhooks Webhooks Webhooks enable you to send notifications to web applications in response to events in a group or project. We recommend using an integration in preference to a web	100K.	
♥ 『♪ Q. Search o Project J java-demo ☆ Pinned	or go to	+ 🗞	ccedeme / jive-demo / Webhook Settings Q. Search page Webhooks Webhooks Webhooks Webhooks Webhooks enable you to send notifications to web applications in response to events in a group or project. We recommend using an integration in preference to a web Project Hooks 🖞 0	hook.	
Q. Search o Project J. Java-demo \$2 Pinned Issues	or go to	+ 28	ccedeme / jire-demo / Webhook Settings	nook.	
C Search of Q Search of Project J Java-demo Ø Pinned Issues Merge request	or go to	+	ccedeme / jine-demo / Webhooks Settings Q. Search page Webhooks Webhooks enable you to send notifications to web applications in response to events in a group or project. We recommend using an integration in preference to a web Project Hooks it 0 No webhooks enabled. Select trigger events above.	nook. Add new webhook	
Q Search of Q Search of Project J Java-demo XP Pinned Issues Merge reques & Manage	or go to	+	ccedeme / jive-demo / Webhooks Settings Q. Search page Webhooks Webhooks enable you to send notifications to web applications in response to events in a group or project. We recommend using an integration in preference to a web Project Hooks	nook. Add new webhook	
□ 13 Q. Search o Project J. Java-demo x? Pinned Issues Merge request & & & Manage ⇒ > > >	or go to		ccedeme / java-demo / Wethnek Settings	nook.	
IP III Q. Search o Project J. Java-demo xP Pinned Issues Merge request & Manage ※ Plan	or go to	+ 20 2 0 0 0 0	ccedeme / jiva-demo / Webheek Settings	nook. Juda new webhook	
Project ↓ Search of Project ↓ Java-demo ☆ Pinned Issues Merge request ☆ Plan ↔ Code ☞ ₽ Build	or go to		ccedeme / jine-demo / Webhooks Settings Q. Search page Webhooks Webhooks enable you to send notifications to web applications in response to events in a group or project. We recommend using an integration in preference to a web Project Heoks J. 0 No webhooks enabled. Select trigger events above.	iook. Ad new webhook	
Dr 11 C, Search of Project J Java-demo Java-demo Java-demo Merge reques Merge reques Manage Plan Code De Secure De Secure	or go to	+	coademe / jive-demo / Webhooks Settings Q. Search page Webhooks Webhooks enable you to send notifications to web applications in response to events in a group or project. We recommend using an integration in preference to a web Project Hooks No webhooks enabled. Select trigger events above.	nook.	
P 111 Q. Search of Project 1 Java-demo - //>/ Pinned Issues Merge reques & Manage & Plan -/>/ Constraints & Build Q. Secure Deploy	or go to	+ 2 0 0 0 0	codema / java-dema / Webhaoks Settings C Occurrent page Webhaoks Webhaoks enable you to send notifications to web applications in response to events in a group or project. We recommend using an integration in preference to a web Project Hooks _0_0 No webhaoks enabled. Select trigger events above.	hook. Add new webhook	
C. Search of Project Project Project I java-demo K Plinned Issues Merge reques Merge reques Plan Code Plan Code Deploy Deploy Operate	L 1 sts		ccedeme / java-demo / Wethnesk Settings Search page Wethnooks Wethnooks enables you to send notifications to web applications in response to events in a group or project. We recommend using an integration in preference to a web Project Hooks _f_0 No webhooks enabled. Select trigger events above.	nook. Add new webhook	
Project J Java-demo Arrow Marga request Marga request Anarga Anarga Baulid Dian Diantin	L J Ssts		ccedema / jire-demo / Webhook Settings Q. Search page Webhooks Webhooks exable you to used notifications to web applications in response to events in a group or project. We recommend using an integration in preference to a web Project Heoks J.0 No webhooks enabled. Select trigger events above.	iook. Ad new webhook	
C. Search of C. S	L 3 Constant		codeme / jine-demo / Webhaok Settings C. Search page Webhaoks enable you to send notifications to web applications in response to events in a group or project. We recommend using an integration in preference to a web Project Hooks No webhaoks enabled. Select trigger events above.	nook.	
C Search of C Sea	L 3 sts		codema / java-dema / Webhaoks Sentings C Observich page Webhaoks enable you to send notifications to web applications in response to events in a group or project. We recommend using an integration in preference to a web Project Hooks _t_0 No webhaoks enabled. Select trigger events above.	nook.	
C. Search of Project J Java-demo JA Project J Java-demo JA Project Sausa P Ploned Sausa P Ploned P Ploned P Plone P P	1) or go to		codema / jier-dema / Webhook Seetaigs C: Search page Webhooks Webhooks enable you to send notifications to web applications in response to events in a group or project. We recommend using an integration in preference to a web Project Hocks d: 0 No webhooks enabled. Select trigger events above.	nook. Add new webhook	
C Search or Project J Java Search or Project J Java Search or Project Sauses Manage Sauses Manage Planed P	1) or go to	+ 200 C C C C C C C C C C C C C	codema / jare-tema / Webhook Settings	nook.	
C. Search of C. Search of Project J. Java-demo % Pindet Issues Merge reques % Manage % Pind	L L L L L L L L L L L L L L L L L L L	+ 200 	codeme / jend-teme / Wethholds Settings C	hook.	
C. Search of C. S	sts		codema / jan-dema / Webhaoks Sentigs Character page Webhaoks enable you to send notifications to web applications in response to events in a group or project. We recommend using an integration in preference to a veb Project Hooks No webhaoks enabled. Select trigger events above.	nook.	
C. Search of C. S	sts		Codema / jest-dema / Webhook Seetalies Codema / jest-dema / Webhook Total Codema / Search page Project Hocks & to to send notifications to web applications in response to events in a group or project. We recommend using an integration in preference to a web Project Hocks & to No webhooks enabled. Select trigger events above.	hook.	
C. Search of C. S	sts		Containing () Workhooks Seetings Containing () Search page Project Hooks () O Project Hooks () O No webhooks enabled: Select trigger events above.	nook.	

Step 5 Log in to the **java-demo** project on GitLab, choose **Settings** > **Webhooks**, and enter the values recorded in Jenkins in **URL** and **Secret token**.

₩		0 -	- 🚵	ccedemo / java-demo / Webhook Settings / Webhook	
	o n		ß		
				Q Search page	
	Q Search o	or go to	0	Webhook	
				Webhooks enable you to send notifications to web applications in response to events in a group or project. We recommend using an integration in preference to a webhook.	
රිපි	Manage		>	URL	
Ū	Plan		>	http://192.168.0.172.8000/project/java-demo	
	Code		2	URL must be percent-encoded if it contains one or more special characters.	
B	Build		\rightarrow	Show full URL	
Φ	Secure		->	Mask portions of URL	
	Dealers			Do not show sensitive data such as tokens in the UI.	
01	Deptoy			Name (optional)	
6	Operate		->		
	Monitor		>	Description (optional)	
<u> _1}1</u>	Analyze		\rightarrow		
ŵ	Settings				
0					
	General		- 1	Secret token	
	Integrations		- 1		
	Webhooks				
	Access Token	IS	- 1	Used to validate received payloads. Sent with the request in the <u>X-Sitlab-Token</u> HTTP header.	
	Deperitory		- 1	Trigger	
	Repository		- 1	Push events	
	Merge reques	ts	- 1	All branches	
	CI/CD		- 1	Wildcard pattern	
	Packages and	l registr	- 1	Regular expression	
	Monitor		- 1	C Tag push events	
	Usage Quotas			A new tag is pushed to the repository.	
	Anotes	,	÷	Comments	
61	Help	@ Admi	n Area	A comment is added to an issue or merge request.	

Step 6 At the bottom of the page, confirm that the webhook is added and perform tests.

Hook executed successfully: HTTP 200	×
Q Search page	
Webhooks Webhooks enable you to send notifications to web applications in response to events in a group or project. We recommend using a	n integration in preference to a webhook.
Project Hooks ئن 1	Add new webhook
http://192.188.0.49:8000/project/java-demo Push events Tag push events SSL Verification: enabled	Test V Edit Delete Push events Issues events Confidential issues events Comments

----End

Configuring Email Notifications

Jenkins often uses email for message notifications and approvals. This solution also employs email as the approval method.

Step 1 On the **Manage Jenkins** page, select **System**.

Dashboard > Mana	ge Jenkins > System >
	Jenkins Location
	Jenkins URL (?)
	http://line 8000/
	System Admin e-mail address 👔
	@ IG3com

Step 2 Configure basic mailbox information and enter the email address of the administrator.

ashboard > Manage Jenkins > System >	
E-mail Notification	
SMTP server	
smtp.163.com	
Default user e-mail suffix 🔞	
(@161.com	
erescen.	
Advanced A 🖉 Edited	
Use SMIP Authentication (2)	
Uker Name	
@163.com	
A Concernitional and the first state of the second state of the STC of CO	
For second when using autoentication is is recommended to enable ender it is or SSC.	
Password	
🕑 Use SSL 👔	
Use TLS	
SMTP Port ?	
400	
Tesh Teshaw	
Reply- to Audress	
Charoot	
Sive Apply	

NOTICE

The password is not the email password but the email authorization code.

----End

Configuring a Credential

Accessing Kubernetes clusters through HTTPS is crucial for maintaining security. To achieve this, you must configure the credentials for accessing the clusters on Jenkins.

- **Step 1** Obtain the kubeconfig configuration of the Kubernetes cluster client. For details, see **Connecting to a Cluster Using kubectl**.
- **Step 2** Log in to Jenkins, click **Manage Jenkins**, choose **Manage Credentials** under **Security**, and create a cluster credential using the secret file.

ł	
cret file	
Scope ?	
Global (Jenkins, r	odes, items, all child items, etc)
File	
🗘 Choose File	No file chosen
ID ?	
test_config	
Description ?	

New credentials

Step 3 Repeat the preceding steps to create an access credential for the production cluster.

----End

Create

Writing Pipeline Scripts

Pipeline is a workflow framework that operates within Jenkins. It links tasks that would typically run independently on one or more nodes, allowing for the orchestration and visualization of complex processes that cannot be completed by a single task. As the primary feature of Jenkins 2.*X*, it enables Jenkins to transition from CI to CD and DevOps. Consequently, the writing of pipeline scripts is crucial to the successful implementation of the entire solution.

The following shows the concepts of pipeline scripts:

Node

A node is a machine which is part of the Jenkins environment and is capable of executing a pipeline.

Stage

A stage block defines a group of specific tasks to be executed in different stages such as the build, test, and deploy stages through the entire pipeline.

• Step

A step is a single task in a stage, such as running a test or deploying code. It tells Jenkins what to do at a specific time, for example, to execute the **shell** command. (For more Jenkins pipeline syntax, see the **Jenkins official documentation**.)

- **Step 1** Go to the Jenkins pipeline and click **Configuration** in the navigation pane.
- **Step 2** Configure the pipeline scripts. The following pipeline scripts are for reference only. You can customize the scripts based on your service requirements.

Some parameters in the example need to be changed:

- **git_url**: specifies the SSH address of the code repository in GitLab. You need to replace it with the actual value.
- **swr_login**: The login command is the command obtained in **Step 3**.
- **swr_region**: specifies the region of SWR. You need to specify the region as needed.
- **organization**: specifies the actual organization name in SWR.
- **build_name**: specifies the name of the created image.
- **credential**: specifies the testing cluster credential added to Jenkins. You need to enter the credential ID. To deploy the service in another cluster, add the access credential of the cluster to Jenkins again. For details, see **cluster access credential configurations**.
- prod_credential: specifies the production cluster credential added to Jenkins. You need to enter the credential ID. To deploy the service in another cluster, add the access credential of the cluster to Jenkins again. For details, see cluster access credential configurations.
- test_apiserver: specifies the API server address of the testing cluster. For details, see Connecting to a Cluster Using kubectl. You have to ensure that the address can be accessed from the Jenkins cluster.
- prod_apiserver: specifies the API server address of the production cluster. For details, see Connecting to a Cluster Using kubectl. You have to ensure that the address can be accessed from the Jenkins cluster.
- **test_email**: specifies the email address of the test personnel.
- **admin_email**: specifies the email address of the approver.

```
#!groovy
//Define the code repository address.
def git_url = 'ssh://git@xxxx:222/ccedemo/java-demo.git'
//Define the SWR login command.
def swr_login = 'docker login -u cn-north-4@xxxx -p xxxxx swr.cn-north-4.myhuaweicloud.com'
//Define the SWR region.
def swr_region = 'cn-north-4'
// Specify the name of an SWR organization to which images are pushed.
def organization = 'testapp'
//Define the image name.
```

```
def build_name = 'demo01'
// Certificate ID of the testing cluster
def test_credential = 'test_config'
// Certificate ID of the production cluster
def prod_credential = 'prod_config'
// API server address of the testing cluster. You have to ensure that the address can be accessed from the
Jenkins cluster.
def test_apiserver = 'https://xxx:5443'
// API server address of the production cluster. You have to ensure that the address can be accessed from
the Jenkins cluster.
def prod_apiserver = 'https://xxxx:5443'
// Email addresses
def test email="xxxxx@xx.com"
def admin_email="xxxx@xx.com"
pipeline{
agent any
stages{
  stage('Git code'){
     steps{
        echo "1. Git code"
        git url: git_url
        script {
           // Specify the return value of git rev-parse --short HEAD as the commit ID, which is then used
as the image tag.
           build_tag = sh(returnStdout: true, script: 'git rev-parse --short HEAD').trim()
           image_url = "swr.${swr_region}.myhuaweicloud.com/${organization}/${build_name}:$
{build_tag}"
           }
        }
  }
  stage('Build') {
     steps{
        echo "2. Build Docker Image Stage and Push Image"
        sh "docker build -t ${image_url}."
        sh swr_login
        sh "docker push ${image_url}"
        // Replace the image URL with that in the Kubernetes resource file.
        sh "sed -i 's+demo01:v1+${image_url}+g' ./demo01.yaml"
     }
  }
  stage('Deploy Test Environment') {
     steps{
        // Configure the testing environment certificate.
        echo "3. Deploy Test Enviroment"
        script {
           try {
             withKubeConfig([credentialsId: test_credential, serverUrl: test_apiserver]) {
                sh 'kubectl apply -f ./demo01.yaml'
                // The YAML file is stored in the code repository. It is only used as an example here, so you
need to replace it as required.
             }
             println "deploy success"
             // Send an email to the test personnel.
             mail subject: "[Please Test] The application has been deployed in the test environment. Please
start the test."
             body: """After the test is passed, <a href="${BUILD_URL}input">click the link and use the
account to log in to the system. The test is successful.</h3>""",
             charset: 'utf-8',
             mimeType: 'text/html',
             to: "$test_email"
           } catch (e) {
             RUN FLAG = false
             println "deploy failed!"
             println e
          }
       }
    }
  }
```

```
stage('Test'){
     // Wait for the test personnel to make confirmation.
     input{
        message "Test Passed or Not"
        submitter "admin"
     }
     steps{
        script{
           println "4. Test Passed"
           // Send an email to the administrator.
           mail subject: "[Please Approve] Release to the production environment",
           body: """The test is passed, <a href="${BUILD_URL}input">click the link and use the account to
log in to the system for approval.</h3>""",
           charset: 'utf-8',
           mimeType: 'text/html',
           to: "$test_email"
        }
     }
  }
  stage('Approve'){
     input{
        message "Release to Production Environment or Not"
        submitter "admin"
     }
     steps{
        script{
           println "5. Approved and release it to the production environment."
        }
     }
  }
  stage('Deploy Produce Environment'){
     steps{
        echo "6. Deploy Produce Enviroment"
        script {
           try {
              withKubeConfig([credentialsId: prod_credential, serverUrl: prod_apiserver]) {
                sh 'kubectl apply -f ./demo01.yaml'
                // The YAML file is stored in the code repository. It is only used as an example here, so you
need to replace it as required.
              println "deploy success"
           } catch (e) {
             println "deploy failed!"
              println e
          }
       }
     }
  }
3
```



----End

Configuring Continuous Build and Deployment

- Step 1 Modify the local code and submit it to trigger compilation. git add . && git commit -m "add template" && git push
- Step 2 Go back to the Jenkins page.

You can see that the project has automatically triggered compilation and building.

Pipeline							Ż
	Start	Git code	Build	Deploy Test En	Test	End	

Step 3 Wait a few minutes for the email notification confirming the test.

Please lestj line	application ha	s been deployed	n the test environn	nent. Please start the t	est. UPC
rom:	@163.com> +				
To:)163.com> +				
Date: 2024-05-29 11:	45				

Step 4 Log in to the testing cluster and verify that the **java-demo** workload has been created.

Deployme	nts StatefulSets	DaemonSets	Jobs Cron Jobs	Pods				Quick Links	Create
Del	ete Export ~								
	elect a property or enter a key	word.							
0	Workload Name		Status	Pods (Normal/All) \ominus	CCI elastic policy \ominus	Namespace 🕀	Created \ominus	Image Name \ominus	
	java-demo		O Running	3/3	Disable scheduling	default	12 minutes ago	demo01:bb71aee	

Step 5 Click the link in the email to confirm the test. (In this example, the upgrade test is considered to pass.)





K Back Reply	Reply to all 🗸	Forward v Delete	e Report	Reject	Mark as 🗸	Move to \vee	More v
[Please Approve]	Release to the	e production envir	onment	090	🖶 🖯 🕀 Safe	browsing ~	
From:	@163.com>	+					
То:	@163.com> +						
Date: 2024-05-29 11:4	19						
10000							

The test is passed, Click the link and use the account to log in to the system for approval.

Step 7 (For the approver) Determine that the service can be deployed in the production environment, click the link, and agree to the request.

	🧌 Jenkins							
Dash	Dashboard > java-demo > #28 > Paused for Input							
F	Status	Release to Production Environment or Not						
	Changes	Proceed Abort						
>	Console Output							
6	View as plain text							
ď	Edit Build Information							
Ũ	Polling Log							
٢	Timings							
-								

Step 8 Access the production cluster console and verify that the java-demo workload has been created and released in the production environment.

Deployment	s StatefulSets	DaemonSets	Jobs	Cron Jobs	Pods				Quick Links	Create
Delet	e Export ~									
Q Sel	ect a property or enter a ke	yword.								
0	Workload Name		Status		Pods (Normal/All) \Leftrightarrow	CCI elastic policy ⊖	Namespace 😂	Created 🔶	Image Name	
	java-demo		O Running		3/3	Disable scheduling	default	12 minutes ago	🗼 demo01:bb71aee	

----End

5 Disaster Recovery

5.1 Recommended Configurations for HA CCE Clusters

ltem	Description	Recommended Operations
Master node	CCE is a hosted Kubernetes cluster service. You do not need to perform O&M on the master nodes. You can configure your cluster specifications to improve the stability and reliability.	 Deploying the Master Nodes in Different AZs Selecting a Network Model Selecting a Service Forwarding Mode Configuring Quotas and Limits for the Cloud Service Resources and Resources in a Cluster Monitoring Metrics of the Master Nodes
Worker node	In a Kubernetes cluster, the data plane consists of worker nodes that can run containerized applications and transmit network traffic. When using CCE, perform O&M on worker nodes by yourself. To achieve HA, ensure the worker nodes' scalability and repairability and pay attention to the running statuses of the worker nodes' key components.	 Partitioning Data Disks Attached to a Node Running npd Configuring the DNS Cache Properly Deploying CoreDNS

This section describes the recommended configurations for a Kubernetes cluster in which applications can run stably and reliably.

ltem	Description	Recommended Operations
Applicatio n	If you want your applications to be always available, especially during peak hours, run them in a scalable and elastic manner and pay attention to their running statuses.	 Running Multiple Pods Configuring Resource Quotas for a Workload Deploying an Application in Multiple AZs Deploying the System Add-ons in Multiple AZs Configuring Auto Scaling Viewing Logs, Monitoring Metrics, and Adding Alarm Rules

Deploying the Master Nodes in Different AZs

Multiple regions are provided for you to deploy your services, and there are different availability zones (AZs) in each region. An AZ is a collection of one or more physical data centers with independent cooling, fire extinguishing, moisture-proof, and electricity facilities in each AZ. AZs within a region are connected using high-speed optical fibers. This allows you to build cross-AZ HA systems.

When creating a cluster, enable the HA mode of the cluster and configure the distribution mode of the master nodes. The master nodes are randomly deployed in different AZs. This ensures a higher disaster recovery (DR) capability of the cluster.

You can also customize the distribution mode. The following two modes are supported:

- Random: Master nodes are deployed in different AZs for DR.
- **Custom**: Master nodes are deployed in specific AZs.
 - Host: Master nodes are deployed on different hosts in the same AZ.
 - **Custom**: Master nodes are deployed in the AZ you specify.

Figure 5-1 Configuring an HA cluster

Cluster Scale	Nodes: 50	Nodes: 200	Nodes: 1000	Nodes: 2000	
	Select the maximum number	r of nodes that can be ma	naged by the cluster. Mas	ter node specifications are	determined based on the cluster scale. After the cluster is created, node specifications can be modified, but nodes can only be upgraded for higher specifications.
	Note: Generally, the cluster i advised to perform security t	may not reach the maxim hardening by referring to t	um number of nodes of the he best practices 🕜 for p	e selected scale. The actua preventing cluster overload	al scale is related to multiple factors, such as the type, quantity, and size of resource objects in the cluster, and external access to the cluster liable-apiserver. To prevent the cluster from being over 1
Master Nodes 🔏	3 Masters	Single			
	CCE will create three master	r nodes for your cluster to	ensure cluster reliability.	The nodes are automatical	ly hosted by CCE and deployed with Kubernetes cluster management components such as kube-apiserver, kube-controller-manager, and kube-scheduler.

Selecting a Network Model

- Network model: CCE supports VPC, Cloud Native 2.0, and container tunnel network models. Different models have different performance and functions. For details, see Network Models.
- VPC network: To enable your applications to access other cloud services like RDS, create related services in the same VPC network as your cluster. This is because services using different VPC networks are isolated from each other. If you have created instances, use VPC peering connections to enable communications between VPCs.
- Container CIDR block: Do not configure a small container CIDR block. Otherwise, the number of supported nodes will be limited.
 - For a cluster using a VPC network, if the subnet mask of the container CIDR block is /16, there are 256 x 256 IP addresses available. If the maximum number of pods reserved on each node is 128, the maximum number of nodes supported is 512.
 - For a cluster using a container tunnel network, if the subnet mask of the container CIDR block is /16, there are 256 x 256 IP addresses assigned to your cluster. The container CIDR block allocates 16 IP addresses to the nodes at a time by default. The maximum number of nodes supported by your cluster is 4096 (65536/16=4096).
 - For a cluster using a Cloud Native 2.0 network, the container CIDR block is the VPC subnet, and the number of containers can be created depends on the size of the selected subnet.
- Service CIDR block: The service CIDR block determines the upper limit of Service resources in your cluster. Evaluate your actual needs and then configure the CIDR block. A created CIDR block cannot be modified. Do not configure an excessively small one.

For details, see **Planning CIDR Blocks for a Cluster**.

Selecting a Service Forwarding Mode

kube-proxy is a key component of a Kubernetes cluster. It is responsible for load balancing and forwarding between a Service and its backend pod. When using clusters, consider the potential performance problems of the forwarding mode.

CCE supports the iptables and IPVS forwarding modes.

- IPVS allows higher throughput and faster forwarding. It applies to scenarios where the cluster scale is large or the number of Services is large.
- iptables is the traditional kube-proxy mode. This mode applies to the scenario where the number of Services is small or there are a large number of short concurrent connections on the client. When there are more than 1000 Services in the cluster, network delay may occur.

Configuring Quotas and Limits for the Cloud Service Resources and Resources in a Cluster

CCE allows you to configure resource quotas and limits for your cloud service resources and resources in your clusters. This prevents excessive use of resources. When creating your applications for CCE clusters, consider these limits and

periodically review them. This will avoid scaling failures caused by insufficient quotas during application running.

- Configuring resource quotas for cloud services: Cloud services like ECS, EVS, VPC, ELB, and SWR are also used to run the CCE clusters. If the existing resource quotas cannot meet your requirements, submit a service ticket to increase the quotas.
- Configuring resource quotas for a cluster: You are allowed to configure the namespace-level resource quotas to limit the number of objects of a certain type created in a namespace and the total computing resources like CPU and memory consumed by the objects. For details, see Configuring Resource Quotas.

Monitoring Metrics of the Master Nodes

Monitoring metrics of the master nodes allows you to check the master nodes' performance and efficiently identify problems occurred on them. The master nodes which are not running properly may lower application reliability.

CCE can monitor kube-apiserver, kube-controller, kube-scheduler, and etcd-server on master nodes. You need to install the **Cloud Native Cluster Monitoring** addon. With the Grafana add-on, you can use the **Kubernetes monitoring overview dashboard** to monitor metrics of Kubernetes API server requests and latency and etcd latency.

If an on-premises Prometheus instance is used, you can manually add monitoring metrics. For details, see **Monitoring Metrics of Master Node Components Using Prometheus**.

Partitioning Data Disks Attached to a Node

By default, the first data disk of a worker node is for storing the container runtime and kubelet components. The remaining capacity of this data disk affects image download and container startup and running. For details, see **Space Allocation of a Data Disk**.

The default space of this date disk is 100 GiB. You can adjust the space as required. Images, system logs, and application logs are stored on data disks. Therefore, you need to evaluate the number of pods to be deployed on each node, the size of logs, images, and temporary data of each pod, as well as some reserved space for the system. For details, see **Selecting a Data Disk for the Node**.

Running npd

A failure in a worker node may affect the availability of the applications. **CCE Node Problem Detector** is used to monitor node exceptions. It helps you detect and handle latent exceptions in a timely manner. You can also customize the check items, including target node, check period, and triggering threshold. For details, see **Configuring Node Fault Detection Policies**.

Configuring the DNS Cache

When the number of DNS requests in a cluster increases, the load of CoreDNS increases and the following issues may occur:
- Increased delay: CoreDNS needs to process more requests, which may slow down the DNS query and affect service performance.
- Increased resource usage: To ensure DNS performance, CoreDNS requires higher specifications.

To minimize the impact of DNS delay, deploy NodeLocal DNSCache in the cluster to improve the networking stability and performance. NodeLocal DNSCache runs a DNS cache proxy on cluster nodes. All pods with DNS configurations use the DNS cache proxy running on nodes instead of the CoreDNS service for domain name resolution. This reduces CoreDNS' load and improves the cluster DNS performance.

You can install the **NodeLocal DNSCache** add-on. For details, see **Using NodeLocal DNSCache to Improve DNS Performance**.

Properly Deploying CoreDNS

Deploy the CoreDNS instances in different AZs and nodes to mitigate the singlenode or single-AZ faults.

Ensure that the CPU and memory of the node where CoreDNS is running are not fully used. Otherwise, the Queries per second (QPS) and response of domain name resolution will be affected.

For details about how to properly configure CoreDNS, see **Configuring CoreDNS**.

Running Multiple Pods

If your application runs in one pod, the application will be unavailable if the pod is abnormal. Use Deployments or other types of replicas to deploy your applications. Each time a pod fails or is terminated, the controller automatically restarts a new pod that has the same specifications as the original one to ensure that a specified number of pods are always running in the cluster.

When creating a workload, set the number of instances to a value greater than 2. If an instance is faulty, the remaining instances still run until Kubernetes automatically creates another pod to compensate for the loss. You can also use HPA and CA (Using HPA and CA for Auto Scaling of Workloads and Nodes) to automatically scale in or out the workloads as required.

Using Containers to Isolate Processes

Containers provide process-level isolation. Each container has its own file system, network, and resource allocation. This prevents interference between different processes and avoids attacks and data leakage from malicious processes. Using containers to isolate processes can improve the reliability, security, and portability of applications.

If several processes work together, create multiple containers in a pod so that they can share the same network, PV, and other resources. Taking the init container as an example. The init containers run before the main containers are started to complete some initialization tasks like configuring environment variables, loading databases or data stores, and pulling Git repositories.

Note that multiple containers in a pod share the lifecycle of this pod. Therefore, if one container is abnormal, the entire pod will be restarted.

Configuring Resource Quotas for a Workload

Configure and adjust resource requests and limits for all workloads.

If too many pods are scheduled to one node, the node will be overloaded and unable to provide services.

To avoid this problem, when deploying a pod, specify the resource request and limit required by the pod. Kubernetes then selects a node with enough idle resources for this pod. In the following example, the Nginx pod requires 1-core CPU and 1024 MiB memory. The actual usage cannot exceed 2-core CPU and 4096 MiB memory.

Kubernetes statically schedules resources. The remaining resources on each node are calculated as follows: Remaining resources on a node = Total resources on the node – Allocated resources (not resources in use). If you manually run a resource-consuming process, Kubernetes cannot detect it.

Additionally, the resource usage must be claimed for all pods. For a pod that does not claim the resource usage, after it is scheduled to a node, Kubernetes does not deduct the resources used by this pod from the node on which it is running. Other pods may still be scheduled to this node.

Deploying an Application in Multiple AZs

You can run pods on nodes in multiple AZs to prevent an application from being affected by faults of a single AZ.

When creating a node, manually specify an AZ for the node.

Figure 5-2 Specifying an AZ of a node



During application deployment, configure anti-affinity policies for pods so that the scheduler can schedule pods across multiple AZs. For details, see **Implementing High Availability for Applications in CCE**. The following is an example:

apiVersion: apps/v1 kind: Deployment metadata: name: web-server labels: app: web-server spec: replicas: 4 selector: matchLabels: app: web-server template: metadata: labels: app: web-server

spec:
containers:
- name: web-app
image: nginx
imagePullSecrets:
- name: default-secret
affinity:
podAntiAffinity: # Workload anti-affinity
preferredDuringSchedulingIgnoredDuringExecution: # Indicates that the rule is met as much as
possible. Otherwise, scheduling cannot be performed when the number of pods exceeds the number of AZs.
- podAffinityTerm:
labelSelector: # Pod label matching rule. Configure anti-affinity policies between pods and their
own labels.
matchExpressions:
- key: app
operator: In
values:
- web-server
topologyKey: topology.kubernetes.io/zone # Topology domain of the AZ where the node is
located
weight: 100

You can also use **Pod Topology Spread Constraints** to deploy pods in multiple AZs.

Deploying the System Add-ons in Multiple AZs

The Deployment pods of CCE system add-ons like CoreDNS and Everest can be deployed in multiple AZs, the same way as deploying an application. This function can satisfy different user requirements.

Mode	Configuration Description	Usage Description	Recommended Configuration Scenario
Preferred	Add-on pods will have labels with the key topology.kubernete s.io/zone for soft anti-affinity deployment, and the anti-affinity type is preferredDuring- SchedulingIgnored- DuringExecution .	Add-on pods will be preferentially scheduled to nodes in different AZs. If resources in some AZs are insufficient, some add-on pods may be scheduled to the same AZ which has enough resources.	No mandatory requirements for multi-AZ DR

Table 5-1 Deployment description

Mode	Configuration Description	Usage Description	Recommended Configuration Scenario
Required	Add-on pods will have labels with the key topology.kubernete s.io/zone for hard anti-affinity deployment, and the anti-affinity type is requiredDuringSche dulingIgnoredDur- ingExecution.	A maximum of one pod of the same add-on can be deployed in each AZ. The number of running pods cannot exceed the number of AZs in the cluster. If the node where the add-on pod runs is faulty, pods running on the faulty node cannot be automatically migrated to other nodes in the same AZ.	Changing number of AZs (This mode is used to prevent all pods from being scheduled to the node in the current AZ in advance.)
Equivale nt mode	Add-on pods will have labels with the key topology.kubernete s.io/zone for configuring topology spread constraints. The pod difference between different topology domains cannot exceed 1 for add-on pods to be evenly distributed in different AZs.	The effect of this mode is between that of the preferred mode and that of the required mode. In the equivalent mode, add- on pods can be deployed in different AZs. Additionally, multiple pods can be deployed in a single AZ when there are more pods than AZs. To use this mode, you need to plan node resources in each AZ in advance to ensure that each AZ has enough node resources for deploying pods. (If there are more than one add-on pods in a single AZ, the nodes to which the add- on pods can be scheduled in each AZ should be one more than the actual add- on pods in the current AZ.) This ensures successful deployment of add-on pods although node resources in some AZ are insufficient and smooth scheduling of add-on pods during update.	Scenarios have high requirements for DR

Configuring Health Check for a Container

Kubernetes automatically restarts pods that are not running properly. This prevents service interruption caused by exceptions of pods. In some cases, however, even if a pod is running, it does not mean that it can provide services properly. For example, a deadlock may occur in a process in a running pod, but Kubernetes does not automatically restart the pod because it is still running. To solve this problem, configure a liveness probe to check whether the pod is healthy. If the liveness probe detects a problem, Kubernetes will restart the pod.

You can also configure a readiness probe to check whether the pod can provide normal services. After an application container is started, it may take some time for initialization. During this process, the pod on which this container is running cannot provide services to external systems. The Services forward requests to this pod only when the readiness probe detects that the pod is ready. When a pod is faulty, the readiness probe can prevent new traffic from being forwarded to the pod.

The startup probe is used to check whether the application container is started. The startup probe ensures that the containers can start successfully before the liveness probe and readiness probe do their tasks. This ensures that the liveness probe and readiness probe do not affect the startup of containers. Configuring the startup probe ensures that the slow-start containers can be detected by the liveness probe to prevent Kubernetes from terminating them before they are started.

You can configure the preceding probes when creating an application. The following is an example:

apiVersion: v1 kind: Pod metadata: labels: test: liveness name: liveness-http spec: containers: - name: liveness image: nginx:alpine aras: - /server livenessProbe: httpGet: path: /healthz port: 80 httpHeaders: - name: Custom-Header value: Awesome initialDelaySeconds: 3 periodSeconds: 3 readinessProbe: exec: command: - cat - /tmp/healthy initialDelaySeconds: 5 periodSeconds: 5 startupProbe: httpGet: path: /healthz port: 80 failureThreshold: 30 periodSeconds: 10

For details, see **Configuring Container Health Check**.

Configuring Auto Scaling

Auto scaling can automatically adjust the number of application containers and nodes as required. Containers and nodes can be quickly scaled out or scaled in to save resources and costs.

Typically, two types of auto scaling may occur during peak hours:

- Workload scaling: When deploying applications in pods, you can configure requested resources and resource limits for the pods to prevent unlimited usage of resources during peak hours. However, after the upper limit is reached, an application error may occur. To resolve this issue, scale in the number of pods to share workloads.
- Node scaling: After the number of pods grows, the resource usage of the node may increase to a certain extent. This results in that the added pods cannot be scheduled. To solve this problem, scale in or out nodes based on the resource usage.

For details, see Using HPA and CA for Auto Scaling of Workloads and Nodes.

Viewing Logs, Monitoring Metrics, and Adding Alarm Rules

- Logging
 - Control plane logs are reported from the master nodes. CCE supports kube-controller-manager, kube-apiserver, kube-scheduler, and audit logs. For details, see Collecting Control Plane Component Logs.
 - Application logs are generated by pods. These logs include logs generated by pods in which the service containers are running and Kubernetes system components like CoreDNS. CCE allows you to configure policies for collecting, managing, and analyzing logs periodically to prevent logs from being over-sized. For details, see Logging Overview.
- Monitoring
 - Metrics of the master nodes: Monitoring these metrics enables you to efficiently identify problems occurred on the master nodes. For details, see Monitoring Metrics of the Master Nodes.
 - Metrics of the applications: CCE can comprehensively monitor applications in clusters by checking these metrics. In addition to standard metrics, you can configure custom metrics of your applications that comply with their specifications to improve the observability. For details, see Monitoring Center Overview.
- Alarm

You can add alarm rules for metrics to detect cluster faults and generate warnings in a timely manner with the monitoring function. This helps you maintain service stability. For details, see **Customized Alarm Configurations**.

5.2 Implementing High Availability for Applications in CCE

Basic Principles

To achieve high availability for your CCE containers, you can do as follows:

- 1. Deploy three master nodes for the cluster.
- 2. Create nodes in different AZs. When nodes are deployed across AZs, you can customize scheduling policies based on your requirements to maximize resource utilization.
- 3. Create multiple node pools in different AZs and use them for node scaling.
- 4. Set the number of pods to be greater than 2 when creating a workload.
- 5. Set pod affinity rules to distribute pods to different AZs and nodes.

Procedure

Assume that there are four nodes in a cluster distributed in different AZs.

```
        kubectl get node -L topology.kubernetes.io/zone,kubernetes.io/hostname

        NAME
        STATUS
        ROLES
        AGE
        VERSION
        ZONE
        HOSTNAME

        192.168.5.112
        Ready
        <none>
        42m
        v1.21.7-r0-CCE21.11.1.B007
        zone01
        192.168.5.112

        192.168.5.179
        Ready
        <none>
        42m
        v1.21.7-r0-CCE21.11.1.B007
        zone01
        192.168.5.179

        192.168.5.252
        Ready
        <none>
        37m
        v1.21.7-r0-CCE21.11.1.B007
        zone02
        192.168.5.252

        192.168.5.8
        Ready
        <none>
        33h
        v1.21.7-r0-CCE21.11.1.B007
        zone03
        192.168.5.85
```

Create workloads according to the following podAntiAffinity rules:

- Pod anti-affinity in an AZ. Configure the parameters as follows:
 - **weight**: A larger weight value indicates a higher priority of scheduling. In this example, set it to **50**.
 - topologyKey: includes a default or custom key for the node label that the system uses to denote a topology domain. A topology key determines the scope where the pod should be scheduled to. In this example, set this parameter to topology.kubernetes.io/zone, which is the label for identifying the AZ where the node is located.
 - labelSelector: Select the label of the workload to realize the anti-affinity between this container and the workload.
- The second one is the pod anti-affinity in the node hostname. Configure the parameters as follows:
 - **weight**: Set it to **50**.
 - topologyKey: Set it to kubernetes.io/hostname.
 - labelSelector: Select the label of the pod, which is anti-affinity with the pod.

```
kind: Deployment
apiVersion: apps/v1
metadata:
name: nginx
namespace: default
spec:
replicas: 2
```

selector: matchLabels: app: nginx template: metadata: labels: app: nginx spec: containers: - name: container-0 image: nginx:alpine resources: limits: cpu: 250m memory: 512Mi requests: cpu: 250m memory: 512Mi affinity: podAntiAffinity: preferredDuringSchedulingIgnoredDuringExecution: - weight: 50 podAffinityTerm: labelSelector: # Select the label of the workload to realize the anti-affinity between this container and the workload. matchExpressions: - key: app operator: In values: - nainx namespaces: - default topologyKey: topology.kubernetes.io/zone # It takes effect in the same AZ. - weight: 50 podAffinityTerm: labelSelector: # Select the label of the workload to realize the anti-affinity between this container and the workload. matchExpressions: - key: app operator: In values: - nainx namespaces: - default topologyKey: kubernetes.io/hostname # It takes effect on the node. imagePullSecrets: - name: default-secret

Create a workload and view the node where the pod is located.

\$ kubectl get pod -owideNAMEREADY STATUSnginx-6fffd8d664-dpwbk1/1Running17s10.0.132192.168.5.112nginx-6fffd8d664-qhclc1/1Running017s10.0.1.133192.168.5.252

Increase the number of pods to 3. The pod is scheduled to another node, and the three nodes are in three different AZs.

\$ kubectl scale --replicas=3 deploy/nginx deployment.apps/nginx scaled
\$ kubectl get pod -owide
NAME READY STATUS RESTARTS AGE IP NODE
nginx-6fffd8d664-8t7rv 1/1 Running 0 3s 10.0.0.9 192.168.5.8
nginx-6fffd8d664-dpwbk 1/1 Running 0 2m45s 10.0.132 192.168.5.112
nginx-6fffd8d664-qhclc 1/1 Running 0 2m45s 10.0.1.133 192.168.5.252

Increase the number of pods to 4. The pod is scheduled to the last node. With podAntiAffinity rules, pods can be evenly distributed to AZs and nodes.

\$ kubectl scalereplicas=4 deploy/nginx							
deployment.apps/nginx	scaled						
\$ kubectl get pod -owic	le						
NAME RE	ADY ST	ATUS R	ESTARTS	AGE	IP	NODE	
nginx-6fffd8d664-8t7rv	1/1	Running	0	2m30s	10.0.0.9	192.168.5.8	
nginx-6fffd8d664-dpwb	k 1/1	Running	0	5m12s	10.0.0.13	32 192.168.5.112	2
nginx-6fffd8d664-h796	b 1/1	Running	0	78s	10.0.1.5	192.168.5.179	
nginx-6fffd8d664-qhclc	1/1	Running	0	5m12s	10.0.1.133	3 192.168.5.252	

5.3 Implementing High Availability for Add-ons in CCE

Application Scenarios

CCE offers various add-ons that enhance the cloud native capabilities of clusters. These add-ons include features like container scheduling and elasticity, cloud native observability, container networking, storage, and security. Helm charts are used to deploy these add-ons. Workload pods of the add-ons are deployed on worker nodes within the clusters.

As add-ons have become more popular, their stability and reliability have become essential requirements. By default, CCE implements a policy for add-on deployment where worker nodes have a hard anti-affinity configuration, and AZs have a soft anti-affinity configuration. This section explains how to enhance the CCE add-on scheduling policy, allowing you to customize the deployment policy according to your requirements.

Deployment Solution

An add-on typically runs as Deployments or DaemonSets. By default, DaemonSet pods are deployed on all nodes. To ensure HA of the add-on, there are multiple pods, AZ affinity policies, and specified node scheduling configured for Deployments.

Pod-level HA solution:

Increasing the Number of Pods: Multi-pod deployment can effectively
prevent service unavailability caused by a single point of failure (SPOF).

Node-level HA solutions:

- **Deploying the Add-on Pods on Dedicated Nodes**: To prevent resource preemption between service applications and core add-ons, it is best to deploy the core add-on pods on dedicated nodes. This ensures that the add-on resources are isolated and restricted on the node level.
- **Deploying the Add-on in Multiple AZs**: Multi-AZ deployment can effectively prevent service unavailability caused by the failure of a single AZ.

Take the CoreDNS add-on as an example. This add-on is deployed as two pods by default in the preferred mode, and the scheduling policies are hard anti-affinity for nodes and soft anti-affinity for AZs. In this case, two nodes are needed to ensure that all add-on pods in the cluster can run properly, and Deployment pods of the add-on can be preferentially scheduled to nodes in different AZs.

The following sections describe how to further improve the add-on SLA.

Increasing the Number of Pods

You can adjust the number of CoreDNS pods ensure high performance and HA.

- Step 1 Log in to the CCE console and click the cluster name to access the cluster console. In the navigation pane, choose Add-ons, locate CoreDNS on the right, and click Edit.
- **Step 2** Increase the number of replicas.

Figure 5-3 Changing the pod quantity

W30049861-turbo 0000 memory Converse Co

Step 3 Click OK.

----End

Deploying the Add-on Pods on Dedicated Nodes

You can adjust the node affinity policy of CoreDNS and make the CoreDNS pods run on dedicated nodes. This can prevent the CoreDNS resources from being preempted by service applications.

A custom policy is used as an example.

- **Step 1** Log in to the CCE console and click the cluster name to access the cluster console. In the navigation pane, choose **Nodes**.
- **Step 2** Click the **Nodes** tab, select the node dedicated for CoreDNS, and click **Labels and Taints** above the node list.

Add the following labels:

- Key: node-role.kubernetes.io/coredns
- Value: true

Add the following taints:

- Key: node-role.kubernetes.io/coredns
- Value: true
- Effect: NoSchedule

Figure 5-4 Adding a label and a taint

Labels and Taints	×
A Changing the Kubernetes labels or taints of a node will only affect the scheduling of new pods on that node, not the existing ones.	
Batch Operation You can add, update, or delete labels or taints of specified nodes in batches.	
Add/Up V Node-role.kubernetes.io/c true Delete	
Add/Up v Taint v Inode-role.kubernetes.io/c true NoSchedule v Delete	
+ Add Operation	

- Step 3 In the navigation pane, choose Add-ons, locate CoreDNS, and click Edit.
- Step 4 Select Custom Policies for Node Affinity and add the preceding node label.Add tolerations for the preceding taint.

Figure 5-5 Adding a toleration

Node Affinity	Not configured	Node Affinity	Specified Node Po	ol Scheduling C	ustom Policies	
	Users can fill in the node la be performed according to	bel of the desired plug-in o the cluster's default sched	deployment to achieve a uling strategy	more flexible scheduling str	ategy. If not filled in, random so	heduling will
	Label Key		Operator ⑦	Label Value		Operation
	node-role.kubernetes.id	o/coredns	In	∽ true		Delete
	\oplus Add Policy					
Toleration 🕜	 By default, two toles seconds. This impre- 	rance policies are added to oves the plug-in availability	o the plug-in. When a noo /. You are advised not to r	de is abnormal, the plug-in o modify the policy.	deployment instance is evicted	after 60
	Taint key 🧿	Operator 🥎 T	aint value	Taint Policy	Toleration Time Windo	Operation
	node.kubernetes.io	Exists ~	-	NoExecute ~	60	Delete
	node.kubernetes.io	Exists ~ -	-	NoExecute ~	60	Delete
	node-role.kubernet	Equal V	true	NoSchedule V	-	Delete

Step 5 Click OK.

----End

Deploying the Add-on in Multiple AZs

By default, the add-on scheduling policy can handle single-node faults. However, if your services require a higher SLA, you can create nodes with different AZ specifications on the node pool page and set the multi-AZ deployment mode of the add-on to the required mode.

- **Step 1** Log in to the CCE console and click the cluster name to access the cluster console.
- **Step 2** Create nodes in different AZs.

To create nodes in different AZs, you can simply repeat these steps. Alternatively, you can create multiple node pools, associate them with different AZ

specifications, and increase the number of nodes in each pool to achieve the same result.

- 1. In the navigation pane, choose **Nodes**, click the **Nodes** tab, and click **Create Node** in the upper right corner.
- 2. On the page displayed, select an AZ for the node.

Figure 5-6 Creating a node

Scheduling

<	Create Node	0		
				1 Configurations
	Configurations Cha	Dose the specifications and OS for a cloud	server to run your containerized applications on	
	AZ	Random AZ1	AZ2 AZ3 ⑦	
	Node Type	Elastic Cloud Server (VM)	Elastic Cloud Server (physical machine)	BMS

- 3. Configure other mandatory parameters following instructions to complete the creation.
- **Step 3** In the navigation pane, choose **Add-ons**. In the right pane, locate **CoreDNS** and click **Edit**.
- **Step 4** In the window that slides out from the right, set **Multi AZ** to **Required** and click **Install**.

Figure 5-7 Changing the multi-AZ deployment mode to the required mode

g							
 1. If you have higher requirements on the plug-in SLA, you are advised to add nodes in different AZs to the cluster and set the multi-AZ deployment mode to forcible mode. Learn more C 2. The plug-in can run properly only when there are at least 2 nodes in the current cluster. Otherwise, the plug-in cannot be used. 							
Multi AZ	Preferred	Equivalent mode	Required				
	Deployment instances are be migrated after a node is	forcibly scheduled to node: s faulty.	s in different AZs. A max	imum of one instance	can run in each AZ. Plug-in	instances may fail to	
Node Affinity	Not configured	Node Affinity	Specified Node Pe	ool Scheduling	Custom Policies		
Toleration ⑦	By default, two tolerance policies are added to the plug-in. When a node is abnormal, the plug-in deployment instance is evicted after 60 seconds. This improves the plug-in availability. You are advised not to modify the policy.						
	Taint key 🕐	Operator 🥎 Ta	aint value	Taint Policy	Toleration Time Win	do Operation	
	node.kubernetes.ic	Exists ~		NoExecute ~	60	Delete	
	node.kubernetes.ic	Exists ~		NoExecute ~	60	Delete	

Step 5 Choose **Workload**, click the **Deployments** tab, and view the CoreDNS pods. Select the **kube-system** namespace to view the pod distribution of the add-on.

Figure 5-8 Viewing the deployment and distribution of CoreDNS pods

< 😄 propieto	< 🙄 🖉 CCE Standard V Namepaor Indexingtion V C Running O Running O Running						\mathbb{E}^0_0 Create Node Pool 🗄 Change Billing Mode \cdots
③ Overview	Deployments StatefulSets DaemonSets	Jobs Cron Jobs Pods					Quick Links Create from YAML Create Workload
Kubernetes Resources							
🗟 Workloads	Delete Export ~						
Gervices & Ingresses	Q Workload Name: coredns × Add filter						× () ()
🖻 Storage	Vorkload Name 🖯	Status	Pods (Normal/All) 🖯	Namespace Θ	Created 🖯	Image Name $ \Theta $	Operation
ConfigMaps and S	• -						
Q Policies		O Kunning	212	kube-system	1 days ago	coredns:1.29.4	Monitor View Log More V
App Templates	The Design of th						
© Custom Resources							
Namespaces							

Step 6 View that the Deployment pods of the add-on has been allocated to nodes in two AZs.



Figure 5-9 Viewing CoreDNS pod distribution

----End

6 Security

6.1 Overview

Based on the shared security responsibility model, CCE safeguards the master nodes in a cluster and CCE components, and provides a series of hierarchical security capabilities at the cluster and container levels. Users are responsible for the security of cluster nodes and comply with the security best practices provided by CCE to perform security configuration and O&M.

Shared Security Responsibilities

Huawei guarantees that its commitment to cyber security will never be outweighed by the consideration of commercial interests. To handle emerging cloud security challenges and pervasive cloud security threats and attacks, Huawei Cloud builds a comprehensive cloud service security assurance system for different regions and industries based on Huawei Cloud's unique software and hardware advantages, laws, regulations, industry standards, and security ecosystem.

Figure 6-1 illustrates the responsibilities shared by Huawei Cloud and CCE users.

- Huawei Cloud responsibility "Security of the Cloud": Huawei Cloud is responsible for protecting the Infrastructure as a service (IaaS), Platform as a service (PaaS), and Software as a service (SaaS) that offered in the Huawei Cloud and the physical environments of the Huawei Cloud data centers that run these services. Huawei Cloud is responsible not only for the security functions and performance of the infrastructure, cloud services, and technologies, but also for the overall cloud O&M security and security compliance.
- **Tenant responsibility "Security in the Cloud"**: Tenants of Huawei Cloud are responsible for the secure and effective management of the tenant-customized configurations of cloud services including IaaS, PaaS, and SaaS. This includes but is not limited to virtual networks, the OS of virtual machine hosts and guests, virtual firewalls, API Gateway, advanced security services, all types of cloud services, tenant data, identity accounts, and key management.

Huawei Cloud Security White Paper elaborates on the ideas and measures for building Huawei Cloud security, including cloud security strategies, the shared responsibility model, compliance and privacy, security organizations and

personnel, infrastructure security, tenant service and security, engineering security, O&M security, and ecosystem security.





CCE Application Scenarios

CCE is a container service built on popular Docker and Kubernetes technologies and offers a wealth of features best suited to enterprises' demand for running container clusters at scale. With unique advantages in system reliability, performance, and compatibility with open-source communities, CCE can suit the diverse needs of enterprises interested in building container clouds.

CCE provides a function list and typical application scenarios. For details about the function list, see **Function Overview**. For details about the application scenarios, see **Application Scenarios**.

Exception Scenarios

You are not advised to use clusters in scenarios that require strong resource isolation. CCE provides tenants with a dedicated, exclusive cluster. Currently, resources such as nodes and networks are not strictly isolated. If no strict security protection measures are available, security risks exist when the cluster is used by multiple external uncontrollable users at the same time. For example, in a development pipeline scenario, when multiple users are allowed to use the pipeline, the service code logic of different users is uncontrollable, and the cluster and services in the cluster may be attacked.

Enabling HSS

Host Security Service (HSS) provides host management, risk prevention, intrusion detection, advanced defense, security operations, and web page anti-tamper functions to comprehensively identify and manage information assets on hosts, monitor risks on hosts in real time, and prevent unauthorized intrusions. You are advised to enable HSS to protect hosts in CCE clusters. For details about HSS and how to use it, see **HSS**.

6.2 Configuration Suggestions on CCE Cluster Security

To ensure security, it is recommended that you configure a cluster in the following manner.

Using the CCE Cluster of the Latest Version

Kubernetes releases a major version in about four months. CCE follows the same frequency as Kubernetes to release major versions. To be specific, a new CCE version is released about three months after a new Kubernetes version is released in the community. For example, Kubernetes v1.19 was released in September 2020 and CCE v1.19 was released in March 2021.

The latest cluster version has known vulnerabilities fixed or provides a more comprehensive security protection mechanism. You are advised to select the latest cluster version when creating a cluster. Before a cluster version is deprecated and removed, upgrade your cluster to a supported version.

Handling Vulnerabilities Released on the Official Website Promptly

CCE releases vulnerabilities irregularly. You need to handle the vulnerabilities in a timely manner. For details, see **Vulnerability Notice**.

Disabling the Automatic Token Mounting Function of the Default Service Account

By default, Kubernetes associates the default service account with every pod, which means that the token is mounted to a container. The container can use this token to pass the authentication by the kube-apiserver and kubelet components. In a cluster with RBAC disabled, the service account who owns the token has the control permissions for the entire cluster. In a cluster with RBAC enabled, the permissions of the service account who owns the token depends on the roles associated by the administrator. The service account's token is generally used by workloads that need to access kube-apiserver, such as coredns, autoscaler, and prometheus. For workloads that do not need to access kube-apiserver, you are advised to disable the automatic association between the service account and token.

Two methods are available:

 Method 1: Set the automountServiceAccountToken field of the service account to false. After the configuration is complete, newly created workloads will not be associated with the default service account by default. Configure this field for each namespace as required.

apiVersion: v1 kind: ServiceAccount metadata: name: default automountServiceAccountToken: false

When a workload needs to be associated with a service account, explicitly set **automountServiceAccountToken** to **true** in the YAML file of the workload.

spec:

```
template:
spec:
serviceAccountName: default
automountServiceAccountToken: true
```

• Method 2: Explicitly disable the function of automatically associating service accounts with workloads.

```
"
spec:
template:
spec:
automountServiceAccountToken: false
```

Configuring Proper Cluster Access Permissions for Users

CCE allows you to create multiple IAM users. Your account can create different user groups, assign different access permissions to different user groups, and add users to the user groups with corresponding permissions when creating IAM users. In this way, users can control permissions on different regions and assign readonly permissions. Your account can also assign namespace-level permissions for users or user groups. To ensure security, it is advised that minimum user access permissions are assigned.

If you need to create multiple IAM users, configure the permissions of the IAM users and namespaces properly.

- For details about how to configure cluster permissions, see Cluster Permissions (IAM-based).
- For details about how to configure namespace permissions, see **Namespace Permissions (Kubernetes RBAC-based)**.

Configuring Resource Quotas for Cluster Namespaces

CCE provides resource quota management, which allows users to limit the total amount of resources that can be allocated to each namespace. These resources include CPU, memory, storage volumes, pods, Services, Deployments, and StatefulSets. Proper configuration can prevent excessive resources created in a namespace from affecting the stability of the entire cluster.

For details, see **Setting a Resource Quota**.

Configuring LimitRange for Containers in a Namespace

With resource quotas, cluster administrators can restrict the use and creation of resources by namespace. In a namespace, a pod or container can use the maximum CPU and memory resources defined by the resource quota of the namespace. In this case, a pod or container may monopolize all available resources in the namespace. You are advised to configure LimitRange to restrict resource allocation within the namespace. The LimitRange parameter has the following restrictions:

• Limits the minimum and maximum resource usage of each pod or container in a namespace.

For example, create the maximum and minimum CPU usage limits for a pod in a namespace as follows:

cpu-constraints.yaml

```
apiVersion: v1
kind: LimitRange
metadata:
name: cpu-min-max-demo-lr
spec:
limits:
- max:
cpu: "800m"
min:
cpu: "200m"
type: Container
```

Then, run **kubectl -n** *<namespace>* **create -f** *cpu-constraints.yaml* to complete the creation. If the default CPU usage is not specified for the container, the platform automatically configures the default CPU usage. That is, the default configuration is automatically added after the container is created.

```
spec:
limits:
- default:
cpu: 800m
defaultRequest:
cpu: 800m
max:
cpu: 800m
min:
cpu: 200m
type: Container
```

• Limits the maximum and minimum storage space that each PersistentVolumeClaim can apply for in a namespace.

storagelimit.yaml

```
apiVersion: v1
kind: LimitRange
metadata:
name: storagelimit
spec:
limits:
- type: PersistentVolumeClaim
max:
storage: 2Gi
min:
storage: 1Gi
```

Then, run **kubectl -n** *<namespace>* **create -f** *storagelimit.yaml* to complete the creation.

Configuring Network Isolation in a Cluster

• Container tunnel network

If networks need to be isolated between namespaces in a cluster or between workloads in the same namespace, you can configure network policies to isolate the networks. For details, see **Network Policies**.

• Cloud Native 2.0 network

In the Cloud Native Network 2.0 model, you can configure security groups to isolate networks between pods. For details, see **Binding a Security Group to a Workload Using a Security Group Policy**.

VPC network

Network isolation is not supported.

Enabling the Webhook Authentication Mode with kubelet

NOTICE

CCE clusters of v1.15.6-r1 or earlier are involved, whereas versions later than v1.15.6-r1 are not.

Upgrade the CCE cluster version to 1.13 or 1.15 and enable the RBAC capability for the cluster. If the version is 1.13 or later, no upgrade is required.

When creating a node, you can enable the kubelet authentication mode by injecting the **postinstall** file (by setting the kubelet startup parameter -- **authorization-node=Webhook**).

Step 1 Run the following command to create a ClusterRoleBinding:

kubectl create clusterrolebinding kube-apiserver-kubelet-admin -clusterrole=system:kubelet-api-admin --user=system:kube-apiserver

Step 2 For an existing node, log in to the node, change authorization mode in /var/ paas/kubernetes/kubelet/kubelet_config.yaml on the node to Webhook, and restart kubelet.

sed -i s/AlwaysAllow/Webhook/g /var/paas/kubernetes/kubelet/ kubelet_config.yaml; systemctl restart kubelet

Step 3 For a new node, add the following command to the post-installation script to change the kubelet permission mode:

sed -i s/AlwaysAllow/Webhook/g /var/paas/kubernetes/kubelet/ kubelet_config.yaml; systemctl restart kubelet

Advanced ECS Settings 🔨	
ECS Group	Anti-affinity ?
	Select V C
	Create ECS Group
Resource Tags	It is recommended that you use TMS's predefined tag function to add the same tag to different cloud resources. View predefined tags
	Key Value
	You can add 5 more tags.
	CCE will automatically create the "CCE-Dynamic-Provisioning-Node=node id" tag.
Agency Name (?)	Select C Create Agency
	Select or create an agency with Agency Type set to "Cloud service" and "ECS BMS" selected as the cloud service.
Pre-installation Script	Enter a script.
	0/1,000
	The script you specify here will be executed before K8S software is installed. Note that if the script is incorrect, K8S software may not be installed successful
Post-installation Script	sed -i s/AlwaysAllow/Webhook/g /var/paas/kubernetes/kubelet/kubelet config.yaml; systemctl restart kubelet
	106/1,000
	The script you specify here will be executed after K8S software is installed. The script is usually used to modify Docker parameters.

----End

Uninstalling web-terminal After Use

The web-terminal add-on can be used to manage CCE clusters. Keep the login password secure and uninstall the add-on when it is no longer needed.

Using Physical Multi-Tenancy

To ensure that resources of different tenants are isolated and to reduce the attack surface, a separate CCE cluster can be created for each tenant. However, this solution comes with some drawbacks.

- High costs: It may increase the costs as more hardware or VM resources need to be purchased, and more manpower and time need to be spent on managing and maintaining these clusters.
- Low resource utilization: Each tenant requires separate hardware or VM resources to run their clusters, which may result in low resource utilization and wastage.
- Increased management complexity: Managing and maintaining separate clusters for each tenant increases management complexity and may require more manpower and time.

Using Secure Containers

Compared with a common runtime, a secure runtime allows each container (pod) to run on its own micro-VM with a separate OS kernel. This ensures secure isolation at the virtualization layer. A secure runtime ensures that kernels, compute resources, and networks are isolated between pods, protecting pod resources and data from any potential theft or interference by other pods. In CCE Turbo clusters, you can deploy workloads using secure runtimes. Additionally, you can purchase physical machines in CCE Turbo clusters and opt for the containerd container engine to use secure containers.

6.3 Configuration Suggestions on CCE Node Security

Handling Vulnerabilities Released on the Official Website Promptly

Before releasing a new image, fix the node vulnerabilities by referring to **Vulnerability Notice**.

Preventing Nodes from Being Exposed to Public Networks

- Do not bind an EIP to a node unless necessary to reduce the attack surface.
- If an EIP must be used, properly configure the firewall or security group rules to restrict access of unnecessary ports and IP addresses.

You may have configured the **kubeconfig.json** file on a node in your cluster. kubectl can use the certificate and private key in this file to control the entire cluster. You are advised to delete unnecessary files from the **/root/.kube** directory on the node to prevent malicious use.

rm -rf /root/.kube

Hardening VPC Security Group Rules

CCE is a universal container platform. Its default security group rules apply to common scenarios. Based on security requirements, you can harden the security group rules set for CCE clusters on the **Security Groups** page of **Network Console**.

For details, see **Configuring Cluster Security Group Rules**.

Hardening Nodes on Demand

CCE cluster nodes use the default settings of open source OSs. After a node is created, you need to perform security hardening according to your service requirements.

In CCE, you can perform hardening as follows:

- Use the post-installation script after the node is created. For details, see the description about **Post-installation Script** in **Advanced Settings** when creating a node. This script is user-defined.
- Build custom images in CCE to create worker nodes. For details about the creation process, see **Creating a Custom CCE Node Image**.

Forbidding Containers to Obtain Host Machine Metadata

If a single CCE cluster is shared by multiple users to deploy containers, containers cannot access the management address (169.254.169.254) of OpenStack, preventing containers from obtaining metadata of host machines.

For details about how to restore the metadata, see the "Notes" section in **Obtaining Metadata**.

MARNING

Notwork

This solution may affect the password change on the ECS console. Therefore, you must verify the solution before rectifying the fault.

Step 1 Obtain the network model and container CIDR of the cluster.

On the **Clusters** page of the CCE console, view the network model and container CIDR of the cluster.

VPC network
vpc-cce
iptables
10.247.0.0/16
10.0.0/16
https://192.168.0.107:5443
Bind EIP

Step 2 Prevent the container from obtaining host metadata.

- VPC network
 - a. Log in to each node in the cluster as user **root** and run the following command:

iptables -I OUTPUT -s {container_cidr} -d 169.254.169.254 -j REJECT

{container_cidr} indicates the container CIDR of the cluster, for example, **10.0.0/16**.

To ensure configuration persistence, write the command to the **/etc/ rc.local** script.

- Run the following commands in the container to access the userdata and metadata interfaces of OpenStack and check whether the request is intercepted: curl 169.254.169.254/openstack/latest/meta_data.json curl 169.254.169.254/openstack/latest/user_data
- Container tunnel network
 - a. Log in to each node in the cluster as user **root** and run the following command:

iptables -I FORWARD -s {container_cidr} -d 169.254.169.254 -j REJECT

{container_cidr} indicates the container CIDR of the cluster, for example, **10.0.0/16**.

To ensure configuration persistence, write the command to the **/etc/ rc.local** script.

- Run the following commands in the container to access the userdata and metadata interfaces of OpenStack and check whether the request is intercepted: curl 169.254.169.254/openstack/latest/meta_data.json curl 169.254.169.254/openstack/latest/user_data
- CCE Turbo cluster

No additional configuration is required for a cluster of a version earlier than v1.23.13-r0, v1.25.8-r0, v1.27.5-r0, or v1.28.3-r0.

For a cluster of v1.23.13-r0, v1.25.8-r0, v1.27.5-r0, v1.28.3-r0, and later version, log in to the CCE console, click the cluster name to access the cluster console. In the navigation pane, choose **Settings**, click the **Network** tab, and view the value of **Pod Access to Metadata**.

- If Pod Access to Metadata is not enabled, no additional configuration is required. The container has been disabled from obtaining the node metadata.
- If Pod Access to Metadata is enabled, take the following steps to disable the container from obtaining the node metadata:
 - i. Log in to each node in the cluster as user **root** and run the following command:

iptables -I FORWARD -s {container_cidr} -d 169.254.169.254 -j REJECT

{container_cidr} indicates the container CIDR block of the cluster, for example, **10.0.0/16**.

To ensure configuration persistence, write the command to the **/etc/rc.local** script.

ii. Run the following commands in the container to access the **userdata** and **metadata** interfaces of OpenStack and check whether the request is intercepted:

curl 169.254.169.254/openstack/latest/meta_data.json curl 169.254.169.254/openstack/latest/user_data

----End

6.4 Configuration Suggestions on CCE Container Runtime Security

Container technology uses Linux namespaces and cgroups to isolate and control resources between containers and nodes. Namespaces provide kernel-level isolation, allowing processes to be restricted from accessing specific sets of resources, such as file systems, networks, processes, and users. Cgroups are a Linux kernel feature that manages and limits the usage of resources, such as CPU, memory, disk, and network, to prevent a single process from consuming too many resources and negatively impacting the overall system performance.

While namespaces and cgroups isolate resources between containers and nodes in an environment, node resources are not visible to containers. However, this isolation does not provide true security isolation because containers share the kernels of their nodes. If a container exhibits malicious behavior or a kernel vulnerability is exploited by attackers, the container may breach resource isolation. This can result in the container escaping and potentially compromising the node and other containers on the node.

To enhance runtime security, there are various mechanisms that can be used to detect and prevent malicious activities in containers. These mechanisms, such as capabilities, seccomp, AppArmor, and SELinux, can be integrated into Kubernetes. By using these mechanisms, container security can be improved and potential threats can be minimized.

Capabilities

Capabilities are a permission mechanism that enables a process to perform certain system operations without requiring full root permissions. This mechanism divides root permissions into smaller, independent permissions known as capabilities. By doing so, the process only obtains the minimum permission set necessary to complete its tasks. This approach enhances system security and helps mitigate potential security risks.

In a containerized environment, you can manage a container's capabilities by configuring its **securityContext**. The following is a configuration example:

```
securityContext:
capabilities:
add:
- NET_BIND_SERVICE
drop:
- all
```

In this way, you can ensure that the container only has the necessary permissions to complete its tasks. This approach eliminates the risk of security breaches caused by excessive permissions. For more information about how to configure capabilities for a container, see **Set capabilities for a Container**.

Seccomp

Seccomp is a mechanism that filters system calls, limiting the ones that processes can use to decrease the potential attack surface. Linux has many system calls, but not all are needed for containerized applications. By restricting the system calls that containers can execute, you can greatly reduce the risk of attacks on your applications.

Seccomp's main principle is to intercept all system calls and only allow trusted ones to pass. Container runtimes, such as Docker and containerd, come with default seccomp configurations that work for most common workloads.

In Kubernetes clusters, you can configure seccomp policies for containers to use the default security configuration. The following shows how to configure seccomp in different versions of Kubernetes clusters:

- For clusters of versions earlier than Kubernetes 1.19, you can use the following annotations to specify the seccomp configuration: annotations: seccomp.security.alpha.kubernetes.io/pod: "runtime/default"
- For clusters of Kubernetes 1.19 and later versions, you can use securityContext to configure seccomp policies.
 securityContext: seccompProfile: type: RuntimeDefault

These configurations use the default seccomp policy, which permits containers to make a limited number of secure system calls. For more configuration options and advanced settings of seccomp, see **Restrict a Container's Syscalls with seccomp**.

AppArmor and SELinux

AppArmor and SELinux are both Mandatory Access Control (MAC) systems that offer a more stringent approach than traditional Discretionary Access Control (DAC) to manage and restrict process permissions. While similar to seccomp in concept, these systems provide more precise access control, including access to file system paths, network ports, and other resources.

AppArmor and SELinux enable administrators to create policies that precisely manage the resources that applications can access. They can limit read and write permissions on specific files or directories, or regulate access to network ports.

Both systems are integrated into Kubernetes, allowing security policies to be applied at the container level.

- For details about how to use AppArmor, see Restrict a Container's Access to Resources with AppArmor.
- For SELinux, you can configure seLinuxOptions in securityContext.

```
securityContext:
seLinuxOptions:
level: "s0:c123,c456"
```

For details, see **Assign SELinux labels to a Container**.

Controlling the Pod Scheduling Scope

The nodeSelector or nodeAffinity is used to limit the range of nodes to which applications can be scheduled, preventing the entire cluster from being threatened due to the exceptions of a single application. For details, see **Node Affinity**.

To achieve strong isolation, like in logical multi-tenancy situations, it is important to have system add-ons run on separate nodes or node pools. This helps keep them separated from service pods and reduces the risk of privilege escalation within a cluster. To do this, you can set the node affinity policy to either **Node Affinity** or **Specified Node Pool Scheduling** on the add-on installation page.

Node Affinity				
Not configured	Node Affinity	Specified Node Pool Scheduling	Custom Policies	
user can specify the no on the default cluster so	de where the plug-in D cheduling policy.	eployment instance is deployed. (0 nodes	s selected) . If this parame	ter is not specified, random scheduling is performed based
Q Select a property	or enter a keyword.			

Suggestions on Container Security Configuration

- Set the computing resource limits (**request** and **limit**) of a container. This prevents the container from occupying too many resources and affecting the stability of the host and other containers on the same node.
- Unless necessary, do not mount sensitive host directories to containers, such as /, /boot, /dev, /etc, /lib, /proc, /sys, and /usr.
- Do not run the sshd process in containers unless necessary.
- Unless necessary, it is not recommended that containers and hosts share the network namespace.
- Unless necessary, it is not recommended that containers and hosts share the process namespace.
- Unless necessary, it is not recommended that containers and hosts share the IPC namespace.
- Unless necessary, it is not recommended that containers and hosts share the UTS namespace.
- Unless necessary, do not mount the sock file of Docker to any container.

Container Permission Access Control

When using a containerized application, comply with the minimum privilege principle and properly set securityContext of Deployments or StatefulSets.

- Configure runAsUser to specify a non-root user to run a container.
- Configure privileged to prevent containers being used in scenarios where privilege is not required.

- Configure capabilities to accurately control the privileged access permission of containers.
- Configure allowPrivilegeEscalation to disable privilege escape in scenarios where privilege escalation is not required for container processes.
- Configure seccomp to restrict the container syscalls. For details, see **Restrict a Container's Syscalls with seccomp** in the official Kubernetes documentation.
- Configure ReadOnlyRootFilesystem to protect the root file system of a container.

Example YAML for a Deployment:

apiVersion: apps/v1
kind: Deployment
metadata:
name: security-context-example
namespace: security-example
snor:
replicas: 1
replicas. I
selector
matchLabels:
app: security-context-example
label: security-context-example
strategy:
rollingUpdate:
maxSurge: 25%
maxUnavailable: 25%
type: RollingUpdate
template.
metadata
annotations
seccomp security alpha kubernetes in/pod-ruptime/default
labole
ladels:
app: security-context-example
label: security-context-example
spec:
containers:
- image:
imagePullPolicy: Always
name: security-context-example
securityContext:
allowPrivilegeEscalation: false
readOnlvRootFilesvstem: true
runAsUser: 1000
capabilities:
add
drop:
- au
mountDath: /otc/localtime
readOnly: true
- mountPath: /opt/write-file-dir
name: tmpfs-example-001
securityContext:
seccompProfile:
type: RuntimeDefault
volumes:
- hostPath:
path: /etc/localtime
type: ""
name: localtime
- emptyDir: {}
name: tmpfs-example-001

Restricting the Access of Service Containers to the Management Plane

To avoid unnecessary service interruption when restricting the service containers on a node from accessing the Kubernetes management plane, consider the following:

• Check whether any containers on the node require access to the cluster management plane.

Once you have restricted the service containers on the node from accessing the management plane, all containers on that node will be unable to access the kube-apiserver of the cluster. Before making the configuration, make sure that none of the containers on the node need to access the kube-apiserver of the cluster.

Keep in mind that certain CCE add-ons, like CCE Advanced HPA, still require access to kube-apiserver. It is not recommended that you configure the access restriction on a node where such add-ons are running.

• Configure taints and affinity for the node.

If the service containers on the node do not need to access kube-apiserver, it is recommended that you configure labels and taints for the node. Additionally, configure **taints, tolerations**, and **node affinity** for the containers on the node. This will prevent other containers from being scheduled to that node, thus avoiding service exceptions.

To restrict the service containers on a node from accessing the management plane, take the following steps:

Step 1 Obtain the container CIDR block and private API server address.

On the **Clusters** page of the CCE console, click the name of the cluster to find the information on the details page.

Networking Configuration				
Network Model	VPC network			
VPC	vpc-cce			
Subnet	subnet-cce			
Container CIDR Block	10.0.0/16			
	Add Container CIDR Block			
Service CIDR Block	10.247.0.0/16			
Forwarding	iptables			
Default Node Security Grou	IP cce-test-cce-node-99es4 🗗 🖉			
Connection Informatio	on			
Private IP	https://192.168.0.80.5443			
FID.	nitp3.1102.100.000 0440			
EIP	Bind			
Custom SAN	🖉			
kubectl	Learn more			
Certificate Authentication	X.509 certificate Download			

Step 2 Configure access rules.

- CCE cluster: Log in to each node in the cluster as user **root** and run the following command:
 - VPC network iptables -I OUTPUT -s {container_cidr} -d {Private API server IP} -j REJECT
 - Container tunnel network
 iptables -I FORWARD -s {container_cidr} -d {Private API server IP} -j REJECT

{container_cidr} indicates the container CIDR of the cluster, for example, 10.0.0/16.

To ensure configuration persistence, write the command to the **/etc/rc.local** script.

- CCE Turbo cluster: Add an outbound rule to the ENI security group of the cluster.
 - a. Log in to the VPC console.
 - b. In the navigation pane, choose **Access Control** > **Security Groups**.
 - c. Locate the ENI security group corresponding to the cluster and name it in the format of *{Cluster name}-cce-eni-{Random ID}*. Click the security group name and configure rules.
 - d. Click the **Outbound Rules** tab and click **Add Rule** to add an outbound rule for the security group.
 - Priority: Set it to 1.
 - Action: Select Deny, indicating that the access to the destination address is denied.
 - **Type**: Select **IPv4**.
 - Protocol & Port: Enter 5443 based on the port in the intranet API server address.
 - Destination: Select IP address and enter the IP address of the internal API server.
 - e. Click **OK**.
- **Step 3** Run the following command in the container to access kube-apiserver and check whether the request is intercepted: curl -k https://{*Private API server IP*}:5443

----End

Properly Setting Volume Propagation

When mounting a host path, set the propagation mode to **None**. Exercise caution when using the **Bidirectional** mode.

The **mountPropagation** field in **Container.volumeMounts** controls the mount propagation feature of a volume. Value options are as follows:

 None: the default value, equal to the private mount propagation option described in Linux kernel documentation. This volume mount will not receive any subsequent mounts that are mounted to this volume or any of its subdirectories by the host. In similar fashion, no mounts created by the container will be visible on the host.

- HostToContainer: equal to the rslave mount propagation option described in Linux kernel documentation. This volume mount will receive all subsequent mounts that are mounted to this volume or any of its subdirectories. In other words, if the host mounts anything inside the volume mount, the container will see it mounted there. If a Bidirectional pod mounts anything to the same volume, this change is visible to all HostToContainer pods.
- Bidirectional: equal to the rshared mount propagation option described in Linux kernel documentation. All volume mounts created by the container will be propagated back to the host and to all containers of all pods that use the same volume.

Bidirectional mount propagation can be dangerous. It can damage the host operating system and therefore it is allowed only in privileged containers. An example is as follows:

```
apiVersion: v1
kind: Pod
metadata:
  name: security-mount
  label:
   app: security-mount
spec:
  containers:
  - name: security-mount
   image: ...
   volumeMounts:
   - name: mount-none
    mountPath: /opt
    mountPropagation: None
  volumes:
  - name: mount-none
   hostPath:
    path: /opt/
```

For details, see the **community documentation**.

6.6 Configuration Suggestions on CCE Container Image Security

Container images are the primary defense against external attacks and are crucial for securing applications, systems, and the entire supply chain. If an image is insecure, it can become a vulnerability for attackers to exploit. This can lead to the container escaping to its node, allowing attackers to access sensitive data on the node or use it as a launching pad to gain control over the entire cluster or tenant account. This section describes some recommended configurations to mitigate such risks.

Minimizing a Container Image

To improve container image security, it is recommended that you remove any unnecessary binary files. When using an unknown image from Docker Hub, you are advised to review the image content with a tool like Dive. Dive provides layerby-layer details of an image, helping to identify potential security risks. For details, see **Dive**. For improved security, it is recommended that you delete binary files with setuid and setgid permissions, because these can be exploited to elevate permissions. It is also wise to remove shell tools and applications that could be used maliciously, like nc and curl. To locate files with setuid and setgid bits, use the following command:

find / -perm /6000 -type f -exec ls -ld {} \;

To remove special permissions from the obtained files, add the following command to your container image:

RUN find / -xdev -perm /6000 -type f -exec chmod a-s {} \; || true

Using Multi-Stage Builds

Multi-stage builds are a great way to create container images efficiently, especially in the CI process. With multi-stage builds, you can perform lint checks on source code or static code analysis during the build process, providing quick feedback to developers. There is no need to wait for the entire build to finish.

Multi-stage builds offer significant security advantages by allowing developers to include only necessary components in container images, excluding build tools and other unnecessary binary files. This approach reduces the attack surface of images and improves overall security.

For more information about the concepts, best practices, and advantages of multistage builds, see the **Docker documentation**. This will help you create streamlined and secure container images while optimizing development and deployment processes.

Using SWR

SWR provides easy, secure, reliable management of container images throughout their lifecycles, featuring image push, pull, and deletion.

SWR stands out for its precise permissions management, allowing administrators to customize access permissions for different users with read, edit, and manage levels. This ensures image security and compliance, meeting the needs of team collaboration.

Additionally, SWR offers automatic deployment capabilities. You can set a trigger to automatically deploy updated image versions. When a new image version is released, SWR automatically triggers the application that uses the image in CCE to update it, streamlining CI/CD.

To further enhance SWR's security and flexibility, fine-grained permissions control can be added to IAM users. For details about authorization management, see **User Permissions**.

Scanning an Image Using SWR

With SWR, you can easily scan and secure your images with just a few clicks. Image scanning provides a thorough security check for your private images in repositories. It detects potential vulnerabilities and offers rectification suggestions.

Using an Image Signature and Configuring a Signature Verification Policy

Image signature verification is a security measure that confirms whether a container image has been tampered with after its creation. The image creator can sign the image content, and a user can verify the image's integrity and source by checking the signature.

This verification is crucial in maintaining container image security. By using image signature verification, organizations can guarantee the security and reliability of their containerized applications and safeguard them from potential security risks.

Adding the USER Instruction to a Dockerfile to Run Commands as a Nonroot User

Properly configuring user permissions during container build and deployment can greatly enhance container security. This not only helps prevent potential malicious activities, but also aligns with the principle of least privilege (PoLP).

By setting the USER instruction in Dockerfiles, subsequent commands are executed as non-root users, which is a standard security practice.

- Limited permissions: Running a container as a non-root user can also mitigate potential security risks, because attackers cannot gain full control over the node even if the container is attacked.
- Restricted access: Non-root users typically have limited permissions, which restrict their access to and operation capabilities on node resources.

In addition to Dockerfiles, the **securityContext** field in **podSpec** of Kubernetes can be used to configure user and group IDs and enforce security policies during container deployment.

6.7 Configuration Suggestions on CCE Secret Security

Currently, CCE has configured static encryption for secret resources. The secrets created by users will be encrypted and stored in etcd of the CCE cluster. Secrets can be used in two modes: environment variable and file mounting. No matter which mode is used, CCE still transfers the configured data to users. Therefore, it is recommended that:

- 1. Do not record sensitive information in logs.
- For the secret that uses the file mounting mode, the default file permission mapped in the container is 0644. Configure stricter permissions for the file. For example: apiversion: v1

```
kind: Pod
metadata:
name: mypod
spec:
containers:
- name: mypod
image: redis
volumeMounts:
- name: foo
mountPath: "/etc/foo"
volumes:
- name: foo
```

secretName: mysecret defaultMode: 256

In **defaultMode: 256**, **256** is a decimal number, which corresponds to the octal number **0400**.

3. When the file mounting mode is used, configure the secret file name to hide the file in the container.

```
apiVersion: v1
kind: Secret
metadata:
name: dotfile-secret
data:
.secret-file: dmFsdWUtMg0KDQo=
apiVersion: v1
kind: Pod
metadata:
name: secret-dotfiles-pod
spec:
 volumes:
 - name: secret-volume
  secret:
   secretName: dotfile-secret
 containers:
 - name: dotfile-test-container
  image: k8s.gcr.io/busybox
  command:
  - Is
  - "-1"
  - "/etc/secret-volume"
  volumeMounts:
  - name: secret-volume
   readOnly: true
   mountPath: "/etc/secret-volume"
```

In this way, **.secret-file** cannot be seen by running **ls** -**l** in the **/etc/secret-volume/** directory, but can be viewed by running **ls** -**al**.

4. Encrypt sensitive information before creating a secret and decrypt the information when using it.

Using a Bound ServiceAccount Token to Access a Cluster

The secret-based ServiceAccount token does not support expiration time or auto update. In addition, after the mounting pod is deleted, the token is still stored in the secret. Token leakage may incur security risks. A bound ServiceAccount token is recommended for CCE clusters of version 1.23 or later. In this mode, the expiration time can be set and is the same as the pod lifecycle, reducing token leakage risks. An example is as follows:

```
apiVersion: apps/v1
kind: Deployment
metadata:
 name: security-token-example
 namespace: security-example
spec:
 replicas: 1
 selector:
  matchLabels:
    app: security-token-example
    label: security-token-example
 template:
  metadata:
    annotations:
     seccomp.security.alpha.kubernetes.io/pod: runtime/default
    labels:
```

```
app: security-token-example
  label: security-token-example
spec:
 serviceAccountName: test-sa
 containers:
  - image: ..
   imagePullPolicy: Always
   name: security-token-example
 volumes:

    name: test-projected

   projected:
    defaultMode: 420
    sources:
      - serviceAccountToken:
        expirationSeconds: 1800
         path: token
      - configMap:
        items:
          - key: ca.crt
           path: ca.crt
         name: kube-root-ca.crt
      - downwardAPI:
         items:
          - fieldRef:
             apiVersion: v1
             fieldPath: metadata.namespace
           path: namespace
```

For details, see Managing Service Accounts.

Using the CCE Secrets Manager for DEW Add-on

The CCE Secrets Manager for DEW add-on (dew-provider) is used to interconnect with DEW. This add-on allows you to mount secrets stored outside a cluster (DEW for storing sensitive information) to pods. In this way, sensitive information can be decoupled from the cluster environment, which prevents information leakage caused by program hardcoding or plaintext configuration. For details, see CCE Secrets Manager for DEW.

6.8 Configuration Suggestions on CCE Workload Identity Security

A workload identity enables workloads within a cluster to act as IAM users, granting them access to cloud services without the need for an IAM account's AK/SK. This helps minimize security risks.

This section describes how to use workload identities in CCE.

Notes and Constraints

The cluster version must be v1.19.16 or later.

Procedure

- **Step 1** Obtain the public key of the cluster service account token from CCE. For details, see **Step 1: Obtain the Public Key for Signature of the CCE Cluster**.
- Step 2 Create an identity provider on IAM. For details, see Step 2: Configure an Identity Provider.

Step 3 Obtain an IAM token from the workload and simulate an IAM user to access a cloud service. For details, see **Step 3: Use a Workload Identity**.

The procedure is as follows:

- 1. Deploy the application pod and obtain the OpenID Connect ID token file by mounting the identity provider.
- 2. Use the mounted OpenID Connect ID token file in programs in the pod to access IAM and obtain a temporary IAM token.
- 3. Access the cloud service using the IAM token in programs in the pod.



Figure 6-2 Workflow

Step 1: Obtain the Public Key for Signature of the CCE Cluster

- **Step 1** Use kubectl to access the target cluster.
- **Step 2** Obtain the public key:

kubectl get --raw /openid/v1/jwks

The returned result is the public key of the cluster. The following is an example of the command output:

```
# kubectl get --raw /openid/v1/jwks
{"keys":[{"use":"sig","kty":"RSA","kid":"*****","alg":"RS256","n":"*****","e":"AQAB"}]}
```

```
----End
```

Step 2: Configure an Identity Provider

Step 1 Log in to the IAM console, choose **Identity Providers** in the navigation pane, and click **Create Identity Provider** in the upper right corner. On the displayed page, set **Protocol** to **OpenID Connect** and **SSO Type** to **Virtual user** and click **OK**.

Identity Providers / Create Identity Provider				
* Name	workload_identity			
* Protocol	OpenID Connect 🔹			
★ SSO Type	Virtual user 💌			
* Status	Enabled Disabled			
Description	Enter a brief description.]		
		6		
	0/25	5		
	Calicei			

Step 2 In the identity provider list, locate the row containing the new identity provider and click **Modify** in the **Operation** column to modify the identity provider information.

Access Type: Select Programmatic access.

Configuration Information

- Identity Provider URL: Enter https://kubernetes.default.svc.cluster.local.
- Client ID: Enter a client ID, which will be used when you create a container.

A client ID cannot contain only digits. If the client ID consists only of digits, enclose it in double quotation marks ("") when editing the YAML file for the workload. For example, if the client ID is **123456789**, it should be entered as **"123456789"** in the YAML file.

• Signing Key: Enter the JWKS of the CCE cluster obtained in Step 1: Obtain the Public Key for Signature of the CCE Cluster.

Configuration Information ⑦							
Identity Provider URL	https://kubernetes.default.svc.cluster.local						
Client ID	client_id						
Scopes	openid 🔘 👻						
Signing Key	<pre>{ "keys": [{ "kky": "RSA", ''e": "AQAB", "use:" 'sig', "kd": " "adj": "RS256", "n" "adj": "RS256", "n" } } </pre>						
	576/30,000						

Identity Conversion Rules

An identity conversion rule maps the ServiceAccount of a workload to an IAM user.

For example, create a ServiceAccount named **oidc-token** in namespace **default** of the cluster and map it to user group **demo**. If you use the identity provider ID to access cloud services, you have the permissions of the **demo** user group. The attribute must be **sub**. The value is in the format of

system:serviceaccount:Namespace:ServiceAccountName.

Create Rule				
★ Username	test			
User Groups	demo 💿	¥		
Rule Conditions	s e for addition: 9			
Attribute	Condition	Value	Operation	
sub	any_one_of	▼ system:serviceaccount:default.oidc-token	Delete	
(+) Add				

Rules are in the JSON format as follows:


Step 3 Click OK.

----End

Step 3: Use a Workload Identity

Step 1 Create a ServiceAccount, whose name must be the value of **ServiceAccountName** set in **Step 2: Configure an Identity Provider**.

apiVersion: v1 kind: ServiceAccount metadata: name: oidc-token

Step 2 Mount the identity provider to the workload and obtain the OpenID Connect ID token file.

```
An example is as follows:
apiVersion: apps/v1
kind: Deployment
metadata:
 name: nginx
spec:
 replicas: 1
 selector:
  matchLabels:
   app: nginx
   version: v1
 template:
  metadata:
   labels:
     app: nginx
     version: v1
  spec:
   containers:
    - name: container-1
     image: nginx:latest
     volumeMounts:
     - mountPath: "/var/run/secrets/tokens"
                                             # Mount the service account token generated by
Kubernetes to the /var/run/secrets/tokens/oidc-token file.
      name: oidc-token
   imagePullSecrets:
   - name: default-secret
   serviceAccountName: oidc-token
                                       # Name of the created ServiceAccount
   volumes:
    - name: oidc-token
     projected:
      defaultMode: 420
      sources:
      - serviceAccountToken:
         audience: client_id # Must be the client ID of the identity provider.
         expirationSeconds: 7200 # Expiry period
         path: oidc-token
                                 # Path name, which can be customized
```

Step 3 After the creation is complete, log in to the container. The content of the **/var/run/secrets/tokens/oidc-token** file is the service account token generated by Kubernetes.

If the service account token is used for more than 24 hours or 80% of its expiry period, kubelet will automatically rotate the service account token.

Step 4 Use the OpenID Connect ID token to call the API for Obtaining a Token with an OpenID Connect ID Token. The X-Subject-Token field in the response header is the IAM token. Then, you can use this token to access cloud services.

The following shows an example:

```
curl -i --location --request POST 'https://{{iam endpoint}}/v3.0/OS-AUTH/id-token/tokens' \
--header 'X-Idp-Id: workload_identity \
--header 'Content-Type: application/json' \
--data @token_body.json
```

Specifically:

- {{iam endpoint}} indicates the endpoint of IAM. For details, see Regions and Endpoints.
- **workload_identity** is the identity provider name, which is the same as that configured in **Step 2: Configure an Identity Provider**.
- token_body.json is a local file and its content is as follows:

```
{
    "auth":{
        "id_token":{
        "id": "eyJhbGciOiJSU..."
     },
     "scope":{
        "project":{
        "id": "46419baef4324...",
        "name":******
     }
     }
}
```

- \$.auth.id_token.id: The value is the content of the /var/run/secrets/ tokens/oidc-token file in the container.
- \$.auth.scope.project.id: indicates the project ID. To obtain the value, see
 Obtaining a Project ID.
- \$.auth.scope.project.name: indicates the project name, for example, cnnorth-4.

----End

7 Auto Scaling

7.1 Using HPA and CA for Auto Scaling of Workloads and Nodes

Application Scenarios

The best way to handle surging traffic is to automatically adjust the number of machines based on the traffic volume or resource usage, which is called scaling.

To prevent pods from using up node resources without limit during peak hours, it is a common practice to specify request and limit values for pods before deploying an application. However, this approach may encounter a resource bottleneck as an application exception may occur once the upper limit of resource usage is reached. To address this problem, you can scale the pods in or out to distribute the workload. If the node resource usage reaches its upper limit after pods are added and new pods cannot be scheduled, you can scale the nodes in or out depending on the node resource usage.

Solution

Two major auto scaling policies are HPA (Horizontal Pod Autoscaling) and CA (Cluster AutoScaling). HPA is for workload auto scaling and CA is for node auto scaling.

HPA and CA work with each other. HPA requires sufficient cluster resources for successful scaling. When the cluster resources are insufficient, CA is needed to add nodes. If HPA reduces workloads, the cluster will have a large number of idle resources. In this case, CA needs to release nodes to avoid resource waste.

As shown in **Figure 7-1**, HPA performs scale-out based on the monitoring metrics. When cluster resources are insufficient, newly created pods are in Pending state. CA then checks these pending pods and selects the most appropriate node pool based on the configured scaling policy to scale out the node pool. For details about how HPA and CA work, see **Workload Scaling Mechanisms** and **Node Scaling Mechanisms**.



Figure 7-1 HPA and CA working flows

Using HPA and CA enables automatic scaling for most scenarios while also providing monitoring capabilities.

This section uses an example to describe the auto scaling process using HPA and CA policies together.

Preparations

Step 1 Create a cluster with one node. The node should have 2 vCPUs and 4 GiB of memory, or a higher flavor, as well as an EIP to allow external access. If no EIP is bound to the node during node creation, you can manually bind one on the ECS console after creating the node.

Network Settings	Configure networking reso	configure networking resources for node and application communication.					
VPC	vpc-I30000890						
Node Subnet	subnet-1 (192.168.0.0/2	4)(Cluster subnet)	ipv6 • C Avai	able Subnet IP Addresse	es: 187		
	👃 If the default DNS	server of the subnet is mo	dified, ensure that the cust	om DNS server can resolv	e the OBS service domair	n name. Otherwise, th	e node cannot be created.
Node IP	Random	Custom					
EIP	Do not use	Use existing	Automatic assign	0			
Specifications	5_TYW	5_g-vm					
Billed By	Bandwidth-bas Scenarios with he	ed charging avy or stable traffic					
Bandwidth	1	5	10	50	100	- 5	+ Mbit/s

Step 2 Install add-ons for the cluster.

- autoscaler: node scaling add-on
- metrics-server: an aggregator of resource usage data in a Kubernetes cluster. It can collect measurement data of major Kubernetes resources, such as pods, nodes, containers, and Services.
- **Step 3** Log in to the cluster node and run a computing-intensive application. When a user sends a request, the result needs to be calculated before being returned to the user.

1. Create a PHP file named **index.php** to calculate the square root of the request for 1,000,000 times before returning **OK!**. vi index.php

```
The file content is as follows:
<?php
$x = 0.0001;
for ($i = 0; $i <= 1000000; $i++) {
    $x += sqrt($x);
    }
echo "OK!";
?>
```

2. Compile a **Dockerfile** file to build an image. vi Dockerfile

```
The content is as follows:
FROM php:5-apache
COPY index.php /var/www/html/index.php
RUN chmod a+rx index.php
```

3. Run the following command to build an image named **hpa-example** with the tag **latest**.

```
docker build -t hpa-example:latest .
```

4. (Optional) Log in to the SWR console, choose **Organizations** in the navigation pane, and click **Create Organization** in the upper right corner.

Skip this step if you already have an organization.

5. In the navigation pane, choose **My Images** and then click **Upload Through Client**. On the page displayed, click **Generate a temporary login command**

and click \square to copy the command.

- 6. Run the login command copied in the previous step on the cluster node. If the login is successful, the message "Login Succeeded" is displayed.
- 7. Tag the hpa-example image.

docker tag {*Image name 1:Tag 1*}/{*Image repository address*}/{*Organization name*}/{*Image name 2:Tag 2*}

- *{Image name 1:Tag 1}*: name and tag of the local image to be uploaded.
- *{Image repository address}*: the domain name at the end of the login command in **login command**. It can be obtained on the SWR console.
- {Organization name}: name of the created organization.
- *{Image name 2:Tag 2}*: desired image name and tag to be displayed on the SWR console.

The following is an example:

docker tag hpa-example:latest swr.ap-southeast-1.myhuaweicloud.com/ cloud-develop/hpa-example:latest

8. Push the image to the image repository.

docker push {*Image repository address*}*|*{*Organization name*}*|*{*Image name 2:Tag 2*}

The following is an example:

docker push swr.ap-southeast-1.myhuaweicloud.com/cloud-develop/hpaexample:latest

The following information will be returned upon a successful push: 6d6b9812c8ae: Pushed

fe4c16cbf7a4: Pushed latest: digest: sha256:eb7e3bbd*** size: **

To view the pushed image, go to the SWR console and refresh the **My Images** page.

----End

Creating a Node Pool and a Node Scaling Policy

- **Step 1** Log in to the CCE console, access the created cluster, click **Nodes** on the left, click the **Node Pools** tab, and click **Create Node Pool** in the upper right corner.
- **Step 2** Configure the node pool.
 - Node Type: Select a node type.
 - **Specifications**: 2 vCPUs | 4 GiB

Retain the default values for other parameters. For details, see **Creating a Node Pool**.

Step 3 Locate the row containing the newly created node pool and click **Auto Scaling** in the upper right corner. For details, see **Creating a Node Scaling Policy**.

If the CCE Cluster Autoscaler add-on is not installed in the cluster, install it first. For details, see **CCE Cluster Autoscaler**.

- **Customize scale-out rules.**: Click **Add Rule**. In the dialog box displayed, configure parameters. If the CPU allocation rate is greater than 70%, a node is added to each associated node pool. A node scaling policy needs to be associated with a node pool. Multiple node pools can be associated. When you need to scale nodes, node with proper specifications will be added or reduced from the node pool based on the minimum waste principle.
- **Nodes**: Modify the node quantity range. The number of nodes in a node pool will always be within the range during auto scaling.
- **Cooldown Period**: a period during which the nodes added in the current node pool cannot be scaled in
- **Specifications**: Configure whether to enable auto scaling for node flavors in a node pool.

Step 4 Click OK.

----End

Creating a Workload

Use the hpa-example image to create a Deployment with one replica. The image path is related to the organization uploaded to the SWR repository and needs to be replaced with the actual value.

```
kind: Deployment
apiVersion: apps/v1
metadata:
name: hpa-example
spec:
replicas: 1
selector:
matchLabels:
app: hpa-example
```

```
template:
  metadata:
    labels:
     app: hpa-example
  spec:
   containers:
    - name: container-1
     image: 'hpa-example:latest' # Replace it with the address of the image you uploaded to SWR.
     resources:
                        # The value of limits must be the same as that of requests to prevent flapping
      limits:
during scaling.
       cpu: 500m
       memory: 200Mi
      requests:
       cpu: 500m
       memory: 200Mi
    imagePullSecrets:
    - name: default-secret
```

Then, create a NodePort Service for the workload so that the workload can be accessed from external networks.

NOTE

To allow external access to NodePort Services, allocate an EIP for the node in the cluster. After the allocation, synchronize node data. For details, see **Synchronizing Data with Cloud Servers.** If the node has already bound with an EIP, you do not need to create one.

Alternatively, you can create a Service with an ELB load balancer for external access. For details, see Using kubectl to Create a Service (Automatically Creating a Load Balancer).

```
kind: Service
apiVersion: v1
metadata:
name: hpa-example
spec:
ports:
- name: cce-service-0
protocol: TCP
port: 80
targetPort: 80
nodePort: 31144
selector:
app: hpa-example
type: NodePort
```

Creating an HPA Policy

Create an HPA policy. As shown below, the policy is associated with the hpaexample workload, and the target CPU usage is 50%.

There are two other annotations. One annotation defines the CPU thresholds, indicating that scaling is not performed when the CPU usage is between 30% and 70% to prevent impact caused by slight fluctuation. The other is the scaling time window, indicating that after the policy is successfully executed, a scaling operation will not be triggered again in this cooling interval to prevent impact caused by short-term fluctuation.

```
apiVersion: autoscaling/v2
kind: HorizontalPodAutoscaler
metadata:
name: hpa-policy
annotations:
extendedhpa.metrics: '[{"type":"Resource","name":"cpu","targetType":"Utilization","targetRange":
{"low":"30","high":"70"}}]'
extendedhpa.option: '{"downscaleWindow":"5m","upscaleWindow":"3m"}'
```

spec:			
scaleTargetRef:			
kind: Deployment			
name: hpa-example			
apiVersion: apps/v1			
minReplicas: 1			
maxReplicas: 100			
metrics:			
- type: Resource			
resource:			
name: cpu			
target:			
type: Utilization			
averageUtilization: 50			

Configure the parameters as follows if you are using the console.

Pod Range	1 ~	100 When a polic	cy is triggered, the workload pods are s	scaled within this range.	
Cooldown Period	For scale-down 5 m	inutes For scale-up 3 m ered, scale-down or scale-up will not trigge	inutes red again within this cooldown period.		
Rules	System Metrics Metric ⑦	Expected Value ②	Threshold ②		Operation
	CPU usage 🔻	50 %	Scale down 30	% Scale up 70 %	Delete
	🕀 Add Rule				

Observing the Auto Scaling Process

```
Step 1 Check the cluster node status. In the following example, there are two nodes.
```

 # kubectl get node

 NAME
 STATUS
 ROLES
 AGE
 VERSION

 192.168.0.183
 Ready
 <none>
 2m20s
 v1.17.9-r0-CCE21.1.1.3.B001-17.36.8

 192.168.0.26
 Ready
 <none>
 55m
 v1.17.9-r0-CCE21.1.1.3.B001-17.36.8

Check the HPA policy. The CPU usage of the target workload is 0%.

kubectl get hpa hpa-policy NAME REFERENCE TARGETS MINPODS MAXPODS REPLICAS AGE hpa-policy Deployment/hpa-example 0%/50% 1 100 1 4m

Step 2 Run the following command to access the workload. In the following command, {ip:port} indicates the access address of the workload, which can be queried on the workload details page.

while true;do wget -q -O- http://{ip:port}; done

If no EIP is displayed, the cluster node has not been assigned any EIP. Allocate one, bind it to the node, and synchronize node data. For details, see **Synchronizing Data with Cloud Servers**.

Observe the scaling process of the workload.

# kubectl g	et hpa hpa-policywatc	:h							
NAME	REFERENCE T	FARGE	ETS	MINP	ODS	MAXPO	DS	REPLICAS	AGE
hpa-policy	Deployment/hpa-examp	ple (0%/50)%	1	100	1	4m	
hpa-policy	Deployment/hpa-examp	ple ´	190%,	/50%	1	100	1	4m2	3s
hpa-policy	Deployment/hpa-examp	ple ´	190%/	/50%	1	100	4	4m3	1s
hpa-policy	Deployment/hpa-example	ple 2	200%/	/50%	1	100	4	5m1	6s
hpa-policy	Deployment/hpa-examp	ple 2	200%/	/50%	1	100	4	6m1	6s
hpa-policy	Deployment/hpa-examp	ple 8	85%/5	50%	1	100	4	7m16	ôs

hpa-policy	Deployment/hpa-example	81%/50%	1	100	4	8m16s
hpa-policy	Deployment/hpa-example	81%/50%	1	100	7	8m31s
hpa-policy	Deployment/hpa-example	57%/50%	1	100	7	9m16s
hpa-policy	Deployment/hpa-example	51%/50%	1	100	7	10m
hpa-policy	Deployment/hpa-example	58%/50%	1	100	7	11m

You can see that the CPU usage of the workload is 190% at 4m23s, which exceeds the target value. In this case, scaling is triggered to expand the workload to four replicas/pods. In the subsequent several minutes, the CPU usage does not decrease until 7m16s. This is because the new pods may not be successfully created. The possible cause is that resources are insufficient and the pods are in the pending state. During this period, nodes are being scaled out.

At 7m16s, the CPU usage decreases, indicating that the pods are successfully created and start to bear traffic. The CPU usage decreases to 81% at 8m, still greater than the target value (50%) and the high threshold (70%). Therefore, 7 pods are added at 9m16s, and the CPU usage decreases to 51%, which is within the range of 30% to 70%. From then on, the number of pods remains 7.

In the following output, you can see the workload scaling process and the time when the HPA policy takes effect.

kubectl describe deploy hpa-example

•••				
Events:				
Type Re	eason	Age Fro	m Message	
Normal S	ScalingReplica	aSet 25m	deployment-controller	Scaled up replica set hpa-example-79dd795485
to 1				
Normal S	ScalingReplica	aSet 20m	deployment-controller	Scaled up replica set hpa-example-79dd795485
to 4				
Normal S	ScalingReplica	aSet 16m	deployment-controller	Scaled up replica set hpa-example-79dd795485
to 7	• •			
# kubectl d	describe hpa l	hpa-policy		
Events:				
Type Re	eason	Age Fro	m Messa	ge
				-
Normal S	SuccessfulRes	cale 20m	horizontal-pod-autosc	aler New size: 4; reason: cpu resource utilization
(percentag	e of request)	above targ	get	
Normal S	SuccessfulRes	cale 16m	horizontal-pod-autosc	aler New size: 7; reason: cpu resource utilization

Normal SuccessfulRescale 16m horizontal-pod-autoscaler New size: 7; reason: cpu resource utilization (percentage of request) above target

Check the number of nodes. The following output shows that two nodes are added.

 # kubectl get node
 STATUS
 ROLES
 AGE
 VERSION

 192.168.0.120
 Ready
 <none>
 3m5s
 v1.17.9-r0-CCE21.1.1.3.B001-17.36.8

 192.168.0.136
 Ready
 <none>
 6m58s
 v1.17.9-r0-CCE21.1.1.3.B001-17.36.8

 192.168.0.183
 Ready
 <none>
 18m
 v1.17.9-r0-CCE21.1.1.3.B001-17.36.8

 192.168.0.26
 Ready
 <none>
 71m
 v1.17.9-r0-CCE21.1.1.3.B001-17.36.8

You can also view the scaling history on the console. For example, the CA policy is executed once when the CPU allocation rate in the cluster is greater than 70%, and the number of nodes in the node pool is increased from 2 to 3. The new node is automatically added by autoscaler based on the pending state of pods in the initial phase of HPA.

The node scaling process is as follows:

1. After the number of pods changes to 4, the pods are in Pending state due to insufficient resources. As a result, the default scale-out policy of the autoscaler add-on is triggered, and the number of nodes is increased by one.

2. The second node scale-out is triggered because the CPU allocation rate in the cluster is greater than 70%. As a result, the number of nodes is increased by one, which is recorded in the scaling history on the console. Scaling based on the allocation rate ensures that the cluster has sufficient resources.

Step 3 Stop accessing the workload and check the number of pods.

# kubectl g	# kubectl get hpa hpa-policywatch							
NAME	REFERENCE TAR	GETS MIN	PODS	MAXPC	DS	REPLICAS	AGE	
hpa-policy	Deployment/hpa-example	50%/50%	1	100	7	12m		
hpa-policy	Deployment/hpa-example	21%/50%	1	100	7	13m		
hpa-policy	Deployment/hpa-example	0%/50%	1	100	7	14m		
hpa-policy	Deployment/hpa-example	0%/50%	1	100	7	18m		
hpa-policy	Deployment/hpa-example	0%/50%	1	100	3	18m		
hpa-policy	Deployment/hpa-example	0%/50%	1	100	3	19m		
hpa-policy	Deployment/hpa-example	0%/50%	1	100	3	19m		
hpa-policy	Deployment/hpa-example	0%/50%	1	100	3	19m		
hpa-policy	Deployment/hpa-example	0%/50%	1	100	3	19m		
hpa-policy	Deployment/hpa-example	0%/50%	1	100	3	23m		
hpa-policy	Deployment/hpa-example	0%/50%	1	100	3	23m		
hpa-policy	Deployment/hpa-example	0%/50%	1	100	1	23m		

You can see that the CPU usage is 21% at 13m. The number of pods is reduced to 3 at 18m, and then reduced to 1 at 23m.

In the following output, you can see the workload scaling process and the time when the HPA policy takes effect.

kubectl describe deploy hpa-example

Events: Type Reason Age From Message Normal ScalingReplicaSet 25m deployment-controller Scaled up replica set hpa-example-79dd795485 to 1 Normal ScalingReplicaSet 20m deployment-controller Scaled up replica set hpa-example-79dd795485 to 4 Normal ScalingReplicaSet 16m deployment-controller Scaled up replica set hpa-example-79dd795485 to 7 Normal ScalingReplicaSet 6m28s deployment-controller Scaled down replica set hpaexample-79dd795485 to 3 Normal ScalingReplicaSet 72s deployment-controller Scaled down replica set hpaexample-79dd795485 to 1 # kubectl describe hpa hpa-policy Events:

 Type
 Reason
 Age
 From
 Message

 ---- ---- ---- ---- ----

Normal SuccessfulRescale 20m horizontal-pod-autoscaler New size: 4; reason: cpu resource utilization (percentage of request) above target

Normal SuccessfulRescale 16m horizontal-pod-autoscaler New size: 7; reason: cpu resource utilization (percentage of request) above target

Normal SuccessfulRescale 6m45s horizontal-pod-autoscaler New size: 3; reason: All metrics below target Normal SuccessfulRescale 90s horizontal-pod-autoscaler New size: 1; reason: All metrics below target

You can also view the HPA policy execution history on the console. Wait until the one node is reduced.

The reason why the other two nodes in the node pool are not reduced is that they both have pods in the **kube-system** namespace, and these pods are not created by DaemonSets. For details about the conditions in which nodes will not be removed, see **Node Scaling Mechanisms**.

----End

Summary

By using HPA and CA, auto scaling can be effortlessly implemented in various scenarios. Additionally, the scaling process of nodes and pods can be conveniently tracked.

7.2 Elastic Scaling of CCE Pods to CCI

The bursting add-on functions as a virtual kubelet to connect Kubernetes clusters to APIs of other platforms. This add-on is mainly used to extend Kubernetes APIs to serverless container services such as Huawei Cloud CCI.

With this add-on, you can schedule Deployments, StatefulSets, jobs, and CronJobs running in CCE clusters to CCI during peak hours. In this way, you can reduce consumption caused by cluster scaling.

Prerequisites

- Before using the add-on, go to the CCI console to grant CCI with the permissions to use CCE.
- VPC endpoints have been purchased for using the CCE Cloud Bursting Engine for CCI add-on. For details, see **Environment Configuration**.

Constraints

- Only CCE standard and CCE Turbo clusters that use the VPC network model are supported.
- The CCE Burst Elastic Engine (CCI) add-on earlier than v1.5.37 cannot be used in clusters of the Arm architecture. The pods for the add-on will not be scheduled on Arm nodes, if any, running in the cluster.
- The subnet where the cluster is located cannot overlap with 10.247.0.0/16, or the subnet conflicts with the Service CIDR block in the CCI namespace.
- Currently, Volcano Scheduler of 1.17.10 or earlier cannot be used to schedule pods with cloud storage volumes mounted to CCI.
- If the bursting add-on is used to schedule the pods to CCI 2.0, dedicated load balancers can be configured for ingresses and Services of the LoadBalancer type. The bursting add-on of a version earlier than 1.5.5 does not support Services of the LoadBalancer type.
- DaemonSets are not supported.
- Dynamic resource allocation is not supported, and related configurations are intercepted in plugin 1.5.27.
- After the add-on is installed, a namespace named **bursting**-{*Cluster ID*} will be created in CCI and managed by the add-on. Do not use this namespace when manually creating pods in CCI.

Installing the Add-on

- 1. Log in to the CCE console.
- 2. Click the name of the target CCE cluster to go to the cluster **Overview** page.
- 3. In the navigation pane, choose **Add-ons**.

- 4. Select the CCE Cloud Bursting Engine for CCI add-on and click Install.
- 5. Select the add-on version. The latest version is recommended.
- 6. On the Install Add-on page, configure the specifications as needed.
 - If you select **Preset**, the system will configure the number of pods and resource quotas for the add-on based on the preset specifications. You can see the configurations on the console.
 - If you select **Custom**, you can adjust the number of pods and resource quotas as needed. High availability is not possible with a single pod. If there is an error on the node where the add-on pod runs, the add-on will not function.

NOTE

- The CCE Cloud Bursting Engine for CCI add-on of v1.5.2 or later uses more node resources. You need to reserve sufficient pods that can be created on a node before upgrading the add-on. For details about the number of pods that can be created on a node, see Maximum Number of Pods That Can Be Created on a Node.
- The resource usages of the add-on vary depending on the workloads scaled to CCI. The resource requests and limits of the proxy, resource-syncer, and bursting-resource-syncer components are related to the maximum number of pods that can be scaled out. The resource requests and limits of the virtual-kubelet, bursting-virtual-kubelet, profile-controller, webhook, and bursting-webhook components are related to the maximum number of pods that can be created or deleted concurrently. For details about the recommended formulas for calculating the resource requests and limits of each component, see **Table 7-1**. P indicates the maximum number of pods that can be scaled out, and C indicates the maximum number of pods that can be created or deleted concurrently. You are advised to evaluate your service volume and select appropriate specifications.

Component	CPU Request (m)	CPU Limit (m)	Memory Request (MiB)	Memory Limit (MiB)
virtual- kubelet/ bursting- virtual- kubelet	(C + 400)/ 2,400 × 1,000	(C + 400)/600 × 1,000	(C + 400)/ 2,400 × 1,024	(C + 400)/300 × 1,024
profile- controller	(C + 1,000)/ 6,000 × 1,000	(C + 400)/ 1,200 × 1,000	(C + 1,000)/ 6,000 × 1,024	(C + 400)/ 1,200 × 1,024
proxy	(P + 2,000)/ 12,000 × 1,000	(P + 800)/ 2,400 × 1,000	(P + 2,000)/ 12,000 × 1,024	(P + 800)/ 2,400 × 1,024

Table 7-1 Formulas for calculating resource requests and limits of each component

Component	CPU Request (m)	CPU Limit (m)	Memory Request (MiB)	Memory Limit (MiB)
resource- syncer/ bursting- resource- syncer	(P + 800)/ 4,800 × 1,000	(P + 800)/ 1,200 × 1,000	(P + 800)/ 4,800 × 1,024	(P + 800)/600 × 1,024
webhook/ bursting- webhook	(C + 400)/ 2,400 × 1,000	(C + 400)/600 × 1,000	(C + 1,000)/ 6,000 × 1,024	(C + 400)/ 1,200 × 1,024

- (Optional) Networking: If this option is enabled, pods in the CCE cluster can communicate with pods in CCI through Services. The Proxy component will be automatically deployed upon add-on installation. For details, see Networking.
- 7. Configure the add-on parameters.
 - Subnet: Pods running workloads scheduled to CCI use IP addresses in the selected subnet. Plan the CIDR block to ensure IP address provisioning is not impacted.
 - **Enterprise Project**: Select an enterprise project.
- 8. Click Install.

Scaling Pods to CCI by Specifying a Profile on the Console

- 1. Log in to the CCE console and click the cluster name to go to the cluster console.
- 2. In the navigation pane, choose **Policies** > **CCI Scaling Policies**.
- 3. Click **Create CCI Scaling Policy** and configure the parameters.

Parameter	Description
Policy Name	Enter a policy name.
Namespace	Select the namespace where the scheduling policy applies. You can select an existing namespace or create a namespace. For details, see Creating a Namespace .
Workload	Enter a key and value or reference a workload label.

 Table 7-2 Parameters for creating a CCI scaling policy

Parameter	Description
Scheduling Policy	Select a scheduling policy.
	• Local priority scheduling: Pods will be preferentially scheduled to nodes in the current CCE cluster. When the node resources are insufficient, pods will be scheduled to CCI.
	• Force scheduling: All pods will be scheduled to CCI.
Scale to	Local: Set the current CCE cluster.
	• CCI : Set the maximum number of pods that can run on CCI.
Maximum Pods	Enter the maximum number of pods that can run in the CCE cluster or on CCI.
CCE Scale-in Priority	Value range: -100 to 100 . A larger value indicates a higher priority.
CCI Scale-in Priority	Value range: -100 to 100 . A larger value indicates a higher priority.
CCI Resource Pool	CCI 2.0 (bursting-node): next-generation serverless resource pool
	• CCI 1.0 (virtual-kubelet) : existing serverless resource pool, which will be unavailable soon.

4. Click OK.

Creating a Workload

- 1. Log in to the CCE console.
- 2. Click the name of the target CCE cluster to go to the cluster console.
- 3. In the navigation pane, choose **Workloads**.
- 4. Click Create Workload. For details, see Creating a Workload.
- Specify basic information. Select Force scheduling for Burst to CCI and then CCI 2.0 (bursting node) for CCI Resource Pool. For more information about scheduling policies, see Scheduling Workloads to CCI.

Basic Info				
Workload Type	Deployment StatefulSet DaemonSet Job Cron Job			
	Switching the workload type will require you to configure workload parameters again.			
Workload Name	bursting-test			
Namespace	default V Q Create Namespace			
Pods				
Burst to CCI	Disable scheduling Local priority scheduling Force scheduling			
	CCI Resource Pool CCI 2.0 (bursting-node) CCI 1.0 (virtual-kubelet)			
	Supports the rapid elastic creation of Pods to the cloud container instance CCI service in short-term high load scenarios to reduce consumption caused by cluster expansion.			

- 6. Configure the container parameters.
- 7. Click Create Workload.
- 8. On the **Workloads** page, click the name of the created workload to go to the workload details page.
- 9. View the node where the workload is running. If the workload is running on bursting-node, it has been scheduled to CCI.

	bursting-te	est 🗇									
	Workload Name	bursting-test					Namespace	default			
	Status	O Running					Created	34 seconds ago			
	Pods (Normal/All)	<u>1/1</u> &					Upgrade	Rolling upgrade			
	Container Runtime	runC					CCI elastic policy	Force scheduling			
	Description										
P	ods Access Mode	Containers	Auto Scaling	Scheduling Policies	Change History E	Events SecurityGroups	APM Settings				
	Dateta Exp	ort v									
	Q. Select a property or	r enter a keyword.									 O
	Pod Name 😣		Status 🖯	Namespace 🖯	Pod IP \varTheta	Node 😣	Restarts 🖯	CPU	Memory	Created 😣	Operation
	bursting-test-85d5	dbd84c-gbzvr	O Running	default	192 168.0.212	bursting-rode 🗹	٥	0.25 Cores 0.25 Cores 0%	512 MB 512 MB <u>0%</u>	34 seconds ago	Monitor More ~

Uninstalling the Add-on

- 1. Log in to the CCE console.
- 2. Click the name of the target CCE cluster to go to the cluster console.
- 3. In the navigation pane, choose **Add-ons**.
- 4. Select the **CCE Cloud Bursting Engine for CCI** add-on and click **Uninstall**.

	CCE Cloud Bursting Engine for CCI
	Preferred Recommendation
Version 1.5.1	5 Updated 5 days ago
ର୍ View Deta	ils Upgrade Edit Uninstall

Table 7-3 Special scenarios for uninstalling the add-on

Scenario	Symptom	Description
There are no nodes in the CCE cluster that the CCE Cloud Bursting Engine for CCI add-on needs to be uninstalled from.	Failed to uninstall the CCE Cloud Bursting Engine for CCI add-on.	If the CCE Cloud Bursting Engine for CCI add-on is uninstalled from the cluster, a job for clearing resources will be started in the cluster. To ensure that the job can be started, there is at least one node in the cluster that can be scheduled.

Scenario	Symptom	Description
The CCE cluster is deleted, but the CCE Cloud Bursting Engine for CCI add- on is not uninstalled.	There are residual resources in the namespace on CCI. If the resources are not free, additional expenditures will be generated.	The cluster is deleted, but the resource clearing job is not executed. You can manually clear the namespace and residual resources.

For more information about the add-on, see **CCE Cloud Bursting Engine for CCI**.

7.3 Auto Scaling Based on Prometheus Metrics

Kubernetes' default HPA policy only allows for auto scaling based on CPU and memory usage. However, in more complex service scenarios, this may not be sufficient to meet routine O&M requirements. To address this, CCE offers a Cloud Native Cluster Monitoring (kube-prometheus-stack) add-on that integrates into the open source Prometheus ecosystem. This add-on allows for monitoring of various components and provides multiple preset monitoring dashboards that are ready to use out-of-the-box. This document describes how to convert Huawei Cloud Prometheus metrics into metrics that can be used by HPA, providing a more convenient scaling mechanism for applications.

Prerequisites

- A cluster has been created.
- You can access the cluster using kubectl. For details, see **Connecting to a Cluster Using kubectl**.

Step 1: Install Cloud Native Cluster Monitoring

- **Step 1** Log in to the CCE console and click the cluster name to access the cluster console. In the navigation pane, choose **Add-ons**.
- Step 2 Locate the Cloud Native Cluster Monitoring add-on and click Install.

You are advised to focus on the following configurations, and adjust any other configurations as necessary. For details, see **Cloud Native Cluster Monitoring**.

- Local Data Storage: Enable this option to store the monitoring data in local storage. You can determine whether to report the monitoring data to AOM or a third-party monitoring platform.
- **Custom Metric Collection**: Enable this option in this practice. If this option is not enabled, custom metrics cannot be collected.

Step 3 Click Install.

----End

Step 2: Obtain Prometheus Monitoring Data

In this section, HPA is performed based on metrics related to pods, for example, pod metrics. You can also perform HPA based on metrics irrelevant to pods, for example, external load balancer metrics. For details, see **Auto Scaling Based on ELB Monitoring Metrics**.

The following describes how to deploy a **sample-app** application and expose the **container_memory_working_set_bytes_per_second** metric in Prometheus standard mode to identify the number of bytes per second in the working set of the container memory. For more information about Prometheus metrics, see **METRIC TYPES**.

Step 1 Deploy the test application.

1. Write a sample-app.yaml file. The file content is as follows:

apiVersion: apps/v1 kind: Deployment metadata: name: sample-app labels: app: sample-app spec: replicas: 1 selector: matchLabels: app: sample-app template: metadata: labels: app: sample-app spec: containers: - image: swr.cn-east-3.myhuaweicloud.com/container/autoscale-demo:v0.1.2 # Sample image name: metrics-provider resources: reauests: cpu: 250m memory: 512Mi limits: cpu: 250m memory: 512Mi ports: name: http containerPort: 8080 #Container port imagePullSecrets: - name: default-secret apiVersion: v1 kind: Service metadata: name: sample-app namespace: default labels: app: sample-app spec: ports: - port: 80 name: http protocol: TCP targetPort: 8080 selector: app: sample-app type: ClusterIP

D NOTE

The application exposes **container_memory_working_set_bytes_per_second**, which is used to check the working memory size of a container per second.

2. Create a workload. kubectl apply -f sample-app.yaml

Step 2 Create a ServiceMonitor to monitor custom metrics.

1.	Create a serv	/icemonitor.yaml file. The file content is as follows:
	apiVersion: moni kind: ServiceMon metadata: name: sample-a namespace: def	toring.coreos.com/v1 iitor app # ServiceMonitor name fault
	spec:	
	endpoints:	# Endpoints of the service to be monitored, including the name, port number, path,
	protocol, and mo	pre
	- interval: 30s monitored target	# Prometheus operator checks whether a service needs to be added to the c list every 30 seconds.
	port: http path: /metrics	
	namespaceSele	ctor:
	any: true	
	matchLabels:	-app # Label of the object whose data needs to be collected
_		
2.	Create Servic	eMonitor.

kubectl apply -f servicemonitor.yaml

----End

Step 3: Modify Configuration Files

Step 1 Modify Prometheus' adapter-config.

You can modify the **rules** field in **adapter-config** to convert the metrics exposed by Prometheus to metrics that can be associated with HPA.

kubectl -n monitoring edit configmap user-adapter-config

Step 2 Add a custom metric collection rule to the rules field. You can add multiple collection rules by adding multiple configurations under the rules field. For details, see Metrics Discovery and Presentation Configuration.

The following is an example of a custom collection rule: rules: - seriesQuery: container_memory_working_set_bytes{namespace!="",pod!=""} resources: overrides: namespace: resource: namespace pod: resource: pod name: matches: ^(.*)_bytes as: \${1}_bytes_per_second #The value of \${1} is the value that matches ^(.*) in matches:"^(.*)_bytes". metricsQuery: sum(<<.Series>>{<<.LabelMatchers>>}) by (<<.GroupBy>>)

- seriesQuery: indicates the PromQL request data. It specifies the metrics to be obtained by users. You can configure this parameter as needed.
- metricsQuery: aggregates the data requested by PromQL in seriesQuery.

- resources: specifies data label in PromQL, which is used to match resources. The resources here refer to the api-resource in a cluster, such as pods, namespaces, and nodes. You can run kubectl api-resources -o wide to check the resources. The key corresponds to LabelName in the Prometheus data. You have to ensure that the Prometheus metric data contains LabelName.
- name: indicates that Prometheus metric names are converted to readable metric names based on regular expression matching. In this example, container_memory_working_set_bytes is converted to container_memory_working_set_bytes_per_second.
- **Step 3** Redeploy the **custom-metrics-apiserver** workload in the **monitoring** namespace. kubectl -nmonitoring delete pod -l app=custom-metrics-apiserver
- **Step 4** Run the following command to check whether the metric is added: kubectl get --raw /apis/custom.metrics.k8s.io/v1beta1/namespaces/default/pods/*/ container_memory_working_set_bytes_per_second



----End

Step 4: Create an HPA Policy

Step 1 Create an HPA policy using custom metrics.

1.	Create an hpa.yaml file. The file content is as follows:
	kind: HorizontalPodAutoscaler apiVersion: autoscaling/v2
	metadata:
	name: sample-app-memory-high
	# Description of an HPA object. HPA dynamically changes the number of pods of the object. scaleTargetRef: apiVersion: apps/v1
	kind: Deployment
	# Minimum number of pods and maximum number of pods of the HPA minReplicas: 1 maxReplicas: 10
	#Monitored metric array. Multiple types of metrics can coexist.
	metrics:
	- type: Pods
	pods:
	name: container_memory_working_set_bytes_per_second # Use the custom container metrics. target:
	type: AverageValue # Target value of the AverageValue type. For the pod metrics, only the target values of the AverageValue type are supported. averageValue: 1024000m # 1024000m specifies 1 KB.
2.	Create the HPA policy.
	kubectl apply -f hpa.yaml



kubectl get hpa sample-app-memory-high



The number of replicas has been increased from 1 to 10.

----End

7.4 Auto Scaling Based on ELB Monitoring Metrics

Background

By default, Kubernetes scales a workload based on resource usage metrics such as CPU and memory. However, this mechanism cannot reflect the real-time resource usage when traffic bursts arrive, because the collected CPU and memory usage data lags behind the actual load balancer traffic metrics. For some services (such as flash sale and social media) that require fast auto scaling, scaling based on this rule may not be performed in a timely manner and cannot meet these services' actual needs. In this case, auto scaling based on ELB QPS data can respond to service requirements more timely.

Solution

This section describes an auto scaling solution based on ELB monitoring metrics. Compared with CPU/memory usage-based auto scaling, auto scaling based on ELB QPS data is more targeted and timely.

The key of this solution is to obtain the ELB metric data and report the data to Prometheus, convert the data in Prometheus to the metric data that can be identified by HPA, and then perform auto scaling based on the converted data.

The implementation scheme is as follows:

- 1. Develop a Prometheus exporter to obtain ELB metric data, convert the data into the format required by Prometheus, and report it to Prometheus. This section uses **cloudeye-exporter** as an example.
- 2. Convert the Prometheus data into the Kubernetes metric API for the HPA controller to use.
- 3. Set an HPA rule to use ELB monitoring data as auto scaling metrics.



Figure 7-2 ELB traffic flows and monitoring data

NOTE

Other metrics can be collected in the similar way.

Prerequisites

- You are familiar with Prometheus.
- You have installed the Cloud Native Cluster Monitoring add-on (kubeprometheus-stack) of version 3.10.1 or later in the cluster.

NOTE

Local Data Storage must be enabled for Data Storage Configuration.

Building an Exporter Image

This section uses **cloudeye-exporter** to monitor load balancer metrics. To develop an exporter, see **Appendix: Developing an Exporter**.

Step 1 Log in to a VM that can access the Internet and has Docker installed and write a Dockerfile.

vi Dockerfile

```
The content is as follows:

RUN apt-get update \

&& apt-get install -y wget ca-certificates \

&& update-ca-certificates \

&& wget https://github.com/huaweicloud/cloudeye-exporter/releases/download/v2.0.16/cloudeye-

exporter-v2.0.16.tar.gz \

&& tar -xzvf cloudeye-exporter-v2.0.16.tar.gz

CMD ["./cloudeye-exporter -config=/tmp/clouds.yml"]
```

cloudeye-exporter of **v2.0.16** is used as an example. For more versions, see **Releases**.

Step 2 Build an image. The image name is **cloudeye-exporter** and the image tag is 2.0.16.

docker build --network host . -t cloudeye-exporter:2.0.16

Step 3 Push the image to SWR.

- (Optional) Log in to the SWR console, choose Organizations in the navigation pane, and click Create Organization in the upper right corner. Skip this step if you already have an organization.
- 2. In the navigation pane, choose **My Images** and then click **Upload Through Client**. On the page displayed, click **Generate a temporary login command**

and click \square to copy the command.

- 3. Run the login command copied in the previous step on the cluster node. If the login is successful, the message "Login Succeeded" is displayed.
- 4. Tag the **cloudeye-exporter** image.

docker tag {*Image name 1:Tag 1*}*|*{*Image repository address*}*|*{*Organization name*}*|*{*Image name 2:Tag 2*}

- *{Image name 1:Tag 1}*: name and tag of the local image to be uploaded.
- *{Image repository address}*: The domain name at the end of the login command in Step 3.2 is the image repository address, which can be obtained on the SWR console.
- *{Organization name}*: name of the organization created in **Step 3.1**.
- *{Image name 2:Tag 2}*: desired image name and tag to be displayed on the SWR console.

The following is an example:

docker tag cloudeye-exporter:2.0.16 swr.apsoutheast-3.myhuaweicloud.com/container/cloudeye-exporter:2.0.16

5. Push the image to the image repository.

docker push {*Image repository address*}*|*{*Organization name*}*|*{*Image name 2:Tag 2*}

The following is an example:

docker push swr.ap-southeast-3.myhuaweicloud.com/container/cloudeye-exporter:2.0.16

The following information will be returned upon a successful push:

```
030***: Pushed
```

```
2.0.16: digest: sha256:eb7e3bbd*** size: **
```

To view the pushed image, go to the SWR console and refresh the **My Images** page.

----End

Deploying the Exporter

Prometheus can dynamically monitor pods if you add Prometheus annotations to the pods (the default path is **/metrics**). This section uses **cloudeye-exporter** as an example.

Common annotations in Prometheus are as follows:

- prometheus.io/scrape: If the value is true, the pod will be monitored.
- **prometheus.io/path**: URL from which the data is collected. The default value is **/metrics**.
- prometheus.io/port: port number of the endpoint to collect data from.
- **prometheus.io/scheme**: Defaults to **http**. If HTTPS is configured for security purposes, change the value to **https**.
- **Step 1** Use kubectl to connect to the cluster.

Step 2 Create a secret, which will be used by **cloudeye-exporter** for authentication.

1. Create the **clouds.yml** file with the following content:

```
global:

prefix: "huaweicloud"

scrape_batch_size: 300

auth:

auth_url: "https://iam.ap-southeast-3.myhuaweicloud.com/v3"

project_name: "ap-southeast-3"

access_key: "********"

secret_key: "********"

region: "ap-southeast-3"
```

Parameters in the preceding content are described as follows:

- auth_url: indicates the IAM endpoint, which can be obtained from Regions and Endpoints.
- project_name: indicates the project name. On the My Credential page, view the project name and project ID in the Projects area.
- access_key and secret_key: You can obtain them from Access Keys.
- region: indicates the region name, which must correspond to the project in project_name.
- 2. Obtain the Base64-encrypted string of the preceding file. cat clouds.yml | base64 -w0 ;echo

Information similar to the following is displayed: ICAga*****

3. Create the **clouds-secret.yaml** file with the following content: apiVersion: v1

```
kind: Secret
data:
clouds.yml: ICAga***** # Replace it with the Base64-encrypted string.
metadata:
annotations:
description: "
name: 'clouds.yml'
namespace: default #Namespace where the secret is in, which must be the same as the
deployment's namespace.
labels: {}
type: Opaque
```

- 4. Create a secret. kubectl apply -f clouds-secret.yaml
- Step 3 Create the cloudeye-exporter-deployment.yaml file with the following content:

kind: Deployment apiVersion: apps/v1 metadata: name: cloudeye-exporter namespace: default spec: replicas: 1 selector:

```
matchLabels:
   app: cloudeye-exporter
   version: v1
 template:
  metadata:
   labels:
     app: cloudeye-exporter
     version: v1
  spec:
   volumes:
     - name: vol-166055064743016314
      secret:
       secretName: clouds.yml
       defaultMode: 420
   containers:
     - name: container-1
      image: swr.ap-southeast-3.myhuaweicloud.com/container/cloudeye-exporter:2.0.16 # exporter
image address built above
      command:
       - ./cloudeye-exporter # Startup command for building the cloudeye-exporter image
       - '-config=/tmp/clouds.yml'
      resources: {}
      volumeMounts:
        - name: vol-166055064743016314
         readOnly: true
         mountPath: /tmp
      terminationMessagePath: /dev/termination-log
      terminationMessagePolicy: File
      imagePullPolicy: IfNotPresent
    restartPolicy: Always
   terminationGracePeriodSeconds: 30
   dnsPolicy: ClusterFirst
   securityContext: {}
   imagePullSecrets:
     - name: default-secret
   schedulerName: default-scheduler
 strategy:
  type: RollingUpdate
  rollingUpdate:
   maxUnavailable: 25%
   maxSurge: 25%
 revisionHistoryLimit: 10
```

progressDeadlineSeconds: 600

Create the preceding workload.

kubectl apply -f cloudeye-exporter-deployment.yaml

Step 4 Create the cloudeye-exporter-service.yaml file.

```
apiVersion: v1
kind: Service
metadata:
 name: cloudeye-exporter
 namespace: default
 labels:
  app: cloudeye-exporter
  version: v1
 annotations:
  prometheus.io/port: '8087'
                                 #Port number of the endpoint to collect data from.
  prometheus.io/scrape: 'true' #If it is set to true, the resource is the monitoring target.
  prometheus.io/path: "/metrics" #URL from which the data is collected. The default value is /metrics.
  prometheus.io/scheme: "http" #The default value is http. If https is set for security purposes, you need
to change it to https.
spec:
 ports:
  - name: cce-service-0
    protocol: TCP
    port: 8087
    targetPort: 8087
 selector:
```

app: cloudeye-exporter version: v1 type: ClusterIP

Create the preceding Service.

kubectl apply -f cloudeye-exporter-service.yaml

----End

Interconnecting with Prometheus

After collecting monitoring data, Prometheus needs to convert the data into the Kubernetes metric API for the HPA controller to perform auto scaling.

In this example, the ELB metrics associated with the workload need to be monitored. Therefore, the target workload must use the Service or ingress of the **LoadBalancer** type.

- **Step 1** View the access mode of the workload to be monitored and obtain the ELB listener ID.
 - On the CCE cluster console, choose Networking. On the Services or Ingresses tab page, view the Service or ingress of the LoadBalancer type and click the load balancer to access the load balancer page.

Services	Ingresses			
Q Selec	Export V	Quotas (Remaining/Total): Load	Balancer (199 /200) Listener (299 /3	000) Backend Server (996 /1,000)
Serv	vice \ominus	Selector	Namespace 🔶	Service Type \ominus
v o 1	nginx	app nginx version	v1 default	LoadBalancer

2. On the **Listeners** tab, view the listener corresponding to the workload and copy the listener ID.

C Elastic Load Balancer / Load balancer (Source View NEW Summary Listeners	Monitoring Access Logs Tag	35		@ Feedback	Add Listener	Vew Backe	nd Server Group	Create Backend Server Group
Add Listener Export ~								00
<table-cell> NamelID 🛞</table-cell>	Frontend Protocol/Port	Default Backend Server Group	Forwarding Policies 😣	Health Check	Monit	Certifi 😣	Access Control	Operation
K8s_TCP_80 376211cb-9014-462a-b696-b5d8906ed	TCP/80	k8s_default_nginx-80_TCP-80 View/Add Backend Server		O Healthy Configure	Θ	-	Al IP addresses Configure	Edit Delete
Total Records: 1 Selected: 1								10 ~ (1) >

- **Step 2** Use kubectl to connect to the cluster and add Prometheus configurations. In this example, collect load balancer metrics. For details about advanced usage, see **Configuration**.
 - 1. Create the **prometheus-additional.yaml** file, add the following content to the file, and save the file:
 - job_name: elb_metric params: services: ['SYS.ELB'] kubernetes_sd_configs:
 - role: endpoints relabel_configs:
 - action: keep regex: '8087' source_labels:
 - __meta_kubernetes_service_annotation_prometheus_io_port

```
- action: replace
 regex: ([^:]+)(?::\d+)?;(\d+)
 replacement: $1:$2
 source labels:
    address
     meta kubernetes service annotation prometheus io port
target_label: __address_
- action: labelmap
regex: __meta_kubernetes_service_label_(.+)
- action: replace
source_labels:
    __meta_kubernetes_namespace
target_label: kubernetes_namespace
- action: replace
 source_labels:
   - __meta_kubernetes_service_name
target_label: kubernetes_service
```

2. Use the preceding configuration file to create a secret named **additional**-**scrape-configs**.

kubectl create secret generic *additional-scrape-configs* --from-file *prometheus-additional.yaml* -n monitoring --dry-run=client -o yaml | kubectl apply -f -

 Edit the persistent-user-config configuration item to enable AdditionalScrapeConfigs. kubectl edit configmap persistent-user-config -n monitoring

Add *--common.prom.default-additional-scrape-configs-key=prometheus-additional.yaml* under **operatorConfigOverride** to enable AdditionalScrapeConfigs as follows:

```
data:

lightweight-user-config.yaml: |

customSettings:

additionalScrapeConfigs: []

agentExtraArgs: []

metricsDeprecated:

globalDeprecateMetrics: []

nodeExporterConfigOverride: []

operatorConfigOverride:

- --common.prom.default-additional-scrape-configs-key=prometheus-additional.yaml
```

4. Go to Prometheus to check whether custom metrics are successfully collected.

Step 3 Modify the user-adapter-config configuration item.

kubectl edit configmap user-adapter-config -nmonitoring

```
Add the following content to the rules field, replace lbaas_listener_id with the
listener ID obtained in Step 1, and save the file.
apiVersion: v1
data:
 config.yaml: |-
  rules:
  - metricsQuery: sum(<<.Series>>{<<.LabelMatchers>>,lbaas_listener_id="*****"}) by (<<.GroupBy>>)
   resources:
     overrides:
      kubernetes namespace:
       resource: namespace
      kubernetes_service:
       resource: service
    name:
     matches: huaweicloud_sys_elb_(.*)
     as: "elb01_${1}"
   seriesQuery: '{lbaas_listener_id="*****"}'
```



(D) Damping							
0.00000	Deployments StatefulSets Daemon	Sets Jobs Cron Jobs	Pods			Quick Links	Greate from YAML Create Works
Kubernetes Resources							
R Workloads	Ublette Batch Hedeproy	xport V					
3 Services & Ingresses	 Select a property or enter a keyword. 						() (O
Storage	Workload Name 🕀	Status	Pods (Normal/AII) 🖯	Namespace 🖯	Created 🖯	Image Name \varTheta	Operation
D ConfigMaps and Se							
App Templates	custom-metrics-apiserver	 Running 	1/1	monitoring	2 minutes ago	k8s-prometheus-adapter 3.12.0	Monitor View Log Mate A
Custom Resources							Auto Scaling
) Namespaces	kube-state-metrics	 Running 	1/.1	monitoring	45 minutes ago	🍲 kube-state-metrics 3 12.0	Monitor View YAML
							Relate
	promethous-operator	O Running	1/1	monitoring	45 minutes ago	prometheus-operator 3 12 0	Monitor view Log incre +
, Nodes							
Settings	Total Descenter 3						10 ··· / /
 onner opgråde 	Total Necurus: a						10 🗸 🤇 1

----End

Creating an HPA Policy

After the data reported by the exporter to Prometheus is converted into the Kubernetes metric API by using the Prometheus adapter, you can create an HPA policy for auto scaling.

Step 1 Obtain the name of a custom metric in the cluster. m7_in_Bps (inbound traffic rate) is used as an example. For details about other ELB metrics, see ELB Listener Metrics.

kubectl get --raw="/apis/custom.metrics.k8s.io/v1beta1" |grep m7_in_Bps

Information similar to that in the figure below is displayed.



Step 2 Create an HPA policy. The inbound traffic of the ELB load balancer is used to trigger scale-out. When the value of **m7_in_Bps** (inbound traffic rate) exceeds 1000, the nginx Deployment will be scaled.

apiVersion: autoscaling/v2 kind: HorizontalPodAutoscaler
metadata:
name: noinx
namesnace: default
Spec:
scaleTargetRef:
apiVersion: apps/v1
kind: Deployment
name: nginx
minReplicas: 1
maxReplicas: 10
metrics:
- type: Object
object:
metric:
name: <i>elb01_m7_in_Bps</i> # Name of the custom monitoring metric obtained in the previous step
describedObject:
apiVersion: v1
kind: Service
name: cloudeye-exporter
target:
type: Value
value: 1000

Figure 7-3 Created HPA Policy

0.0									
(i) Overview	Scaling Policies Node S	scaling Policies							Create from YAML Create HPA Policy
Kubernetes Resources	HPA Policies CronHPA Po	licies CustomedHPA Polic	ies						
🚝 Workloads									
Gervices & Ingresses									
🖺 Storage	 To create or modify a particular 	policy, go to Workloads > Auto Sc	aing.						
D ConfigMaps and Se	Export V								
Q Policies									
App Templates	C. Select an attribute or er	ther a keyword to search for polici	65.						Q 🔘
S Custom Resources	Policy Name 🖯	Latest Status 😑	Associated Workload Θ	Rule (Metric Expected Value Tolerance Range) 🖯	Pod Range 🖯	Scaling Behavior Θ	Namespace Θ	Created \varTheta	Operation
Namespaces	nginx	 Started 	ngirox	elb01_m7_in_Bps(cloudeye-exporter (Service)) Value: 1k	1 - 10	Default	default	3 minutes ago	View Events Edit YAML More 🛩
Cluster									
"Å, Nodes	Total Records: 1								10 ~ < 1 >

Step 3 After the HPA policy is created, perform a pressure test on the workload (accessing the pods through ELB). Then, the HPA controller determines whether scaling is required based on the configured value.

In the **Events** dialog box, obtain scaling records in the **Kubernetes Event** column.

vent						
Real-Time Events	Persistent Even	ts				
 Event data is retain 	ed for 1 hour and the	en automatically	deleted. To enable e	vent persistence, go to Logging		
Export ~						
Q Select a property or	enter a keyword.					0
Kubernetes Co…	Type 🔤	Occur	New size: 10; reas	on: Service metric elb01_m7_in_	Bps above target	Last Occurred
horizontal-pod-autos	 Normal 	1	Normal	New size: 10; re 🗍	Apr 22, 2025 22:10:5	Apr 22, 2025 22:10:5.
horizontal-pod-autos	 Normal 	1	Normal	New size: 8; rea	Apr 22, 2025 22:10:4	Apr 22, 2025 22:10:4.
horizontal-pod-autos	 Normal 	1	Normal	New size: 4; rea	Apr 22, 2025 22:10:2	Apr 22, 2025 22:10:2.
horizontal-pod-autos	 Normal 	1	Normal	New size: 1; rea	Apr 22, 2025 22:09:4	Apr 22, 2025 22:09:4.
horizontal-pod-autos	 Normal 	1	Normal	New size: 4; rea	Apr 22, 2025 22:06:4	Apr 22, 2025 22:06:4.
otal Records: 5						10 ~ < 1

Figure 7-4 Scaling events

----End

ELB Listener Metrics

The following table lists the ELB listener metrics that can be collected using the method described in this section.

Table 7-4 ELB listener metrics

Metric	Name	Unit	Description
m1_cps	Concurrent Connections	Count	Number of concurrent connections processed by a load balancer.

Metric	Name	Unit	Description
m1e_server_r ps	Reset Packets from Backend Servers	Count/ Second	Number of reset packets sent from the backend server to clients. These reset packages are generated by the backend server and then forwarded by load balancers.
m1f_lvs_rps	Reset Packets from Load Balancers	Count/ Second	Number of reset packets sent from load balancers.
m21_client_rp s	Reset Packets from Clients	Count/ Second	Number of reset packets sent from clients to the backend server. These reset packages are generated by the clients and then forwarded by load balancers.
m22_in_band width	Inbound Bandwidth	bit/s	Inbound bandwidth of a load balancer.
m23_out_ban dwidth	Outbound Bandwidth	bit/s	Outbound bandwidth of a load balancer.
m2_act_conn	Active Connections	Count	Number of current active connections.
m3_inact_con n	Inactive Connections	Count	Number of current inactive connections.
m4_ncps	New Connections	Count	Number of current new connections.
m5_in_pps	Incoming Packets	Count	Number of packets sent to a load balancer.
m6_out_pps	Outgoing Packets	Count	Number of packets sent from a load balancer.
m7_in_Bps	Inbound Rate	byte/s	Number of incoming bytes per second on a load balancer.
m8_out_Bps	Outbound Rate	byte/s	Number of outgoing bytes per second on a load balancer.

Appendix: Developing an Exporter

Prometheus periodically calls the **/metrics** API of the exporter to obtain metric data. Applications only need to report monitoring data through **/metrics**. You can select a Prometheus client in a desired language and integrate it into applications to implement the **/metrics** API. For details about the client, see **Prometheus CLIENT LIBRARIES**. For details about how to write the exporter, see **WRITING EXPORTERS**.

The monitoring data must be in the format that Prometheus supports. Each data record provides the ELB ID, listener ID, namespace where the Service is located, Service name, and Service UID as labels, as shown in the following figure.

HELP ml cps Number of concurrent connections.
TYPE m1 cps gauge
n1 cps{lb_instance id="eab8f0fd-9997-468c-8ce2-bdaeee416dc5", lb listener id="929747a9-ba55-472b-a1e4-8b8d6e076054", namespace="default", service name="nginx", uid="3b74f807-addf-11e9-bccb-fa163ea2c926"} 0
HELP m5 in pps the packets count that are currently flowing into.
∉ TYPE m5 in pps gauge
n5 in pps{lb instance id="eab8f6fd-9997-468c-8ce2-bdaeee416dc5", lb listener id="929747a9-ba55-472b-a1e4-8b8d6e076054", namespace="default", service name="nginx", uid="3b74f807-addf-11e9-bccb-fa163ea2c926"} 0
HELP m6_out_pps the packets count that are currently flowing out.
¢ TYPE m6_out_pps gauge
n6_out_pps{lb_instance_id="eabBf0fd-9997-468c-8ce2-bdaeee416dc5", lb_listener_id="929747a9-ba55-472b-a1e4-8b8d6e076054", namespace="default", service_name="nginx", uid="3b74f807-addf-11e9-bccb-fa163ea2c926"} 0
HELP m7_in_Bps network traffic flowing into the measurement object per second.
¢ TYPE π7_in_Bps gauge
n7_in_Bps{lb_instance_id="eab8f6fd-9997-468c-8ce2-bdaeee416dc5",lb_listener_id="929747a9-ba55-472b-a1e4-8b8d6e076054",namespace="default",service_name="nginx",uid="3b74f807-addf-11e9-bccb-fa163ea2c926"} 0
HELP m8_out_Bps network traffic flowing out of the measurement object per second.
TYPE m8_out_Bps gauge
n8_out_Bps{lb_instance_id=*eab8f0fd-9997-468c-8ce2-bdaeee416dc5", lb_listener_id="929747a9-ba55-472b-a1e4-8b8d6e076054", namespace=*default", service_name="nginx", uid="3b74f807-addf-11e9-bccb-fa163ea2c926"} 0

To obtain the preceding data, perform the following operations:

Step 1 Obtain all Services.

The **annotations** field in the returned information contains the ELB associated with the Service.

- kubernetes.io/elb.id
- kubernetes.io/elb.class
- **Step 2** Use APIs in **Querying Listeners** to get the listener ID based on the load balancer ID obtained in the previous step.
- **Step 3** Obtain the ELB monitoring data.

The ELB monitoring data is obtained using the CES APIs described in **Querying Monitoring Data of Multiple Metrics**. For details about ELB monitoring metrics, see **Monitoring Metrics**. Example:

- **m1_cps**: number of concurrent connections
- m5_in_pps: number of incoming data packets
- **m6_out_pps**: number of outgoing data packets
- **m7_in_Bps**: incoming rate
- **m8_out_Bps**: outgoing rate
- **Step 4** Aggregate data in the format that Prometheus supports and expose the data through the **/metrics** API.

The Prometheus client can easily call the **/metrics** API. For details, see **CLIENT LIBRARIES**. For details about how to develop an exporter, see **WRITING EXPORTERS**.

----End

7.5 Auto Scaling of Multiple Applications Using Nginx Ingresses

Deploying applications in multiple pods in a production environment can enhance their stability and reliability, but it can also lead to increased resource waste and costs. To strike a balance between resource utilization and application performance, manually adjusting the number of pods may not be efficient or effective.

However, if the application uses Nginx ingresses to route and forward external traffic, you can configure HPA policies using the

nginx_ingress_controller_requests metric. This allows for dynamic adjustment of pods based on traffic changes, optimizing resource utilization.

Prerequisites

- The NGINX Ingress Controller add-on has been installed in the cluster.
- You have installed the Cloud Native Cluster Monitoring add-on in the cluster and enabled **Local Data Storage** for the add-on.
- You have connected the cluster with the kubectl command line tool or CloudShell.
- The pressure testing tool Apache Benchmark has been installed.

Creating a Workload and a Service for the Workload

This section provides an example of how to route external traffic for two Services using Nginx ingresses.

Step 1 Create a **test-app** workload and a Service for it.

1.	Write a test-app.yaml file.
	apiVersion: apps/v1
	kind: Deployment
	metadata:
	name: test-app
	labels:
	app: test-app
	spec:
	replicas: 1
	selector:
	matchLabels:
	app: test-app
	template:
	metadata:
	labels:
	app: test-app
	spec:
	containers:
	 image: skto/sample-app:v2
	name: metrics-provider
	ports:
	- name: http
	containerPort: 8080
	apiVersion: v1
	kind: Service
	metadata:
	name: test-app
	namespace: default
	labels:
	app: test-app
	spec:
	ports:
	- port: 8080
	name: http
	protocol: TCP
	LargetPort: 8080
	Selector.
	app. test-app
	type: Clusterip

2. Deploy the **test-app** workload and the corresponding Service. kubectl apply -f test-app.yaml

Step 2 Create a **sample-app** workload and a Service for it.

1. Write a **sample-app.yaml** file.

apiVersion: apps/v1 kind: Deployment metadata: name: sample-app labels: app: sample-app spec: replicas: 1 selector: matchLabels: app: sample-app template: metadata: labels: app: sample-app spec: containers: - image: skto/sample-app:v2 name: metrics-provider ports: - name: http containerPort: 8080 apiVersion: v1 kind: Service metadata: name: sample-app namespace: default labels: app: sample-app spec: ports: - port: 80 name: http protocol: TCP targetPort: 8080 selector: app: sample-app type: ClusterIP

2. Deploy the **sample-app** workload and the corresponding Service. kubectl apply -f sample-app.yaml

Step 3 Deploy an ingress.

1. Write an **ingress.yaml** file. apiVersion: networking.k8s.io/v1 kind: Ingress metadata: name: test-ingress namespace: default spec: ingressClassName: nginx rules: - host: test.example.com http: paths: - backend: service: name: sample-app port: number: 80 path: / pathType: ImplementationSpecific - backend: service: name: test-app

port: number: 8080 path: /home pathType: ImplementationSpecific

- host: specifies the Service access domain name. In this example, test.example.com is used.
- path: specifies the URL to be accessed. After receiving a request, the system matches the request with the corresponding Service based on the routing rules and accesses the corresponding pod through the Service.
- backend: consists of the Service name and Service port and specifies the Service forwarded by the current path.
- 2. Deploy an ingress. kubectl apply -f ingress.yaml
- 3. Obtain an ingress.



4. After the deployment is successful, log in to the target node and add the service domain name and the IP address of the load balancer associated with NGINX Ingress Controller to the local **hosts** file of the node. The IP address of the load balancer associated with NGINX Ingress Controller is that obtained in **Step 3.3**.
export NGINXELB=xx.xx.xx

echo -n "\${NGINXELB} test.example.com" >> /etc/hosts

5. Log in to the cluster node and access the host address through the / and / home paths.

NGINX Ingress Controller accesses **sample-app** and **test-app** based on the preceding configurations.

curl test.example.com/ Hello from '/' path!

curl test.example.com/home Hello from '/home' path!

----End

Modifying user-adapter-config in Prometheus

- Step 1 Run the following command to edit user-adapter-config: kubectl -n monitoring edit configmap user-adapter-config
- **Step 2** Add the following rules to the ConfigMap of the adapter:

```
apiVersion: v1
```

```
data:

config.yaml: |

rules:

- metricsQuery: sum(rate(<<.Series>>{<<.LabelMatchers>>}[2m])) by (<<.GroupBy>>)

name:

as: ${1}_per_second

matches: ^(.*)_requests

resources:

namespaced: false

overrides:

exported_namespace:

resource: namespace

service:
```

resource: service
seriesQuery: nginx_ingress_controller_requests
resourceRules:
cpu:
containerQuery: sum(rate(container_cpu_usage_seconds_total{<<.LabelMatchers>>,container!="",pod!
=""}[1m])) by (<<.GroupBy>>)
nodeQuery: sum(rate(container_cpu_usage_seconds_total{<<.LabelMatchers>>, id='/'}[1m])) by
(<<.GroupBy>>)
resources:
overrides:
instance:
resource: node
namespace:
resource: namespace
pod:
resource: pod
containerLabel: container
memory:
containerQuery: sum(container_memory_working_set_bytes{<<.LabelMatchers>>,container!="",pod!
=""}) by (<<.GroupBy>>)
nodeQuery: sum(container_memory_working_set_bytes{<<.LabelMatchers>>,id='/'}) by (<<.GroupBy>>)
resources:
overrides:
instance:
resource: node
namespace:
resource: namespace
pod:
resource: pod
containerLabel: container

Step 3 Restart custom-metrics-apiserver.

kubectl -n monitoring delete pod -l app=custom-metrics-apiserver

Step 4 Log in to the cluster node and access the host address through the / and /home paths for multiple times.

curl test.example.com/ Hello from '/' path!

curl test.example.com/home Hello from '/home' path!

Step 5 Run the following command to check whether the metric is added:

kubectl get --raw /apis/custom.metrics.k8s.io/v1beta1/namespaces/default/services/*/ nginx_ingress_controller_per_second | python -m json.tool



----End

Creating an HPA Policy

Step 1 Create an **hpa.yaml** file and configure auto scaling for the **test-app** and **sample-app** workloads based on Prometheus metrics.

apiVersion: autoscaling/v2 kind: HorizontalPodAutoscaler metadata: name: sample-hpa # HPA name spec: scaleTargetRef: apiVersion: apps/v1 kind: Deployment name: sample-app # Deployment name minReplicas: 1 # Minimum number of pods maxReplicas: 10 # Maximum number of pods metrics: - type: Object object: metric: name: nginx_ingress_controller_per_second # Metric describedObject: apiVersion: v1 kind: service name: sample-app # Service of the Deployment target: type: Value value: 30 # Scaling is triggered when the metric value is within the range of $(Actual value/30)\pm 0.1$. apiVersion: autoscaling/v2 kind: HorizontalPodAutoscaler metadata: name: test-hpa spec: scaleTargetRef: apiVersion: apps/v1 kind: Deployment name: test-app minReplicas: 1 maxReplicas: 10 metrics: - type: Object object: metric: name: nginx_ingress_controller_per_second describedObject: apiVersion: v1 kind: service name: test-app target: type: Value value: 30

Step 2 Deploy the HPA policy.

kubectl apply -f hpa.yaml

Step 3 Check the HPA deployment.

kubectl get hpa

user@lvb6mb5e3expf6p-machine:~\$ kubect1 get hpa							
NAME	REFERENCE	TARGETS	MINPODS	MAXPODS	REPLICAS	AGE	
sample-hpa	Deployment/sample-app	0/30	1	10	1	171m	
test-hpa	Deployment/test-app	0/30	1	10	1	171m	
user@lvb6mb5e3expf6p-machine:~\$							

----End

Verifying Scaling

Step 1 Log in to the target cluster node and perform a pressure testing on the **/home** path.

ab -c 50 -n 5000 test.example.com/home

Step 2 Check the HPA.

kubectl get hpa



- **Step 3** Log in to the target cluster node and perform a pressure testing on the root path. ab -c 50 -n 5000 test.example.com/
- **Step 4** Check the HPA.

kubectl get hpa



Compared with the HPA metrics obtained before the pressure testing, the service application is scaled out when the number of requests exceeds the threshold.

----End
8 Monitoring

8.1 Monitoring Multiple Clusters Using Prometheus

Application Scenarios

Generally, a user has different clusters for different purposes, such as production, testing, and development. To monitor, collect, and view metrics of these clusters, you can deploy a set of Prometheus.

Solution Architecture

Multiple clusters are connected to the same Prometheus monitoring system, as shown in the following figure. This reduces maintenance and resource costs and facilitates monitoring information aggregation.



Prerequisites

- The target cluster has been created.
- Prometheus has been properly connected to the target cluster.
- Prometheus has been installed on a Linux host using a binary file. For details, see Installation.

Procedure

Step 1 Obtain the **bearer_token** information of the target cluster.

1. Create the RBAC permission in the target cluster.

```
Log in to the background node of the target cluster and create the
prometheus_rbac.yaml file.
apiVersion: v1
kind: ServiceAccount
metadata:
name: prometheus-test
namespace: kube-system
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
name: prometheus-test
rules:
- apiGroups:
- ""
resources:
- nodes
- services
- endpoints
- pods
 - nodes/proxy
verbs:
 - get
- list
 - watch
- apiGroups:
 - "extensions"
resources:
 - ingresses
verbs:
 - get
- list
- watch
- apiGroups:
 _ ....
resources:
 - configmaps
- nodes/metrics
verbs:
- get
- nonResourceURLs:
 - /metrics
verbs:
- get
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
name: prometheus-test
roleRef:
apiGroup: rbac.authorization.k8s.io
kind: ClusterRole
```

name: prometheus-test subjects: - kind: ServiceAccount

name: prometheus-test namespace: kube-system

Run the following command to create the RBAC permission:

kubectl apply -f prometheus_rbac.yaml

2. Obtain the **bearer_token** information of the target cluster.

NOTE

 In clusters earlier than v1.21, tokens are obtained by mounting the secret of a service account to a pod. Such tokens are permanent. However, this approach is not recommended in clusters of v1.21 or later. Starting from v1.25, Kubernetes no longer automatically creates secrets for service accounts as part of its community iteration policy.

Instead, in clusters of v1.21 and later, the recommended approach is to use the **TokenRequest** API to **obtain tokens** and mount them via a projected volume to pods. These tokens remain valid only for a fixed period and become invalid once the pods are deleted. For details, see **Service Account Token Security Improvement**.

 If you need a token that never expires, you can manually manage secrets for service accounts. Although a permanent service account token can be created manually, you are advised to use a short-lived token by calling the TokenRequest API for better security.

Obtain the **serviceaccount** information.

kubectl describe sa prometheus-test -n kube-system

🧴 🛶 มารถกอก กฎสภา	■■■kj ~]# kubectl describe sa	prometheus-test	-n kube-system
Name:	prometheus-test		
Namespace:	kube-system		
Labels:	<none></none>		
Annotations:	<none></none>		
Image pull secrets:	<none></none>		
Mountable secrets:	prometheus-test-token-hdhkg		
Tokens :	prometheus-test-token-hdhkg		
Events:	<none></none>		
[mustql:1500000-20700	t_kj ~]# ∎		

kubectl describe secret prometheus-test-token-hdhkg -n kube-system

[root@h105966	02 20702 etakj ~]# kubertl describe secret prometheus-test-taken-hdhkg -n kube-system
Name: Namespace:	prometheus-test-token-hdhkg ruha-esetam
Labels:	
Annotations:	kubernetes.io/service-account.name: prometheus-test kubernetes.io/service-account.uid: 2f8171f7-6b81-436c-8343-2aed9484673d
Type: kuberr	netes,io/service-account-token
Data	
ca.crt:	123 bytes
namespace:	II bytes
token: 0	syJhbgc101JS021IN1ISIImtp2C161LBgwVetbGhSc0ayS3pbhJJP01200Dx1Hh0cmvLamp1CE9Mm050eGpX2LA1TQ.eyJpc3M101JrdwJ1cm50d6v2L3N1Ch2p12vH2NVdwS011w1a3v12XJU2XH1CySpb
y9zZXJ2aWNUY	W10394Vdc90YW11c3BNY20101J7GWJ1LXN53R1B51s1mT1YmV9DmV02XMLaW8vc2Vyam1j2WFjY291DnQvc2Vj2mV0LmShDw0101JWcm97zXH82XV9LXR1c3QTGG97zW4TaGH8882c1LCJ7GWJ1Cm51dGV2L
mtvL3NtcnZpY	20172Nv0WS0L3NLCn2pY2017WN1p32V0dC5UYW1L1j01cHJV0WV0aGV1Cy102XN011W1a3V12XJU2XNLCY5p5p5y22XJ2aWN1CWN1p32V0dC52X2aWN1CWFJ7291bf0Q0dW(k1j01MmY4M1cX2jcCtmm14M500M
ZZJEIGZNUMTM	m (ZDKOUDQZNZNKIIWIC3VII) olc3(ZOGVTUNN(CNZPYZNYAWSUUMTIYMUTC3(ZOGVTUNBVOZICOUTONSVOZICOUTON) och zakod kase e Asia 42/DBFPosigNu-1/96Dm95/SodrceUtONA
mbu2ogctp6wwa	swpkk-jbmxkHeevib//c_eotvniejilgrtks/v-kJzqtP5zwrrwq0nx/PXSy15LrA6MqEUGxcU1PQRcogz/Fhc0jtoo5gv8xnn2raqcipnweiizdin_mcumLw-6KewGB1/n590dK5]_CB4IDwtDk3ChEvc
S RIB-IRB-DO	12WUO-VUX3DD3BTA012CFVJBA6H0B10B2J SmDSSXSS DGpCuToWHC30XVKU8SCeXUVSAB WallCWP8VG/6H211SdFd2FKm]B6E6Gg
be an address of the second	and the second se

Record the token value, which is the **bearer_token** information to be collected.

Step 2 Configure bearer_token information.

Log in to the host where Prometheus is located, go to the Prometheus installation directory, and save the token information of the target cluster in a file.

<pre>[root@hjm-ecs prometheus-2.23.0.linux-amd64]# pwd // cost / cost /</pre>	
Froot/prometheus-2.23.0. Chux-amdo4	
[Totter] m-ecs prometheus-2.23.0. [Thux-amov4]#	
[root@hjm-ecs prometheus-2.23.0.linux-amd64]#	
[root@hjm-ecs prometheus-2.23.0.linux-amd64]# ll	
total 162488	
-rw 1 root root 5316 Jun 23 22:37 '\'	
drwxr-xr-x 2 3434 3434 🛛 4096 Nov 26 2020 console_libraries 🚬 Toke	ns of the target clusters
drwxr-xr-x 2 3434 3434 4096 Nov 26 2020 consoles	
drwx 9 root root 4096 Jun 27 11:00 data	
-rw 1 root root 943 Jun 27 11:45 k8s02_token 🦯	
-rw-rr 1 root root 943 Jun 22 11:58 k8s_token	
-rw-rr 1 3434 3434 11357 Nov 26 2020 LICENSE	
-rw-rr 1 3434 3434 3420 Nov 26 2020 NOTICE	
-rwxr-xr-x 1 3434 3434 88153522 Nov 26 2020 prometheus	
-rw 1 root root	
-rw-rr 1 3434 3434 926 Nov 26 2020 prometheus.yml.bak	
-rwxr-xr-x 1 3434 3434 78172790 Nov 26 2020 promtool	
[root@hjm-ecs prometheus-2.23.0.linux-amd64]# 📕	

Step 3 Configure a Prometheus monitoring job.

The example job monitors container metrics. To monitor other metrics, you can add jobs and compile capture rules.

```
- job_name: k8s_cAdvisor
  scheme: https
  bearer_token_file: k8s_token # Token file in the previous step.
  tls_config:
    insecure_skip_verify: true
  kubernetes_sd_configs: # kubernetes automatic discovery configuration
   - role: node # Automatic discovery of the node type
    bearer_token_file: k8s_token # Token file in the previous step
    api_server: https://192.168.0.153:5443 # API server address of the Kubernetes cluster
    tls_config:
     insecure_skip_verify: true # Skip the authentication on the server.
  relabel_configs: ## Modify the existing label of the target cluster before capturing metrics.
  - target_label: __address
    replacement: 192.168.0.153:5443
    action: replace
    ## Convert metrics_path to /api/v1/nodes/${1}/proxy/metrics/cadvisor.

# Obtain data from kubelet using the API server proxy.
- source_labels: [___meta_kubernetes_node_name] # Specifies the source label to be processed.

    regex: (.+) # Matched value of the source label. (.+) indicates that any value of the source label can
be matched.
    target_label: __metrics_path_
                                      # Specifies the label to be replaced.
    replacement: /api/v1/nodes/${1}/proxy/metrics/cadvisor # Indicates the new label, that is, the value of
__metrics_path__ ${1} indicates the value that matches the regular expression, that is, node name.
  - target_label: cluster
    replacement: xxxxx ## (Optional) Enter the cluster information.
### The following job monitors another cluster.

    job_name: k8s02_cAdvisor

  scheme: https
  bearer_token_file: k8s02_token # Token file in the previous step
  tls_config:
    insecure_skip_verify: true
  kubernetes_sd_configs:
  - role: node
    bearer_token_file: k8s02_token # Token file in the previous step
    api_server: https://192.168.0.147:5443 # API server address of the Kubernetes cluster
    tls config:
     insecure_skip_verify: true # Skip the authentication on the server.
  relabel_configs: ## Modify the existing label of the target cluster before capturing metrics.

    target_label: __address_

    replacement: 192.168.0.147:5443
    action: replace
  - source_labels: [__meta_kubernetes_node_name]
    regex: (.+)
    target label: metrics path
    replacement: /api/v1/nodes/${1}/proxy/metrics/cadvisor
```

```
- target_label: cluster
```

replacement: xxxx ## (Optional) Enter the cluster information.

Step 4 Enable Prometheus.

After the configuration, enable Prometheus.

./prometheus --config.file=prometheus.yml

Step 5 Log in to Prometheus and view the monitoring information.

n Onicalaly					
s02_cAdvisor (2/2 up) show less					
ndpoint	State	Labels	Last Scrape	Scrape Duration	Erro
tps://192.168.0.223:5443/api/v1/nodes/192.168.0.110:1025 proxy/metrics/cadvisor	UP	cluster-"h11111111" job="k8s02_cAdvisor"	1.689s	47.677ms	
tps://192.168.0.223:5443/api/v1/nodes/192.168.0.162:1025 proxy/metrics/cadvisor	UP	cluster="http://www.instance="192.168.0.162" job="k8s02_cAdvisor"	7.279s	65.193ms	
s_cAdvisor (4/4 up) show less					
ndpoint	State	Labels	Last Scrape	Scrape Duration	Erro
tps://192.168.0.153:5443/api/v1/nodes/192.168.0.65:10250/ oxy/metrics/cadvisor	UP	cluster="H"""""""""""""""""""""""""""""""""""	12.365s	37.925ms	
ttps://192.168.0.153:5443/api/v1/nodes/192.168.0.250:1025 /proxy/metrics/cadvisor	UP	cluster="1"#IL#U ##I#-testhpa" instance="192.168.0.250" job="k8s_cAdvisor"	2.390s	29.235ms	
ttps://192.168.0.153:5443/api/v1/nodes/192.168.0.109:1025 /proxy/metrics/cadvisor	UP	cluster="1NNU.JUW-testhpa" instance="192.168.0.109" job="k8s_cAdvisor"	1.578s	102.146ms	
ttps://192.168.0.153:5443/api/v1/nodes/192.168.0.228:1025 /proxy/metrics/cadvisor	UP	cluster-"	416.000ms	21.256ms	
ometheus Alerts Graph Status • Help Classic UI Enable query history ② Use local time ③ Enable autocomplete ④ container,cpu_Joad_average_100jnamespace="default"}				- 20ma Baselation 1	E
			Monito	ring task of clust	er 2
Table Graph				1	
Table Graph Containing Containin	eeOfc-as4d-4ce4-a8a8-1f7eba03d2bf/0008 d="test-7b554bd6fc-xpbhc"] od946ae0fc-as4d-4ce4-a8a8-1f7eba03d2b cefault 946ae0fc-as4d-4ce4-a8a8-1f7eba	b 1860a 1187214c1fb48421814d1a 1b3d1944ed3e27d7b6ad3d188387v*, image =*cce-paule3.1*, instan https://www.instan.uk/abad3sed8ed6dc218832e8453863a fc137b57; image =*cjoinst3uta2564 d0127d2810* / imanesce *d4daff : dde1*r553d8d8/cdc21	:e= "192.168.0.162", job= "k8s02_cAdvisor 16d511ffa63777489af47f250b70d1570e42	name="k8s_POD_te	st- e3571ad
Table Graph C Evaluation time 2 Evaluation time 2	eofic-aa4d-4cc4-a8a5-117/ebs03d2bf/0008 d="test-70554bd6f-rspbhc") d445ac9tc-aa4d-4ca4-38a5-117eba03d2b d4fault_946ac9fc-aa4d-4cc4-38a8-117eba 38-117eba03d2bf', instance="192.166.0.1 5yfburt3bbe/pod94bac9c-aa4d-4ce4-38a synburt3bbe/pod94bac9c-aa4d-4ce4-38a synburt3bbe/pod94bac9c-aa4d-4ce4-38a	b halocu i 16774cm (hiskas) i kala in bis 1944ch (2007) kala do di bis 1976; "Imagee" sce passe 3.11, intan 1978/2017 i la superiori di bis 1978/2017 i la superiori di bis 1978/2017 i la superiori di bis 1978/2017 i la 2017 julio - Riangga con di bis 1978/2017 i la superiori di bis 2017 i la superiori di bis 2017 i la superiori 2017 julio - Riangga con di bis 2017 i la superiori di bis 2017 i l	se="192.168.0.162", job="k8s02_c4dvisor 16d511ffs63777499af47f250b7od1570e42 cce-pause3.1", instance="192.168.0.10p"	hame="k8s_POD_te 8b67666567085f2bec Monitoring job="k8s_cAdvisor",	st- e3571ad task of

----End

8.2 Monitoring GPU Metrics Using DCGM-Exporter

Application Scenarios

If a cluster contains GPU nodes, learn about the GPU resources used by GPU applications, such as the GPU usage, memory usage, running temperature, and power. You can configure auto scaling policies or set alarm rules based on the obtained GPU metrics. This section walks you through how to observe GPU resource usage based on open source Prometheus and DCGM Exporter. For more details about DCGM Exporter, see **DCGM Exporter**.

NOTICE

DCGM Exporter is an open source component that specifically applies to the native GPUs (nvidia.com/gpu) within the Kubernetes community. It is important to note that GPU virtualization resources provided by CCE cannot be monitored.

Prerequisites

- You have created a cluster and there are GPU nodes and GPU related services running in the cluster.
- The CCE AI Suite (NVIDIA GPU) and Cloud Native Cluster Monitoring add-ons have been installed in the cluster.
 - CCE AI Suite (NVIDIA GPU) is a device management add-on that supports GPUs in containers. To use GPU nodes in the cluster, this add-on must be installed. Select and install the corresponding GPU driver based on the GPU type and CUDA version.
 - Cloud Native Cluster Monitoring monitors the cluster metrics. During the installation, you can interconnect this add-on with Grafana to gain a better observability of your cluster.

D NOTE

- The deployment mode of the add-on should be Local Data Storage.
- The configuration for interconnecting with Grafana is supported by the Cloud Native Cluster Monitoring add-on of a version earlier than 3.9.0. For the add-on of version 3.9.0 or later, if Grafana is required, install the Grafana add-on separately.

Enabling DCGM-Exporter Using CCE AI Suite (NVIDIA GPU)

Step 1 Enable the core component DCGM-Exporter using the CCE AI Suite (NVIDIA GPU) add-on.

- 1. Log in to the CCE console and click the cluster name to access the cluster console. In the navigation pane, choose Add-ons, locate CCE AI Suite (NVIDIA GPU) on the right, and click Install.
- 2. Enable **Use DCGM-Exporter to Observe DCGM Metrics**. Then, the DCGM-Exporter component will be deployed on the GPU nodes.

NOTICE

If the add-on version is 2.7.40 or later, DCGM-Exporter can be deployed.

After DCGM-Exporter is enabled, if you want to report GPU monitoring data to AOM, enable the function of reporting data to AOM in the Cloud Native Cluster Monitoring add-on. GPU metrics reported to AOM are custom metrics and you will be billed on a pay-per-use basis for them. For details, see **Pricing Details**.

3. Configure other parameters for the add-on and click **Install**. For details about parameter settings, see **CCE AI Suite (NVIDIA GPU)**.

Step 2 Monitor the GPU metrics of the target application.

1. Check whether the DCGM-Exporter component is running properly: kubectl get pod -n kube-system -owide

Information similar to the following is displayed:

NAMEREADYSTATUSRESTARTSAGEIPNODENOMINATED NODEREADINESS GATESdcgm-exporter-hkr771/1Running017m172.16.0.11192.168.0.73<none><none>

2. Call the DCGM-Exporter API and verify the collected GPU information.

172.16.0.11 indicates the pod IP address of the DCGM-Exporter component. curl 172.16.0.11:9400/metrics | grep DCGM_FI_DEV_GPU_UTIL

LT C							CULC 1/2	.10.0.11		strics (l dreb no	OH_IT_DEV_OF	0_0115								
2	i Tot	al	% F	Receive	ed 🕷	Xferd	Average	Speed	Time	Time	Time	Current									
							Dload	Upload	Total	Spent	Left	Speed									
100	55	14		5514			5384k	# HE	LP DCGM	FI_DEV	GPU_UTIL	GPU utiliza	tion (in %)								
0#	TYPE	DCGM		DEV_G		TIL gau	ige														
	DCGM	FI DE	EV 0	SPU_UT1	[L{g	ou="0",	UUID="GP	U-a247b5	63-c253-	-f17e-d1	lfa-14617	43653f1",dev	ice="nvidia	0",modelName=	"Tesla T	4",Hostname=	dcgm-exporter-	hkr77"	DCGM F	I DRIVER	VE
RSI	ON="	525.66	0.13	3",con	tain	er="con	tainer-1	*,namesp	ace="def	fault",p	ood="gpu-	test-7766b78	45d-t985k*}								
•					· •	5384k															

Step 3 Enable custom metric collection for the Cloud Native Cluster Monitoring addon.

- 1. Log in to the CCE console and click the cluster name to access the cluster console.
- 2. In the navigation pane, choose **Add-ons**, locate the Cloud Native Cluster Monitoring add-on, and click **Edit**.
- 3. In the **Parameters** area, enable **Custom Metric Collection**.

Parameters



Automatically collects application metrics in the form of Annotations service discovery. By default, applications in the kube-system and monitoring namespaces are ignored. Custom Metric Configuration

4. Click OK.

Step 4 Enable metric collection for the DCGM-Exporter component.

- 1. Log in to the CCE console and click the cluster name to access the cluster console.
- 2. In the navigation pane, choose **Settings**. Then, click the **Monitoring** tab.
- 3. In the **Collection configuration** area, click the **ServiceMonitor Policies** tab and click **Manage**.
- 4. Find the ServiceMonitor policy of the DCGM-Exporter component and enable it.

Collection config	uration			
ServiceMonitor Polic	ies () PodM	onitor Policies ⑦		
Create from YAML	\supset			
Q Select a property	or enter a keyword.			0
Name \ominus	Name 🔶	Label \ominus	Created 🔶 Enabled	⊖ Operation
dcgm-exporter	monitoring	app.kubernetes.io/name dcgm-exporter managed-by prometheus-operator	10 days ago	Edit YAML Delete

Step 5 View the metric monitoring information on Prometheus.

After Prometheus and the related add-on are installed, Prometheus creates a ClusterIP Service by default. To expose Prometheus to external systems, create a NodePort or LoadBalancer Service for it. For details, see **Monitoring Custom Metrics Using Cloud Native Cluster Monitoring**.

You can view the GPU utilization and other metrics on a GPU node, as shown in the figure below. For more GPU metrics, see **Observable Metrics**.

8 C U
Execute
Load time 31ms Resolution 14s Result series
; job="dogm-exporter", modelName="Tesla P4", 0
Remove Panel

Step 6 Log in to Grafana and view the GPU information.

If you have installed Grafana, you can import the **NVIDIA DCGM Exporter Dashboard** to display GPU metrics.

For details about how to import dashboards to Grafana, see Manage dashboards.



----End

Manually Deploying DCGM-Exporter

This section describes how to deploy the DCGM-Exporter component in a cluster to collect GPU metrics and expose GPU metrics through port 9400.

- **Step 1** Log in to a node with an EIP bound and that uses the Docker container engine.
- Step 2 Pull the DCGM-Exporter image to the local host. The image address comes from the DCGM official example. For details, see https://github.com/NVIDIA/dcgm-exporter/blob/main/dcgm-exporter.yaml. docker pull nvcr.io/nvidia/k8s/dcgm-exporter:3.0.4-3.0.0-ubuntu20.04
- **Step 3** Push the DCGM-Exporter image to SWR.
 - 1. (Optional) Log in to the SWR console, choose **Organizations** in the navigation pane, and click **Create Organization** in the upper right corner.

Skip this step if you already have an organization.

2. In the navigation pane, choose **My Images** and then click **Upload Through Client**. On the page displayed, click **Generate a temporary login command**

and click \square to copy the command.

- 3. Run the login command copied in the previous step on the cluster node. If the login is successful, the message "Login Succeeded" is displayed.
- 4. Add a tag to the DCGM-Exporter image.

docker tag {*Image name 1:Tag 1*}*|*{*Image repository address*}*|*{*Organization name*}*|*{*Image name 2:Tag 2*}

- *{Image name 1:Tag 1}*: name and tag of the local image to be uploaded.
- {Image repository address}: The domain name at the end of the login command in Step 3.2 is the image repository address, which can be obtained on the SWR console.
- *{Organization name}*: name of the organization created in **Step 3.1**.
- *{Image name 2:Tag 2}*: desired image name and tag to be displayed on the SWR console.

The following is an example:

docker tag nvcr.io/nvidia/k8s/dcgm-exporter:3.0.4-3.0.0-ubuntu20.04 swr.cn-east-3.myhuaweicloud.com/container/dcgm-exporter:3.0.4-3.0.0-ubuntu20.04

5. Push the image to the image repository.

docker push {*Image repository address*}*|*{*Organization name*}*|*{*Image name 2:Tag 2*}

The following is an example:

docker push swr.cn-east-3.myhuaweicloud.com/container/dcgm-exporter:3.0.4-3.0.0-ubuntu20.04

The following information will be returned upon a successful push: 489a396b91d1: Pushed

... c3f11d77a5de: Pushed 3.0.4-3.0.0-ubuntu20.04: digest: sha256:bd2b1a73025*** size: 2414

6. To view the pushed image, go to the SWR console and refresh the **My Images** page.

Step 4 Deploy the core component DCGM-Exporter.

When deploying DCGM-Exporter on CCE, add some specific configurations to monitor GPU information. The following shows a detailed YAML file, and the information in red is important.

NOTICE

After Cloud Native Cluster Monitoring is interconnected with AOM, metrics will be reported to the AOM instance you select. **Basic metrics** can be monitored for free, and custom metrics are billed based on the standard pricing of AOM. For details, see **Pricing Details**.

apiVersion: apps/v1 kind: DaemonSet metadata: name: "dcgm-exporter"

```
namespace: "monitoring"
                             # Select a namespace as required.
 labels:
  app.kubernetes.io/name: "dcgm-exporter"
  app.kubernetes.io/version: "3.0.0"
spec:
 updateStrategy:
  type: RollingUpdate
 selector:
  matchLabels:
   app.kubernetes.io/name: "dcgm-exporter"
   app.kubernetes.io/version: "3.0.0"
 template:
  metadata:
   labels:
     app.kubernetes.io/name: "dcgm-exporter"
     app.kubernetes.io/version: "3.0.0"
   name: "dcgm-exporter"
  spec:
    containers:
    - image: "swr.cn-east-3.myhuaweicloud.com/container/dcgm-exporter:3.0.4-3.0.0-ubuntu20.04" # The
SWR image address of DCGM-Exporter. The address is the image address in Step 3.5.
     env:
     - name: "DCGM_EXPORTER_LISTEN"
                                                     # Service port number
      value: ":9400"
     - name: "DCGM_EXPORTER_KUBERNETES"
                                                        # Supports mapping of Kubernetes metrics to
pods.
      value: "true"
     - name: "DCGM_EXPORTER_KUBERNETES_GPU_ID_TYPE" # GPU ID type. The value can be uid or
device-name.
     value: "device-name"
     name: "dcgm-exporter"
     ports:
     - name: "metrics"
      containerPort: 9400
                 # Request and limit resources as required.
     resources:
      limits:
       cpu: '200m'
       memory: '256Mi'
      requests:
       cpu: 100m
       memory: 128Mi
     securityContext:
                        # Enable the privilege mode for the DCGM-Exporter container.
      privileged: true
      runAsNonRoot: false
      runAsUser: 0
     volumeMounts:
     - name: "pod-gpu-resources"
      readOnly: true
      mountPath: "/var/lib/kubelet/pod-resources"
     - name: "nvidia-install-dir-host" # The environment variables configured in the DCGM-Exporter
image depend on the file in the /usr/local/nvidia directory of the container.
      readOnly: true
      mountPath: "/usr/local/nvidia"
   imagePullSecrets:
     - name: default-secret
   volumes:
    - name: "pod-gpu-resources"
     hostPath:
      path: "/var/lib/kubelet/pod-resources"
     name: "nvidia-install-dir-host"
                                       # The directory where the GPU driver is installed.
     hostPath:
      path: "/opt/cloud/cce/nvidia"
                                       #If the GPU add-on version is 2.0.0 or later, replace the driver
installation directory with /usr/local/nvidia.
               # Label generated when CCE creates GPU nodes. You can set node affinity for this
   affinity:
component based on this label.
     nodeAffinity:
      requiredDuringSchedulingIgnoredDuringExecution:
       nodeSelectorTerms:
       - matchExpressions:
```

```
- key: accelerator
          operator: Exists
kind: Service
apiVersion: v1
metadata:
 name: "dcgm-exporter"
 namespace: "monitoring"
                              # Select a namespace as required.
 labels:
  app.kubernetes.io/name: "dcgm-exporter"
  app.kubernetes.io/version: "3.0.0"
spec:
 selector:
  app.kubernetes.io/name: "dcgm-exporter"
  app.kubernetes.io/version: "3.0.0"
 ports:
 - name: "metrics"
  port: 9400
apiVersion: monitoring.coreos.com/v1
kind: ServiceMonitor
metadata:
 labels:
  app.kubernetes.io/name: "dcgm-exporter"
  app.kubernetes.io/version: "3.0.0"
 name: dcgm-exporter
 namespace: monitoring #Select a namespace as required.
spec:
 endpoints:
 - honorLabels: true
  interval: 15s
  path: /metrics
  port: metrics
  relabelings:
  - action: labelmap
   regex: __meta_kubernetes_service_label_(.+)
  - action: replace
   sourceLabels:
    - __meta_kubernetes_namespace
    targetLabel: kubernetes_namespace
  - action: replace
   sourceLabels:
      __meta_kubernetes_service_name
   targetLabel: kubernetes_service
  scheme: http
 namespaceSelector:
  matchNames:
  - monitoring
                    # Select a namespace as required.
 selector:
  matchLabels:
   app.kubernetes.io/name: "dcgm-exporter"
```

Step 5 Monitor the GPU metrics of the target application.

1. Check whether the DCGM-Exporter component is running properly: kubectl get po -n monitoring -owide

```
Information similar to the following is displayed:
```

```
# kubectl get po -n monitoring -owide

NAME READY STATUS RESTARTS AGE IP NODE

NOMINATED NODE READINESS GATES

dcgm-exporter-hkr77 1/1 Running 0 17m 172.16.0.11 192.168.0.73

<none> <none>
```

2. Call the DCGM-Exporter API and verify the collected GPU information.

```
172.16.0.11 indicates the pod IP address of the DCGM-Exporter component. curl 172.16.0.11:9400/metrics | grep DCGM_FI_DEV_GPU_UTIL
```

[roo]# curl 172.16.0.11:9400/metrics grep DCGM_FI_DEV_GPU_UTIL
% Total % Received % Xferd Average Speed Time Time Time Current
Dload Upload Total Spent Left Speed
100 5514 0 5514 0 0 5384k # HELP DCGN FI DEV GPU UTIL GPU utilization (in %).
8# TYPE DCGM_FI_DEV_GPU_UTIL gauge
DCGM FI DEV GPU UTIL{gpu="0",UUID="GPU-a247b563-c253-f17e-d1fa-1461743653f1", device="nvidia0", modelName="Tesla T4", Hostname="dcgm-exporter-hkr77", DCGM FI DRIVER V
RSION="525.60.13",container="container-1",namespace="default" pod="gpu-test-7766b7845d-t985k"} 0

Step 6 View the metric monitoring information on Prometheus.

After Prometheus and the related add-on are installed, Prometheus creates a ClusterIP Service by default. To expose Prometheus to external systems, create a NodePort or LoadBalancer Service for it. For details, see **Monitoring Custom Metrics Using Cloud Native Cluster Monitoring**.

You can view the GPU utilization and other metrics on a GPU node, as shown in the figure below. For more GPU metrics, see **Observable Metrics**.

🔮 Prometheus Alerts Graph Status * Help	⇒ ()
Use local time Enable query history 🛛 Enable autocomplete 🖓 Enable highlighting 🖓	Enable linter
Q DCGM_FI_DEV_GPU_UTIL	Secure Execute
Table Graph	Load time: 151ms Resolution: 14s Result series: 1
< Evaluation time >	
DCGM, FL_DEV_GPU_UTIL(DCGM, FL_DRIVER, VERSION="525.60.13", Hostname="dcgm-exporter-hkr77", UUID="GPU-a247 device="midia0", gpu="0", instance="172.16.0.119400", job="dcgm-exporter", kubernetes, namespace="monitoring", kub	x563-223-417e-d1fa-146174365311*, app_kubernetes_jo_name="dcgm-exporter" app_kubernetes_jo_version="3.0.0"; container="container=1"; operators_service="dcgm-exporter"; modelName="Tesla T4"; namespace="dcfault" [pdd="gpu-test-7766b7845d-19835"; pervice="dcgm-exporter"]
	Remove Panel

Step 7 Log in to Grafana and view the GPU information.

If you have installed Grafana, you can import the **NVIDIA DCGM Exporter Dashboard** to display GPU metrics.

For details about how to import dashboards to Grafana, see Manage dashboards.



----End

Observable Metrics

The following table lists some observable GPU metrics. For details about more metrics, see **Field Identifiers**.

Table 8-1 Usage

Metric Name	Metric Type	Unit	Description
DCGM_FI_DEV_GP U_UTIL	Gauge	%	GPU usage
DCGM_FI_DEV_M EM_COPY_UTIL	Gauge	%	Memory usage
DCGM_FI_DEV_EN C_UTIL	Gauge	%	Encoder usage
DCGM_FI_DEV_DE C_UTIL	Gauge	%	Decoder usage

Table 8-2 Memory

Metric Name	Metric Type	Unit	Description
DCGM_FI_DEV_FB _FREE	Gauge	MB	Number of remaining frame buffers. The frame buffer is called VRAM.
DCGM_FI_DEV_FB _USED	Gauge	МВ	Number of used frame buffers. The value is the same as the value of memory-usage in the nvidia- smi command.

Table 8-3 Temperature and power

Metric Name	Metric Type	Unit	Description
DCGM_FI_DEV_GP U_TEMP	Gauge	°C	Current GPU temperature of the device
DCGM_FI_DEV_P OWER_USAGE	Gauge	W	Power usage of the device

8.3 Reporting Prometheus Monitoring Data to a Third-Party Monitoring Platform

Application Scenarios

The Cloud Native Cluster Monitoring add-on can report Prometheus metrics collected from clusters to a specified platform, for example, AOM or a third-party

platform that supports Prometheus metrics. This section explains how to configure settings for Cloud Native Cluster Monitoring to send collected metrics to a third-party's Prometheus instance.

Step 1: Obtain the Data Reporting Address

Prometheus provides standard Remote Write APIs. You can enter the source address (Remote Write URL) in the Cloud Native Cluster Monitoring add-on for storing the locally collected monitoring data in a Prometheus instance remotely.

- If the Prometheus instance for receiving data is provided by a third-party vendor, view the Remote Write URL on the vendor's console.
- If the Prometheus instance for receiving data is an on-premises one, the Remote Write URL is **https://** {prometheus_addr} /api/v1/write, where {prometheus_addr} indicates the IP address and port number for external access.

Step 2: Obtain the Authentication Mode

- For the third-party Prometheus instance, go to the vendor's console to view the token or account password used for authorized access.
- For the on-premises Prometheus instance, perform the following operations to obtain a token:
 - a. If this Prometheus instance is deployed in a Kubernetes cluster, view the token in the corresponding container. If this Prometheus instance is deployed on a VM, skip this step. kubectl exec -ti -n *monitoring prometheus-server-0* -- sh

Replace the variables in the command as needed:

monitoring. indicates the namespace where a Prometheus pod is in.

entheus --web.comsole.templates=/etc/prometheus/comsoles --web.comsole.libraries=/etc/prometheus/comsole_libraries --storage.tsdb.retention.time.ld ometheus --web.endole-lifecycle --guery.lookdack.delta=3m --endole-feature=remote-write-receiver --web.route-prefixe/ --web.listen-address=172.16.0.3

- *prometheus-server-0*. indicates the name of a Prometheus pod.
- b. Check the location of the configuration file. ps -aux | grep prometheus

Information similar to the following is displayed:

c. View and record the token information in **prometheus.env.yaml**. cat /etc/prometheus/config_out/prometheus.env.yaml



Step 3: Connect to a Third-Party Monitoring Platform

- **Step 1** Log in to the CCE console, click the name of a cluster with the Cloud Native Cluster Monitoring add-on installed to access the cluster console.
- **Step 2** In the navigation pane, choose **Add-ons**, locate the Cloud Native Cluster Monitoring add-on, and click **Edit**.
- **Step 3** Enable **Report Monitoring Data to a Third-Party Platform** so that the data collected by Cloud Native Cluster Monitoring can be reported to a third-party monitoring platform.
 - Source Address: Remote Write URL obtained in step 1, for example, https:// 127.0.0.1:9090/api/v1/write.
 - **Authentication method**: Select the authentication method supported by the third-party monitoring platform in **step 2**.
 - Basic Auth: Enter the user name and password.
 - **Bearer Token**: Enter the identity credential (token).

Report Monitoring Data to a Third-Party Platform

Enter the address from which to report data. Enter a complete RemoteWrite address, for example, https://127.0.0.1:9090/api/v1/write. Authentication method Basic Auth Bearer Token None account password Please enter the account number Please enter your password	Enter the address from which to Enter a complete RemoteWrite add Authentication method Basic Auth Bearer Tok	report dai dress, for	ta. example, http	s://127.0.0.1	:9090/a	pi/v1/write	B.
Enter a complete RemoteWrite address, for example, https://127.0.0.1:9090/api/v1/write. Authentication method Basic Auth Bearer Token None account password Please enter the account numbei Please enter your password	Enter a complete RemoteWrite add Authentication method Basic Auth Bearer Tok	dress, for	example, http	s://127.0.0.1	:9090/a	pi/v1/write	B.
Authentication method Basic Auth Bearer Token None account password Please enter the account number Please enter your password 🐼	Authentication method Basic Auth Bearer Tok						
account password Please enter the account number Please enter your password		ken	None				
paccount password Please enter the account number Please enter your password (20)							
Please enter the account number Please enter your password @	account	pass	word				
	Please enter the account number	e Ple	ease enter yo	ur password	Ø		
	Skip certificate authentication (?)						
Skip certificate authentication 🕜	Yes No						

Step 4 After the modification is complete, click **OK**.

----End

Step 4: Check the Data Sending and Receiving Statuses

After the preceding configuration is complete, log in to the Prometheus console supported by the third-party platform and view the Prometheus metrics with remote write on the **Graph** page.



8.4 Obtaining Prometheus Data Using PromQL Statements

Prometheus Query Language (PromQL) is a language that retrieves and consolidates time series data. Prometheus, an open-source monitoring system, is responsible for collecting and storing this data. Each time series has a unique identifier and a series of timestamp-value pairs. PromQL is a fundamental part of Prometheus, using simple expressions that include identifiers and tags to retrieve and consolidate time series data. This allows you to quickly identify and resolve issues as needed.

For details about how to use PromQL, see **QUERYING PROMETHEUS**.

Obtaining Huawei Cloud Prometheus Monitoring Data Through the Console

- **Step 1** Install the Cloud Native Cluster Monitoring add-on in a cluster to collect Prometheus monitoring data.
 - 1. Log in to the CCE console and click the cluster name to access the cluster console. In the navigation pane, choose **Add-ons**.
 - 2. Locate the Cloud Native Cluster Monitoring add-on and click Install.

You are advised to focus on the following configurations, and adjust any other configurations as necessary. For details, see **Cloud Native Cluster Monitoring**.

- **Report Monitoring Data to AOM**: After this function is enabled, the add-on will report monitoring data to AOM.
- Target AOM Instance: Select an AOM instance for the add-on to report metrics.

Data Storage Configuration (At least one item must be enabled.)				
Report Monitoring Data to AOM				
Enable				
Target AOM Instance ⑦				
cia-test	Creating Instance View AOM Instances			
This add-on collects monitoring data from clusters and custom metrics are billed on a pay-per-use basis.View	is sends it to AOM, where Prometheus data collected by CCE is stored. Basic container metrics are free, but ing Basic Container Indicators [2] View Billing Metrics [2]			

3. Click Install.

Step 2 Go to the AOM console to check monitoring data.

- 1. Log in to the AOM 2.0 console. In the navigation pane, choose **Metric Browsing**.
- 2. Select the Prometheus instance interconnected with AOM, select **Prometheus statement**.

For example, to obtain the number of requests received by CoreDNS and the response success rate, run the following PromQL statement: sum(rate(coredns_dns_responses_total{instance=~"10.0.0.2:9153",rcode=~"NOERROR|NXDOMAIN"}[5m]))/sum(rate(coredns_dns_responses_total{instance=~"10.0.0.2:9153"}[5m]))

10.0.0.2 is the IP address of the CoreDNS container. The following figure shows the command output.



----End

Obtaining Open Source Prometheus Monitoring Data Through the Console

Step 1 After the Cloud Native Cluster Monitoring add-on is installed in a cluster, create a NodePort or LoadBalancer Service for Prometheus to access the external networks.

NOTICE

After external access is provided, you can access the Prometheus web page without authentication, which poses security risks. If you have high security requirements, you can do security hardening. You can use, for example, the Nginx reverse proxy and HTTP basic authentication to protect the console and restrict users who can access the Prometheus web page.

The following shows an example for creating a NodePort Service:
apiVersion: v1
kind: Service
metadata:
name: prom-np
labels:
app.kubernetes.io/name: prometheus
prometheus: server
namespace: monitoring
spec:
selector:
app.kubernetes.io/name: prometheus
prometheus: server
externalTrafficPolicy: Cluster
ports:
- name: cce-service-0
targetPort: 9090
nodePort: 0
port: 9090
protocol: TCP
type: NodePort

Step 2 After an EIP is bound to a node, enter *Node EIP:Node port* in the address box of a browser to obtain the monitoring data through PromQL.



⁻⁻⁻⁻End

Obtaining Huawei Cloud Prometheus Monitoring Data Through APIs

Step 1 Log in to the AOM 2.0 console. In the navigation pane, choose **Prometheus Monitoring** > **Instances**.

- **Step 2** Click the name of the Prometheus instance interconnected with AOM and click **Settings**.
- **Step 3** Check the Grafana data source configurations and copy the public or intranet HTTP URL as required.

NOTE

If no access code is created, create one first.

Grafana Data Source Info				
Intranet Public Net	work			
HTTP URL	https://aom.cn-east-3.myhuaweicloud.com/v1/			
Username	No. Advantation of the second			
Password	6t**hJ			

Step 4 Use an HTTP URL to obtain Prometheus monitoring data.

The following shows an example. (For more information about how to use HTTP APIs to obtain Prometheus monitoring data, see HTTP API.)

```
GET <HTTP URL>/api/v1/query
Content-Type: application/json
Authorization: <Token>
{
"query":" coredns_dns_responses_total",
"time":" 1708655317.719",
"timeout":"1000"
}
```

In the preceding command, *<HTTP URL>* specifies the HTTP URL in the previous step, and *<Token>* specifies the password obtained in the previous step. Other URL query parameters are as follows:

- query: Prometheus expression query string
- time: timestamp for obtaining monitoring data
- timeout: timeout interval, in milliseconds

The following shows an example command:

curl -H 'Authorization: *<Token>*' -H 'Content-Type: application/json' -X GET '*<HTTP URL>*/api/v1/query? query=coredns_dns_responses_total&time=1708655317.719&timeout=1000'

The expected results are as follows:

root@lts-turbo-1232-82433 ~]# curl -H 'Authorization:
'-H 'Content-Type: application/json' -X GET 'https://
/v1/query?query=coredns_dns_responses_total&time=1708655317.719&timeout=1000'
["status":"success","data":{"resultType":"vector","result":[{"metric":{"name":"coredns_dns_responses_total","app":"coredns
,"app_kubernetes_io_managed_by":"Helm","cluster":"8ae8a007-a5f1-11ee-b8d8-0255ac10026e","cluster_category":"self","cluster_nam
':"lts-turbo-1232","instance":"192.168.2.116:9153","job":"coredns","k8s_app":"coredns","kubernetes_io_cluster_service":"true",
<pre>kubernetes_io_name":"CoreDNS","kubernetes_namespace":"kube-system","kubernetes_service":"coredns","namespace":"kube-system","p</pre>
ıgin":"loadbalance","pod":"coredns-6fdd4dfc68-gcdrh","prometheus":"monitoring/server","rcode":"NOERROR","release":"cceaddon-co
edns","server":"dns://192.168.2.116:5353","service":"coredns","zone":"."},"value":[1708655317.719,"429922"]},{"metric":{"nam
<pre>":"coredns_dns_responses_total","app":"coredns","app_kubernetes_io_managed_by":"Helm","cluster":"8ae8a007-a5f1-11ee-b8d8-025</pre>
ac10026e","cluster_category":"self","cluster_name":"lts-turbo-1232","instance":"192.168.2.116:9153","job":"coredns","k8s_app":
coredns","kubernetes_io_cluster_service":"true","kubernetes_io_name":"CoreDNS","kubernetes_namespace":"kube-system","kubernete:
_service":"coredns", ["] namespace":"kube-system","plugin":"loadbalance","pod":"coredns-6fdd4dfc68-gcdrh","prometheus":"monitoring,
server","rcode":"NXDOMAIN","release":"cceaddon-coredns","server":"dns://192.168.2.116:5353","service":"coredns","zone":"."},"v
lue":[1708655317.719,"926364"]}]}}[root@lts-turbo-1232-82433 ~]#

----End

Obtaining Open Source Prometheus Monitoring Data Through APIs

Step 1 Obtain the HTTP API URL.

- If you access open source Prometheus in a cluster, the HTTP API UR is the default Service address http://<cluster IP>:9090 of Prometheus server.
- If you use the public network to access open source Prometheus, you need to associate a LoadBalancer Service to the Prometheus server. http://<ELB IP>9090 is the HTTP API URL.
- Step 2 Use HTTP APIs to obtain Prometheus monitoring data.

The following describes how to obtain open source Prometheus monitoring data through the HTTP APIs in the cluster.

Obtain instantaneous monitoring data. GET </HTTP API>/api/v1/query Content-Type: application/json Param: { "query":" coredns_dns_responses_total", "time":" 1708655317.719", "timeout":"1000" }

The following shows an example command:

curl -X GET -H "Content-Type: application/json" '*<HTTP API>*/api/v1/query? query=coredns_dns_responses_total&time=1708655317&timeout=1000'

The expected results are as follows:

root@lts-turbo-1232-82433 ~]# curl -X GET -H "Content-Type: application/json" -H "Accept: application/json" '
0/api/v1/query?query=coredns_dns_responses_total&time=1708655317&timeout=1000'
"status":"success","data":{"resultType":"vector","result":[{"metric":{"name":"coredns_dns_responses_total","app":"coredns
"app_kubernetes_io_managed_by":"Helm","instance":"192.168.2.116:9153","job":"coredns","k8s_app":"coredns","kubernetes_io_clus
r_service":"true","kubernetes_io_name":"CoreDNS","kubernetes_namespace":"kube-system","kubernetes_service":"coredns","namespa
":"kube-system","plugin":"loadbalance","pod":"coredns-6fdd4dfc68-gcdrh","rcode":"NOERROR","release":"cceaddon-coredns","serve
:"dns://192.168.2.116:5353","service":"coredns","zone":"."},"value":[1708655317,"429922"]},{"metric":{"name":"coredns_dns
esponses_total","app":"coredns","app_kubernetes_io_managed_by":"Helm","instance":"192.168.2.116:9153","job":"coredns","k8s_ap
:"coredns","kubernetes_io_cluster_service":"true", ["] kubernetes_io_name":"CoreDNS","kubernetes_namespace":"kube-system","kubern
es_service":"coredns", ["] namespace":"kube-system","plugin":"loadbalance","pod":"coredns-6fdd4dfc68-gcdrh","rcode":"NXDOMAIN","r
ease":"cceaddon-coredns","server":"dns://192.168.2.116:5353","service":"coredns","zone":"."},"value":[1708655317,"926364"]}]}
root@lts-turbo-1232-82433 ~1#

 Obtain monitoring data within a specified time range. GET </HTTP API>/api/v1/query Content-Type: application/json Param: {
 "query":" coredns_dns_responses_total",
 "start":1708655317.

"step":30 }

"end":1708655318,

Other URL query parameters are as follows:

- **start**: time when you start to obtain monitoring data
- end: time when you stop obtaining the monitoring data
- step: step of the data interval when the monitoring data is returned

The following shows an example statement:

curl -X GET -H "Content-Type: application/json" -H "Accept: application/json" '<*HTTP API>*/api/v1/ query?query=coredns_dns_responses_total&start=1708655317&end=1708655318&setp=30'

The expected results are as follows:



----End

8.5 Collecting GPU Pod Monitoring Metrics and Setting Up a Grafana Dashboard

Application Scenarios

Monitoring metrics for GPU pods, such as gpu_pod_core_percentage_total, gpu_pod_core_percentage_used, gpu_pod_memory_total, and gpu_pod_memory_used, have been added to CCE AI Suite (NVIDIA GPU) of versions 2.1.30, 2.7.46, and later. These metrics allow for comprehensive monitoring of the resource usage of a GPU card and the compute and memory usage of the GPU workloads that run on a GPU. By using these metrics, you can create monitoring dashboards at different levels (for example, by pod or node) to stay informed about cluster information in real-time. For details about GPU metrics, see GPU Metrics.

The Cloud Native Cluster Monitoring add-on does not automatically collect these metrics. To view them on Grafana dashboards, you must manually configure the Cloud Native Cluster Monitoring add-on for data collection and set up a dashboard. This section provides an example using pod monitoring to explain how to collect GPU pod monitoring metrics and establish a dashboard.

Prerequisites

- A NVIDIA GPU node is running properly in the cluster.
- The CCE AI Suite (NVIDIA GPU) add-on has been installed in the cluster and its version is 2.1.30, 2.7.46, or later. For details about how to install the add-on, see CCE AI Suite (NVIDIA GPU).
- The Cloud Native Cluster Monitoring add-on and Grafana add-on have been installed in the cluster, and **public access** has been enabled for Grafana. For details about how to install these add-ons, see Cloud Native Cluster Monitoring and Grafana.
 - To use AOM data sources (Configuring an AOM Data Source), enable Report Monitoring Data to AOM in the Cloud Native Cluster Monitoring add-on and Interconnect with AOM in the Grafana add-on. In addition, make sure to use the same AOM instances for both add-ons.
 - To use Prometheus data sources (Configuring a Prometheus Data Source), enable Local Data Storage in the Cloud Native Cluster Monitoring add-on.

Process

Procedure	Description	Billing
Step 1: Collect GPU Pod Monitoring Metrics	The Cloud Native Cluster Monitoring add-on does not automatically collect GPU pod monitoring metrics. To view these metrics in the monitoring center, you need to manually configure the add-on for data collection.	If Report Monitoring Data to AOM is enabled for the Cloud Native Cluster Monitoring add-on, uploading GPU pod monitoring data to AOM will incur fees. For details, see Product Pricing Details .
Step 2: Configure a Data Source for Grafana	When setting up a GPU pod monitoring dashboard on the Grafana GUI, make sure to choose either the AOM or Prometheus data sources that can be properly connected to Grafana.	N/A
Step 3: Configure a Grafana Dashboard	Pod monitoring is used as an example to explain how to establish a GPU pod monitoring dashboard.	N/A

Step 1: Collect GPU Pod Monitoring Metrics

The Cloud Native Cluster Monitoring add-on does not automatically collect GPU pod monitoring metrics. To view these metrics in the monitoring center, you need to manually configure the add-on for data collection.

- **Step 1** Log in to the CCE console and click the cluster name to access the cluster console.
- Step 2 In the navigation pane, choose Cluster > Settings. In the right pane, click the Monitoring tab. In the Monitoring Settings area, locate Preset Policies under Collection Settings and click Enable. In the dialog box displayed, click OK. If you have already enabled Preset Policies, skip this step.

NOTE

The **Preset Policies** feature is used to configure preset collection policies for the Cloud Native Cluster Monitoring add-on. Enabling this feature will result in the deletion of preset ServiceMonitor and PodMonitor policies. Any manually created collection configurations will remain unaffected. Once this feature is enabled, customizing preset collection tasks becomes effortless.

Figure 8-1 Enabling the Preset Policies feature

Dashboard	Cluster Access	Network	Scheduling	Auto Scaling	Monitoring	Kubernetes	Heterogeneous Resources	Q
Monit Log S	oring Settings ettings	Monitoring Collection Preset Polici Custom confi	g Settings Settings es ⑦ igurations for preset	data collection policies	Enable			

Step 3 In the Monitoring Settings area, locate Metric Settings and click Refresh to obtain data. After you enabled Preset Policies, the Cloud Native Cluster Monitoring add-on will be automatically upgraded. Then, choose Monitoring Settings > Collection Settings > Preset Policies and click Manage.

On the **Collection Settings** page, click the search box, select **Task Name**, and choose **nvidia-gpu-device-plugin** from the drop-down list. In the **Collection Metric** column, click **Edit Trustlist**.

Figure 8-2 Editing trustlist

Collection Settings			×
Preset Policies ⑦	ServiceMonitor Policies ⑦ PodMonito	r Policies ③	
Q Task Name: nvidia-	gpu-device-plugin × Add filter		× Q
Task Name	Collection Metric	Collection Period	Enabled 🔶
nvidia-gpu-device-pl	Trustlist metrics V	- 15 + Second	
Total Records: 1			10 ~ < 1 >

Step 4 In the upper left corner of the Trustlist metrics page, click Add Metric and add the gpu_pod_core_percentage_total, gpu_pod_core_percentage_used, gpu_pod_memory_total, and gpu_pod_memory_used metrics. After completing the operation, close the current page.

----End

Step 2: Configure a Data Source for Grafana

Grafana supports:

- AOM data sources: Grafana creates the **prometheus-aom** data source automatically. Make sure this data source can be properly connected to Grafana.
- Prometheus data sources: You can use the **prometheus** data source preset in Grafana. Make sure this data source can be properly connected to Grafana.

Configuring an AOM Data Source

To use an AOM data source, ensure that **Report Monitoring Data to AOM** has been enabled for the Cloud Native Cluster Monitoring add-on, **Interconnect with AOM** has been enabled for the Grafana add-on, and the two add-ons are connected to the same AOM instance. After you enabled **Interconnect with AOM** for the Grafana add-on, the **prometheus-aom** data source is automatically generated on the Grafana GUI. Ensure that the data source can be properly connected. After the connectivity test has been passed, you can start using the AOM data source.

Step 1 In the navigation pane, choose **Cluster** > **Add-ons**. In the right pane, find the Grafana add-on and click **Access** to go to the Grafana GUI.

- **Step 2** Enter the username and password when you access the Grafana GUI for the first time. The default username and password are both **admin**. After entering the username and password, reset the password following instructions.
- **Step 3** In the upper left corner, click **a**, click **a** on the left of **Connections**, and click **Data sources** to access the **Data sources** page.
- Step 4 In the data source list, click prometheus-aom. Click Save & test at the bottom of the prometheus-aom data source page to check the data source connectivity. If "Successfully queried the Prometheus API" is displayed, the connectivity test has been passed.

Figure 8-3 Connectivity test passed

	Curses of ally superiod the Descention of DI
	Next, you can start to visualize data by building a dashboard, or by querying data in the Explore view.
Dele	Save & test

----End

Configuring a Prometheus Data Source

Before using a Prometheus data source, ensure that **Local Data Storage** has been enabled for the Cloud Native Cluster Monitoring add-on. The Grafana **prometheus** data source can connect directly to the local Prometheus data source after **Local Data Storage** is enabled. Ensure that the data source can be properly connected. After the connectivity test has been passed, you can start using the Prometheus data source.

- **Step 1** In the navigation pane, choose **Cluster** > **Add-ons**. In the right pane, find the Grafana add-on and click **Access** to go to the Grafana GUI.
- **Step 2** Enter the username and password when you access the Grafana GUI for the first time. The default username and password are both **admin**. After entering the username and password, reset the password following instructions.
- **Step 3** In the upper left corner, click , click on the left of **Connections**, and click **Data sources** to access the **Data sources** page.
- Step 4 In the data source list, click prometheus. Click Save & test at the bottom of the prometheus data source page to check the data source connectivity. If "Successfully queried the Prometheus API" is displayed, the connectivity test has been passed.

Figure 8-4 Connectivity test passed





Step 3: Configure a Grafana Dashboard

Grafana dashboards are essential for centralized monitoring and visualizing data from various data sources. They provide real-time insights into system statuses and service metrics using charts, graphs, and alarms. Grafana can generate monitoring dashboards based on GPU pod metrics, categorizing them by pod, GPU, node, and cluster. This section focuses on configuring a Grafana dashboard for pods as an example. Additional PromQL statements for different monitoring panels are available in **PromQL Statement**. You can perform the following operations to set up related dashboards.

- **Step 1** Create a Grafana dashboard to show GPU pod monitoring metrics.
 - 1. On the Grafana GUI, click it to open the menu bar on the left and click **Dashboards**. In the upper right corner of the **Dashboards** page, click **New** and choose **New dashboard** from the drop-down list.

Figure 8-5 Creating a dashboard



2. In the upper right corner of the **New dashboard** page, click to name the dashboard. Enter **GPU Pod Dashboard** for **Title** and click **Save dashboard** in the upper right corner. On the **Save dashboard** page displayed, click **Save**. In this example, the dashboard is named **GPU Pod Dashboard**. You can use another name.

Figure 8-6 Renaming a dashboard

Home > Dashboards > New dashboard > Settings	Close Save as Save dashboard
Settings	2
t] General 다 Annotations 圖 Variables & Links ↔ JSON Model	
Title	
GPU Pod Dashboard	
Description	

Step 2 Configure variables on the Grafana dashboard.

Grafana variables allow you to create dynamic, customizable, templated dashboards. By changing variable values within the dashboard, you can modify the queries and panels. There is no need to create separate dashboards for each service or metric. **Table 8-4** outlines the variables to configure in this example. Using **instance** as an example, this step demonstrates how to configure variables on a Grafana dashboard. You can refer to the **instance** example to add the variables listed in **Table 8-4** in order.

Variable	Description	PromQL Statement
instance	Pod name of dp. It is used to obtain GPU nodes.	label_values(up{job=~".*nvidia-gpu-device- plugin.*"},instance)
gpu_inde x	GPU card number, for example, 0, 1, or 2	label_values(cce_gpu_temperature{instance =~"\$instance"}, gpu_index)
namespa ce	Namespace where the GPU workload is in	label_values(gpu_pod_memory_used{instan ce=~"\$instance",gpu_index=~"\$gpu_index"}, namespace)
pod	Pod name of a GPU workload	label_values(gpu_pod_memory_used{instan ce=~"\$instance",gpu_index=~"\$gpu_index", namespace=~"\$namespace"}, pod)

Table 8-4 Variables

1. In the upper right corner of the **New dashboard** page, click **Settings** page, click the **Variables** tab and click **Add variable**.

Figure 8-7 Adding a variable



2. Configure **Select variable type** and **General**. For details, see **Figure 8-8** and **Table 8-5**.

Figure	8-8	Configurin	a Select	variable	type	and	Genera	I
Iguie	0-0	Conniguni	y select	variable	type	anu	Genera	ι

instance				
Select variable type				
Query	~			
General				
Name The name of the templat	e variable. (Ma	ax. 50 characte	ers)	
instance				
Label Optional display name				
Label name				
Description				
Descriptive text				
Show on dashboard				
Label and value	Value	Nothing		

Table 8-5 Configuration example for Select variable type and General

Paramete r	Example Value	Description
Select variable type	Query	Choose a variable type and select Query to obtain the values of variables in Table 8-4 from data source queries.
Name	instance	Configure a variable name.

3. Configure **Query options**. For details, see **Figure 8-9** and **Table 8-6**.



Query options
Data source
left prometheus v
Query
Query type O Classic query V
Classic Query 💿 label_values(up{job=~".*nvidia-gpu-device-plugin.*"},instance) 2
Optiona, if you want to extract part of a series name or metric node segment. Named capture groups can be used to separate the display text and value (see examples). /.*-(? <text>.*)-(?<value>.*)*/</value></text>
Sort How to sort the values of this variable
Alphabetical (asc) v 3
Refresh When to update the values of this variable
On dashboard load On time range change

Table 8-6 Conf	iguration exa	ample for Q	uery options
----------------	---------------	-------------	--------------

Parame ter	Example Value	Description
Query type	Classic query	Select the data query mode and display mode.
		In this example, select Classic query , which is a text-based query mode where you can write query statements (such as PromQL) directly to retrieve data.
Classic	label_values(up{jo	Enter a Prometheus query statement.
Query	b=~".*nvidia-gpu- device- plugin.*"},instance)	The query statement in this example is used to get the up metrics whose job labels contain nvidia-gpu-device-plugin from Prometheus and extract the instance label values of these metrics.
Sort	Alphabetical (asc)	Select the sorting mode of variable values. You can select a value as needed.
		In this example, Alphabetical (asc) is selected, which means the records are sorted in ascending alphabetical order.
Refresh	On time range change	Manage the update timing of variable values to specify when they are retrieved and refreshed from the data source. You can select a value as needed.
		In this example, On time range change is selected, meaning variable values are fetched from the data source again when the dashboard's time window changes.

4. Configure **Selection options**. For details, see **Figure 8-10** and **Table 8-7**.

Selection options
Multi-value
c ables multiple values to be selected at the same time
c include All option
c include All option
c include all variables

Custom all value
blank = auto
1

Figure 8-10 Configuring Selection options

Table 8-7 Configuration example for Selection options

Parame ter	Example Value	Description
Multi- value	Selected	After this function is enabled, you can select multiple values of a variable.
Include All option	Selected	After this function is enabled, the All option is added to the variable drop-down list, allowing you to select all values with just one click.
Preview of values	-	Variable value of instance . It is the pod name of dp.

Repeat the preceding steps to add variables listed in Table 8-4 in order. After

the variables are configured, appears on the right of the variables, showing a message that indicates the variables are not being referenced. These variables will be referenced when you configure the monitoring panel.

- 5. Click **Save dashboard** in the upper right corner. On the **Save dashboard** page displayed, click **Save** to save the variables.
- **Step 3** Configure the Grafana monitoring panels.

The following uses the monitoring panels listed in the below table as examples to describe how to create different types of monitoring panels. For details, see **Table 8-8**. You can set up different monitoring panels as needed. For details, see **Table 8-9**.

Panel Name	Description	Panel Type
Total Pods	Total number of GPU pods running on non-sharable GPUs	Stat
Pod Details	Summary of information such as the pod names, allocated GPU memory, and used GPU memory in a table	Table
Pod GPU Memory Allocation	Amount of GPU memory allocated to pods of the GPU workloads	Bar gauge
Pod GPU Memory Allocation Usage Rate	Usage of GPU memory allocated to pods of the GPU workloads	Gauge

Example of Stat Visualization: Total Pods

- 1. In the upper left corner of the current page, click **GPU Pod Dashboard**, click **Add** on the right, and choose **Visualization** from the drop-down list box to create a panel.
- 2. On the **Query** tab in the lower left corner of the **Edit panel** page, select the data source configured in **Step 2: Configure a Data Source for Grafana** for

Data source. Click on the left of **A**, click **Code** on the right of the expanded content, and enter the corresponding PromQL statements in **Metrics browser** to collect data. For details about PromQL statements, see **Table 8-9**.

Figure 8-11 Entering PromQL statements



3. In the upper right corner, switch the panel type to **Stat** and enter a panel title in **Panel options** > **Title**. In this example, the title is set to **Total Pods**. You can use another title.



Figure 8-12 Configuring the panel type and title

4. Configure the color mode and graph mode in **Stat styles**. In this example, **Color mode** is set to **Background Gradient** and **Graph mode** is set to **None**. You can customize the values as required.

Figure 8-13 Configuring Stat styles

Stat styles			
Orientation Layout orien	tation		
Auto	Horizontal	Vertical	
Text mode Control if na	me and value is	displayed or jus	st name
Auto			~
Color mode			
Backgrou	ind Gradient	1	~
Graph mode Stat panel g	raph / sparkline	mode	
None	Area	2	
Text alignme	ent		
Auto	Center		
Show perce	nt change		

5. Click **Save** in the upper right corner. On the **Save dashboard** page displayed, click **Save** again to save the panel configuration. In the upper right corner, click **Apply** to go to the dashboard page. The **Total Pods** panel has been created. The number in **Figure 8-14** is 3, which means that there are three pods in the cluster.



Figure 8-14 Checking the total number of pods

Example of Table Visualization: Pod Details

- 1. On the right, click **Add** and choose **Visualization** from the drop-down list to create a panel.
- In the upper right corner of the Edit panel page, switch the panel type to Table and enter a panel title in Panel options > Title. In this example, the title is set to Pod Details. You can use another title.
- 3. On the **Query** tab in the lower left corner, click **Add query** twice and enter three PromQL statements in **Metrics browser** of **A**, **B**, and **C** in order. For details about PromQL statements, see **Table 8-9**.

Figure 8-15 Entering PromQL statements

B Query 3 🖸 Transform data 0 🛕 Alert 0	
Data source O Pometheus V O Query options MD = auto = 1385 Interval = 15s	Query inspector
Kick start your query Explain •	ueries Builder Code
Metrics browser> Sum by (instance,pod,container,namespace)(gpu_pod_memory_total{instance~^\$instance",gpu_index~^\$gpu_index",namespace~^\$namespace", pod!=""}) /1024/1024/1024	pod=~"\$pod",
Options Legend: Auto Format: Time series Step: auto Type: Range Exemplars: false	
Kick start your query Explain	uerles Builder Code
Metrics browser> sum by (instance,pod,container,namespace)(gpu_pod_memory_used{instance"\$instance",gpu_index"\$gpu_index",namespace"\$namespace",pod=""}) /1024/1024/1024	od=~"\$pod",
Options Legend: Auto Format: Time series Step: auto Type: Range Exemplars: false	
Kick start your query Explain •	ueries Builder Code
Metrics browser > avg by (instance,pod,container,namespace,xgpu_index) (xgpu_memory_total{pod!=""})	
> Options Legend: Auto Format: Time series Step: auto Type: Range Exemplars: false	
+ Add query + Expression	

In the **Options** of **A**, **B**, and **C**, set **Format** to **Table** and **Type** to **Instant**. This setting will display the query results in a table and retrieve only the data at the current time point.

Figure 8-16 Changing the display mode and query mode of query results

🗄 Query 🚯 😋 Transform data 🚳	
Deta sources 🜔 promethieus - 🖉 > Query options M0 = auto = 2009 Interval = 155	
Kick start your query Erplan	
Matting browser > Luns by {Instance.pcd.container_namespace)(gpu_pod_memory_total{instance-"\$instance",gpu_index="\$gpu_index="\$gamespace="\$namespace";pod="pod"-pod"-pod"-"pod"-pod=""})/1824/1824/1824	
> dyslema Legend Modate Permat Type Auto - auto Table - Runga Runga	
Rick start your query Estals D	
Meters borner > sun by (instance.pod, container, namespace)(gou.god_newbry.used(instance="\$jintance", gou_index="\$	
v Options Lexend ⊖ Ministre ⊖ Format Type	
Auto - auto Tobo - Range Tester Both	
Rick slart your quary Esplah •	
Metrics browser > mg by (instance, pod, container, somespace, sppu_index) (ppu_moory_total(podi=""))	
- Epition	
Legend © Montep © Forul Type Luto • auto Table - Range Instant Both	
+ Add mary + Fromsion	

4. Switch to the **Transform data** tab, click **Add transformation**, and select **Merge series/tables** to merge the added PromQL statements.

Figure 8-17 Configuring Merge series/tables



5. In the lower part of the Transform data tab, click Add another transformation and select Organize fields by name to configure the table header. Additionally, Organize fields by name supports column hiding, column sequence adjustment, and column name setting. For details, see Figure 8-19. You can customize the settings.

Figure 8-18 Configuring Organize fields by name

dd another transformation	
rganize	1/29 × Show images
View all 🗸 Combine Calculate new fields Create new visualization Filter Perform spatial operations Reformat Re	
Organize fields by name Re-order, hide, or rename fields.	

Figure 8-19 Configuring the column name

memory allocated (G)	
memory used (G)	

6. Click **Save** in the upper right corner. On the **Save dashboard** page displayed, click **Save** again to save the panel configuration. In the upper right corner, click **Apply** to go to the dashboard page. The **Pod Details** panel has been created.

Figure 8-20 Checking pod details



Example of Bar Gauge Visualization: GPU Memory Allocated to Pods

- 1. On the right, click **Add** and choose **Visualization** from the drop-down list to create a panel.
- In the lower left corner of the Edit panel page, enter PromQL statements in Metrics browser of A. For details about PromQL statements, see Table 8-9. After entering the PromQL statements, select Custom in Options > Legend and enter {{Pod}}. This operation is used to replace the label in the legend with the value of the variable pod.

Figure 8-21 Adding PromQL statements



- 3. In the upper right corner, switch the panel type to **Bar gauge** and enter a panel title in **Panel options** > **Title**. In this example, the title is set to **Pod GPU Memory Allocation**. You can use another title.
- 4. In **Standard options**, set **Unit** to **Data > bytes (IEC)** to convert the data unit to GiB and set **Min** to **0** to display data more intuitively.

Figure 8-22 Configuring Standard options

~	Standard options	
	Unit	
1	bytes(IEC)	~
	Min Leave empty to calculate based on all values	
2	0	
	Max Leave empty to calculate based on all values	

5. Click **Save** in the upper right corner. On the **Save dashboard** page displayed, click **Save** again to save the panel configuration. In the upper right corner, click **Apply** to go to the dashboard page. The **Pod GPU Memory Allocation** panel has been created.





Dashboard Panel Example: The Usage of GPU Memory Allocated to Pods

- 1. On the right, click **Add** and choose **Visualization** from the drop-down list to create a panel.
- In the lower left corner of the Edit panel page, enter PromQL statements in Metrics browser of A. For details about PromQL statements, see Table 8-9. After entering the PromQL statement, select Custom in Options > Legend
and enter **{{Pod}}**. This operation is used to replace the label in the legend with the value of the variable pod.

Figure 8-24 Modifying a legend label



- 3. In the upper right corner, switch the panel type to **Gauge** and enter a panel title in **Panel options** > **Title**. In this example, the title is set to **Pod GPU Memory Allocation Usage Rate**. You can use another title.
- 4. In **Standard options**, set **Unit** to **Misc > Percent (0-100)** to convert the data unit to percentage ranging from 0% to 100%.

Figure 8-25 Changing data unit

~	Standard options	
	Unit	
	Percent (0-100)	•

5. Click **Save** in the upper right corner. On the **Save dashboard** page displayed, click **Save** again to save the panel configuration. In the upper right corner, click **Apply** to go to the dashboard page. The **Pod GPU Memory Allocation Usage Rate** panel has been created.

Figure 8-26 Checking the usage of GPU memory allocated to pods





PromQL Statement

This section provides PromQL statements for each monitoring panel in different dimensions. You can use these statements to create Grafana dashboards in various dimensions.

NOTE

If the allocated GPU memory or compute is 0, calculating their utilization is meaningless. The PromQL statements here exclude cases where GPU memory or compute allocation is 0 when calculating their utilization.

Panel Name	PromQL Statement
Total Pods	count(avg by (instance,pod,container,pod,container,namespace) (gpu_pod_memory_total{instance=~"\$instance",gpu_index=~"\$gp u_index",namespace=~"\$namespace",pod=~"\$pod",pod!=""}))
Pod Details	sum by (instance,pod,container,namespace) (gpu_pod_memory_total{instance=~"\$instance",gpu_index=~"\$gp u_index",namespace=~"\$namespace",pod=~"\$pod",pod!=""}) / 1024/1024/1024
	sum by (instance,pod,container,namespace) (gpu_pod_memory_used{instance=~"\$instance",gpu_index=~"\$gp u_index",namespace=~"\$namespace",pod=~"\$pod",pod!=""}) / 1024/1024/1024
	avg by (instance,pod,container,namespace,xgpu_index) (xgpu_memory_total{pod!=""})
Pod GPU Memory Allocation	sum by (instance,pod,container,pod,container,namespace) (gpu_pod_memory_total{instance=~"\$instance",gpu_index=~"\$gp u_index",namespace=~"\$namespace",pod=~"\$pod",pod!=""})
Used GPU Memory of Pods	sum by (instance,pod,container,pod,container) (gpu_pod_memory_used{instance=~"\$instance",gpu_index=~"\$gp u_index",namespace=~"\$namespace",pod=~"\$pod",pod!=""})
Pod GPU Memory Allocation Usage Rate	<pre>sum by (instance,pod,container,pod,container) (gpu_pod_memory_used{instance=~"\$instance",gpu_index=~"\$gp u_index",namespace=~"\$namespace",pod=~"\$pod",pod!=""})/sum by (instance,pod,container,pod,container) (gpu_pod_memory_total{instance=~"\$instance",gpu_index=~"\$gp u_index",namespace=~"\$namespace",pod=~"\$pod",pod!=""}!=0) * 100</pre>
GPU Compute Allocated to Pods	<pre>sum by (instance,pod,container,pod,container) (gpu_pod_core_percentage_total{instance=~"\$instance",gpu_index =~"\$gpu_index",namespace=~"\$namespace",pod=~"\$pod",pod! =""})</pre>
Used GPU Compute of Pods	<pre>sum by (instance,pod,container,pod,container) (gpu_pod_core_percentage_used{instance=~"\$instance",gpu_index =~"\$gpu_index",namespace=~"\$namespace",pod=~"\$pod",pod! =""})</pre>

Panel Name	PromQL Statement
Usage of GPU Compute Allocated to Pods	<pre>sum by (instance,pod,container,pod,container) (gpu_pod_core_percentage_used{instance=~"\$instance",gpu_index =~"\$gpu_index",namespace=~"\$namespace",pod=~"\$pod",pod! =""})/sum by (instance,pod,container,pod,container) (gpu_pod_core_percentage_total{instance=~"\$instance",gpu_index =~"\$gpu_index",namespace=~"\$namespace",pod=~"\$pod",pod! =""}!=0) * 100</pre>

Table 8-10 PromQL statements for GPU monitoring panels

Panel Name	PromQL Statement
Total GPUs	count(avg by (instance,gpu_index) (cce_gpu_temperature{instance=~"\$instance",gpu_index=~"\$gpu_i ndex",namespace=~"\$namespace",pod=~"\$pod"}))
GPU Details	avg by (instance,gpu_index,gpu_id,modelName) (cce_gpu_memory_total{instance=~"\$instance",gpu_index=~"\$gpu _index",namespace=~"\$namespace",pod=~"\$pod"}) / 1024/1024/1024
	<pre>sum by (instance,gpu_index,gpu_id,modelName) (gpu_pod_memory_total{instance=~"\$instance",gpu_index=~"\$gp u_index",namespace=~"\$namespace",pod=~"\$pod",pod!=""}) / 1024/1024/1024</pre>
Allocated GPU Memory	sum by (instance,gpu_index) (gpu_pod_memory_total{instance=~"\$instance",gpu_index=~"\$gp u_index",namespace=~"\$namespace",pod=~"\$pod",pod!=""})
Used GPU Memory	sum by (instance,gpu_index) (gpu_pod_memory_used{instance=~"\$instance",gpu_index=~"\$gp u_index",namespace=~"\$namespace",pod=~"\$pod",pod!=""})
Usage of Allocated GPU Memory	<pre>sum by (instance,gpu_index) (gpu_pod_memory_used{instance=~"\$instance",gpu_index=~"\$gp u_index",namespace=~"\$namespace",pod=~"\$pod",pod!=""})/sum by (instance,gpu_index) (gpu_pod_memory_total{instance=~"\$instance",gpu_index=~"\$gp u_index",namespace=~"\$namespace",pod=~"\$pod",pod!=""}!=0) * 100</pre>
Total GPU Memory	avg by (instance,gpu_index) (cce_gpu_memory_total{instance=~"\$instance",gpu_index=~"\$gpu _index",namespace=~"\$namespace",pod=~"\$pod"})

Panel Name	PromQL Statement
Allocated GPU Compute	<pre>sum by (instance,gpu_index) (gpu_pod_core_percentage_total{instance=~"\$instance",gpu_index =~"\$gpu_index",namespace=~"\$namespace",pod=~"\$pod",pod! =""})</pre>
Used GPU Compute	<pre>sum by (instance,gpu_index) (gpu_pod_core_percentage_used{instance=~"\$instance",gpu_index =~"\$gpu_index",namespace=~"\$namespace",pod=~"\$pod",pod! =""})</pre>
Usage of Allocated GPU Compute	<pre>sum by (instance,gpu_index) (gpu_pod_core_percentage_used{instance=~"\$instance",gpu_index =~"\$gpu_index",namespace=~"\$namespace",pod=~"\$pod",pod! =""})/sum by (instance,gpu_index) (gpu_pod_core_percentage_total{instance=~"\$instance",gpu_index =~"\$gpu_index",namespace=~"\$namespace",pod=~"\$pod",pod! =""}!=0) * 100</pre>
Total GPU Compute	count by (instance, gpu_index) (avg by (instance,gpu_index) (cce_gpu_temperature{instance=~"\$instance",gpu_index=~"\$gpu_i ndex",namespace=~"\$namespace",pod=~"\$pod"})) * 100

Table 8-11 PromQL	statements for	node	monitoring	panels
-------------------	----------------	------	------------	--------

Panel Name	PromQL Statement
Total GPU Nodes	count(avg by (instance) (cce_gpu_temperature{instance=~"\$instance",gpu_index=~"\$gpu_i ndex",namespace=~"\$namespace",pod=~"\$pod"}))
GPU Node Details	avg by (instance) (cce_gpu_memory_total{instance=~"\$instance",gpu_index=~"\$gpu _index",namespace=~"\$namespace",pod=~"\$pod"}) / 1024/1024/1024
	<pre>sum by (instance) (gpu_pod_memory_total{instance=~"\$instance",gpu_index=~"\$gp u_index",namespace=~"\$namespace",pod=~"\$pod",pod!=""}) / 1024/1024/1024</pre>
GPU Memory Allocated to Nodes	sum by (instance) (gpu_pod_memory_total{instance=~"\$instance",gpu_index=~"\$gp u_index",namespace=~"\$namespace",pod=~"\$pod",pod!=""})
Used GPU Memory of Nodes	sum by (instance) (gpu_pod_memory_used{instance=~"\$instance",gpu_index=~"\$gp u_index",namespace=~"\$namespace",pod=~"\$pod",pod!=""})

Panel Name	PromQL Statement
Usage of GPU Memory Allocated to Nodes	<pre>sum by (instance) (gpu_pod_memory_used{instance=~"\$instance",gpu_index=~"\$gp u_index",namespace=~"\$namespace",pod=~"\$pod",pod!=""})/sum by (instance) (gpu_pod_memory_total{instance=~"\$instance",gpu_index=~"\$gp u_index",namespace=~"\$namespace",pod=~"\$pod",pod!=""}!=0) * 100</pre>
Total GPU Memory of Nodes	<pre>sum by (instance) (avg by (instance,gpu_index) (cce_gpu_memory_total{instance=~"\$instance",gpu_index=~"\$gpu _index",namespace=~"\$namespace",pod=~"\$pod"}))</pre>
GPU Compute Allocated to Nodes	<pre>sum by (instance) (gpu_pod_core_percentage_total{instance=~"\$instance",gpu_index =~"\$gpu_index",namespace=~"\$namespace",pod=~"\$pod",pod! =""})</pre>
Used GPU Compute of Nodes	sum by (instance)(gpu_pod_core_percentage_used{pod!=""})
Usage of GPU Compute Allocated to Nodes	sum by (instance)(gpu_pod_core_percentage_used{pod!=""})/sum by (instance)(gpu_pod_core_percentage_total{pod!=""}!=0) * 100
Total GPU Compute of Nodes	count by (instance) (avg by (instance,gpu_index) (cce_gpu_temperature)) * 100

Table o 12 Home Statements for claster monitoring partets

Panel Name	PromQL Statement
GPU Memory Allocated to Clusters	sum (gpu_pod_memory_total{pod!=""})
Used GPU Memory of Clusters	sum (gpu_pod_memory_used{instance=~"\$instance",gpu_index=~"\$gp u_index",namespace=~"\$namespace",pod=~"\$pod",pod!=""})

Panel Name	PromQL Statement
Usage of GPU Memory Allocated to Clusters	<pre>sum (gpu_pod_memory_used{instance=~"\$instance",gpu_index=~"\$gp u_index",namespace=~"\$namespace",pod=~"\$pod",pod!=""})/sum (gpu_pod_memory_total{instance=~"\$instance",gpu_index=~"\$gp u_index",namespace=~"\$namespace",pod=~"\$pod",pod!=""}) * 100</pre>
Total GPU Memory of Clusters	sum (avg by (instance,gpu_index) (cce_gpu_memory_total{instance=~"\$instance",gpu_index=~"\$gpu _index",namespace=~"\$namespace",pod=~"\$pod"}))
GPU Compute Allocated to Clusters	<pre>sum (gpu_pod_core_percentage_total{instance=~"\$instance",gpu_index =~"\$gpu_index",namespace=~"\$namespace",pod=~"\$pod",pod! =""})</pre>
Used GPU Compute of Clusters	<pre>sum (gpu_pod_core_percentage_used{instance=~"\$instance",gpu_index =~"\$gpu_index",namespace=~"\$namespace",pod=~"\$pod",pod! =""})</pre>
Usage of GPU Memory Allocated to Clusters	<pre>sum (gpu_pod_core_percentage_used{instance=~"\$instance",gpu_index =~"\$gpu_index",namespace=~"\$namespace",pod=~"\$pod",pod! =""})/sum (gpu_pod_core_percentage_total{instance=~"\$instance",gpu_index =~"\$gpu_index",namespace=~"\$namespace",pod=~"\$pod",pod! =""}) * 100</pre>
Total GPU Compute of Clusters	count(avg by (instance,gpu_index) (cce_gpu_temperature{instance=~"\$instance",gpu_index=~"\$gpu_i ndex",namespace=~"\$namespace",pod=~"\$pod"})) * 100

9_{Cluster}

9.1 Suggestions on CCE Cluster Selection

When you use CCE to create a Kubernetes cluster, there are multiple configuration options and terms. This section compares the key configurations for CCE clusters and provides recommendations to help you create a cluster that better suits your needs.

Cluster Types

CCE supports CCE Turbo clusters and CCE standard clusters to meet your requirements. This section describes the differences between these two types of clusters.

Categ ory	Subcate gory	CCE Turbo Cluster	CCE Standard Cluster
Cluste r	Positioni ng	Next-gen container cluster designed for Cloud Native 2.0, with accelerated computing, networking, and scheduling	Standard cluster for common commercial use
	Node type	Deployment of VMs and bare metal servers	Deployment of VMs and bare metal servers
Netw orkin g	Model	Cloud Native 2.0 networks : for scenarios where requirements on performance are high and there are many containers Max networking scale: 2,000 nodes	 Cloud native 1.0 networks: for scenarios where requirements on performance are not high and there are not so many containers Tunnel network model VPC network model

 Table 9-1 Cluster types

Categ ory	Subcate gory	CCE Turbo Cluster	CCE Standard Cluster
	Performa nce	Flattens the VPC network and container network into one, achieving zero performance loss.	Overlays the VPC network with the container network, causing certain performance loss.
	Containe r network isolation	Associates pods with security groups. Unifies security isolation in and out the cluster via security groups' network policies.	 Tunnel network model: supports network policies for intra-cluster communications. VPC network model: supports no isolation.
Securi ty	Isolation	 Physical machine: runs Kata containers, allowing VM-level isolation. VM: runs common containers, isolated by cgroups. 	Runs common containers, isolated by cgroups.

Cluster Versions

Due to the fast iteration, many bugs are fixed and new features are added in the new Kubernetes versions. The old versions will be gradually eliminated. When creating a cluster, select the latest commercial version supported by CCE.

Network Models

This section describes the network models supported by CCE clusters. You can select one model based on your requirements.

NOTICE

After a cluster is created, the network model cannot be changed.

Dimensio n	Tunnel Network	VPC Network	Cloud Native Network 2.0
Applicatio n scenarios	 Low requirements on performance: As the container tunnel network requires additional VXLAN tunnel encapsulation, it has about 5% to 15% of performance loss when compared with the other two container network models. Therefore, the container tunnel network applies to the scenarios that do not have high performance requirements, such as web applications, and middle-end and back-end services with a small number of access requests. Large-scale networking: Different from the VPC network that is limited by the VPC route quota, the container tunnel network does not have any restriction 	 High performance requirements: As no tunnel encapsulation is required, the VPC network model delivers the performance close to that of a VPC network when compared with the container tunnel network model. Therefore, the VPC network model applies to scenarios that have high requirements on performance, such as AI computing and big data computing. Small- and medium-scale networks: Due to the limitation on VPC route tables, it is recommended that the number of nodes in a cluster be less than or equal to 1000. 	 High performance requirements: Cloud Native 2.0 networks use VPC networks to construct container networks, eliminating the need for tunnel encapsulation or NAT when containers communicate. This makes Cloud Native 2.0 networks ideal for scenarios that demand high bandwidth and low latency, such as live streaming and e-commerce flash sales. Large-scale networking: Cloud Native 2.0 networks support a maximum of 2,000 ECS nodes and 100,000 pods.

Table 9-2 Network model comparison

Dimensio n	Tunnel Network	VPC Network	Cloud Native Network 2.0
	on the infrastructure. In addition, the container tunnel network controls the broadcast domain to the node level. The container tunnel network supports a maximum of 2000 nodes.		
Core technolog y	OVS	IPVLAN and VPC route	VPC ENI/sub-ENI
Applicable clusters	CCE standard cluster	CCE standard cluster	CCE Turbo cluster
Container network isolation	Kubernetes native NetworkPolicy for pods	No	Pods support security group isolation.
Interconne cting pods to a load balancer	Interconnected through a NodePort	Interconnected through a NodePort	Directly interconnected using a dedicated load balancer Interconnected using a shared load balancer through a NodePort
Managing container IP addresses	 Separate container CIDR blocks needed Container CIDR blocks divided by node and dynamically added after being allocated 	 Separate container CIDR blocks needed Container CIDR blocks divided by node and statically allocated (the allocated CIDR blocks cannot be changed after a node is created) 	Container CIDR blocks divided from a VPC subnet (You do not need to configure separate container CIDR blocks.)

Dimensio n	Tunnel Network	VPC Network	Cloud Native Network 2.0
Network performan ce	Performance loss due to VXLAN encapsulation	No tunnel encapsulation, and cross-node traffic forwarded through VPC routers (The performance is so good that is comparable to that of the host network, but there is a loss caused by NAT.)	Container network integrated with VPC network, eliminating performance loss
Networkin g scale	A maximum of 2000 nodes are supported.	Suitable for small- and medium-scale networks due to the limitation on VPC route tables. It is recommended that the number of nodes be less than or equal to 1000. Each time a node is added to the cluster, a route is added to the VPC route tables (including the default and custom ones). Evaluate the cluster scale that is limited by the VPC route tables before creating the cluster. For details about route tables, see Constraints .	A maximum of 2000 nodes are supported. In a cloud-native network 2.0 cluster, containers' IP addresses are assigned from VPC CIDR blocks, and the number of containers supported is restricted by these blocks. Evaluate the cluster's scale limitations before creating it.

For more information, see **Container Network Models Overview**.

Cluster CIDR Blocks

There are node CIDR blocks, container CIDR blocks, and Service CIDR blocks in CCE clusters. When planning network addresses, note that:

• These three types of CIDR blocks cannot overlap with each other. Otherwise, a conflict will occur. All subnets (including those created from the secondary CIDR block) in the VPC where the cluster resides cannot conflict with the container and Service CIDR blocks.

- There are sufficient IP addresses in each CIDR block.
 - The IP addresses in a node CIDR block must match the cluster scale. Otherwise, nodes cannot be created due to insufficient IP addresses.
 - The IP addresses in a container CIDR block must match the service scale. Otherwise, pods cannot be created due to insufficient IP addresses.

In complex scenarios, for example, multiple clusters use the same VPC or clusters are interconnected across VPCs, determine the number of VPCs, the number of subnets, the container CIDR blocks, and the communication modes of the Service CIDR blocks. For details, see **Planning CIDR Blocks for a Cluster**.

Service Forwarding Modes

kube-proxy is a key component of a Kubernetes cluster. It is responsible for load balancing and forwarding between a Service and its backend pod.

CCE supports the iptables and IPVS forwarding modes.

- IPVS allows higher throughput and faster forwarding. It applies to scenarios where the cluster scale is large or the number of Services is large.
- iptables is the traditional kube-proxy mode. This mode applies to the scenario where the number of Services is small or there are a large number of short concurrent connections on the client.

If high stability is required and the number of Services is less than 2000, the iptables forwarding mode is recommended. In other scenarios, the IPVS forwarding mode is recommended.

For details, see Comparing iptables and IPVS.

Node Specifications

The minimum specifications of a node are 2 vCPUs and 4 GiB memory. Evaluate based on service requirements before configuring the nodes. However, using many low-specification ECSs is not the optimal choice. The reasons are as follows:

- The upper limit of network resources is low, which may result in a single-point bottleneck.
- Resources may be wasted. If each container running on a low-specification node needs a lot of resources, the node cannot run multiple containers and there may be idle resources in it.

Advantages of using large-specification nodes are as follows:

- The upper limit of the network bandwidth is high. This ensures higher resource utilization for high-bandwidth applications.
- Multiple containers can run on the same node, and the network latency between containers is low.
- The efficiency of pulling images is higher. This is because an image can be used by multiple containers on a node after being pulled once. Low-specifications ECSs cannot respond promptly because the images are pulled many times and it takes more time to scale these nodes.

Additionally, select a proper vCPU/memory ratio based on your requirements. For example, if a service container with large memory but fewer CPUs is used,

configure the specifications with the vCPU/memory ratio of 1:4 for the node where the container resides to reduce resource waste.

In a CCE Turbo cluster, pods use elastic or supplementary network interfaces on nodes. The maximum number of pods that can be created on a node is related to the number of network interfaces that can be used by the node. So you need to evaluate the number of network interfaces that can be used by the node. This determines the maximum number of pods that can run on the node. For details, see **Maximum Number of Pods That Can Be Created on a Node**. For details about the number of network interfaces supported by different node flavors, see **Node Specifications**.

Container Engines

CCE supports the containerd and Docker container engines. **containerd is recommended for its shorter traces, fewer components, higher stability, and less consumption of node resources**. Since Kubernetes 1.24, Dockershim is removed and Docker is no longer supported by default. For details, see **Kubernetes is Moving on From Dockershim: Commitments and Next Steps**. CCE clusters of v1.27 do not support the Docker container engine.

Use containerd in typical scenarios. The Docker container engine is supported only in the following scenarios:

- Docker in Docker (usually in CI scenarios)
- Running the Docker commands on the nodes
- Calling Docker APIs

Node OS

Service container runtimes share the kernel and underlying calls of nodes. To ensure compatibility, select a Linux distribution version that is the same as or close to that of the final service container image for the node OS.

9.2 Creating an IPv4/IPv6 Dual-Stack Cluster in CCE

This section describes how to set up a VPC with IPv6 CIDR block and create a cluster and nodes with an IPv6 address in the VPC, so that the nodes can access the Internet.

Overview

IPv6 addresses are used to deal with the problem of IPv4 address exhaustion. If a worker node (such as an ECS) in the current cluster uses IPv4, the node can run in dual-stack mode after IPv6 is enabled. Specifically, the node has both IPv4 and IPv6 addresses, which can be used to access the intranet or public network.



Figure 9-1 Diagram of an IPv4/IPv6 dual-stack enabled cluster

Application Scenarios

- If your application needs to provide services for users who use IPv6 clients, you can use IPv6 EIPs or IPv4/IPv6 dual-stack.
- If your application needs to both provide Services for users who use IPv6 clients and analyze the access request data, you can use only the IPv4 and IPv6 dual-stack function.
- If internal communication is required between your application systems or between your application system and another system (such as the database system), you can use only the IPv4 and IPv6 dual-stack function.

For details about the dual-stack, see **IPv4 and IPv6 Dual-Stack Network** and **IPv6 EIP**.

Notes and Constraints

• Clusters that support IPv4/IPv6 dual-stack:

Cluster Type	Cluster Network Model	Version	Remarks
CCE standard cluster	Container tunnel network	v1.15 or later	IPv4/IPv6 dual-stack will be generally available for clusters of v1.23.

9 Cluster

Cluster Type	Cluster Network Model	Version	Remarks
CCE Turbo cluster	Cloud Native 2.0 network	v1.23.8-r0 or later v1.25.3-r0 or later	Currently, Kata containers do not support IPv4/IPv6 dual-stack. Only ECS-VM or ECS-physical server (c6.22xlarge.4.physical or c7.32xlarge.4.physical) supports IPv4/IPv6 dual-stack.

- Worker nodes and master nodes in Kubernetes clusters use IPv4 addresses to communicate with each other.
- When there is a **DNAT** Service in a cluster, only IPv4 addresses are supported.
- Only one IPv6 address can be bound to each network interface.
- When IPv4/IPv6 dual-stack is enabled for the cluster, DHCP unlimited lease cannot be enabled for the selected node subnet.
- If a dual-stack cluster is used, do not change the load balancer protocol version on the ELB console.
- ELB dual-stack can be used in only CCE Turbo clusters with the following restrictions.

Application Scenario	Dedicated Load Balancer	Shared Load Balancer
LoadBalancer ingress	Dual stack is supported. Layer-7 dedicated load balancers can only communicate with their backend servers using IPv4. For details, see Does ELB Support IPv6 Networks? If an ingress uses IPv6/IPv4 dual-stack, related alarms will be generated. (Backends with IPv6 addresses cannot be added to the associated load balancer.) You can view related alarms by referring to the events of the corresponding ingress.	Only IPv4 is supported.
Nginx ingress	 Dual-stack is supported when the following conditions are met: For clusters from v1.19 to v1.23, nginx-ingress of v2.1.7 or later supports dual-stack. For clusters of v1.25 or later, nginx-ingress of v2.2.5 or later supports dual-stack. 	Only IPv4 is supported.

Application Scenario	Dedicated Load Balancer	Shared Load Balancer
LoadBalancer Service	 Layer 7 (HTTP/HTTPS): Dual-stack is supported. Layer-7 dedicated load balancers can only communicate with their backend servers using IPv4. For details, see Does ELB Support IPv6 Networks? If a Service uses IPv6/IPv4 dual-stack, related alarms will be generated. (Backends with IPv6 addresses cannot be added to the associated load balancer.) You can view related alarms by referring to the events of the corresponding Service. To avoid alarms, you can select the IPv4 protocol when creating a Service and select a dedicated Layer-7 load balancer with dual-stack enabled. Layer 4 (TCP/UDP): Dual-stack is supported. 	Only IPv4 is supported.

Step 1: Create a VPC and Subnet

Before creating your VPCs, determine how many VPCs, the number of subnets, and what IP address ranges you will need. For details, see **Network Planning**.

NOTE

- The basic operations for IPv4 and IPv6 dual-stack networks are the same as those for IPv4 networks. Only some parameters are different.
- For details about the IPv6 billing policy, supported ECS types, and supported regions, see IPv4 and IPv6 Dual-Stack Network.

Perform the following operations to create a VPC named **vpc-ipv6** and its default subnet named **subnet-ipv6**.

- 1. Log in to the management console.
- 2. Click in the upper left corner of the management console and select a region and a project.
- 3. Choose Networking > Virtual Private Cloud.
- 4. Click Create VPC.
- 5. Configure the VPC and subnet following instructions. For details about the mandatory parameters, see **Table 9-3** and **Table 9-4**. For details about other parameters, see **Creating a VPC and Subnet**.

When configuring a subnet, select **Enable** for **IPv6 CIDR Block** to automatically allocate an IPv6 CIDR block to the subnet. IPv6 cannot be disabled after the subnet is created. Currently, you are not allowed to specify a custom IPv6 CIDR block.

Parame	ter	Description	Example Value
Region		Specifies the desired region. Regions are geographic areas that are physically isolated from each other. The networks inside different regions are not connected to each other, so resources cannot be shared across different regions. For lower network latency and faster access to your resources, select the region nearest you.	AP- Singapore
Name		VPC name.	vpc-ipv6
IPv4 CID Block)R	Specifies the Classless Inter-Domain Routing (CIDR) block of the VPC. The CIDR block of a subnet can be the same as the CIDR block for the VPC (for a single subnet in the VPC) or a subset (for multiple subnets in the VPC). The following CIDR blocks are supported: 10.0.0/8-24 172.16.0.0/12-24 192.168.0.0/16-24	192.168.0.0 /16
Enterpri Project	se	When creating a VPC, you can add the VPC to an enabled enterprise project. An enterprise project facilitates project-level management and grouping of cloud resources and users. The name of the default project is default . For details about how to create and manage enterprise projects, see <i>Enterprise</i> <i>Management User Guide</i> .	default

Table 9-4 Subnet parameters

Parameter	Description	Example Value
Subnet Name	Specifies the subnet name.	subnet-ipv6
AZ	An AZ is a geographic location with independent power supply and network facilities in a region. AZs are physically isolated, and AZs in the same VPC are interconnected through an internal network.	AZ 2

Parameter	Description	Example Value
IPv4 CIDR Block	Specifies the IPv4 CIDR block for the subnet. This value must be within the VPC CIDR range.	192.168.0.0 /24
IPv6 CIDR Block	Select Enable for IPv6 CIDR Block . An IPv6 CIDR block will be automatically assigned to the subnet. IPv6 cannot be disabled after the subnet is created. Currently, you are not allowed to specify a custom IPv6 CIDR block.	N/A
Associated Route Table	Specifies the default route table to which the subnet will be associated. You can change the route table to a custom route table.	Default
Advanced Set	ttings	
Gateway	Specifies the gateway address of the subnet. This IP address is used to communicate with other subnets.	192.168.0.1
DNS Server Address	By default, two DNS server addresses are configured. You can change them if necessary. When multiple IP addresses are available, separate them with a comma (,).	100.125.x.x
IPv4 DHCP Lease Time	Specifies the period during which a client can use an IP address automatically assigned by the DHCP server. After the lease time expires, a new IP address will be assigned to the client. If a DHCP lease time is changed, the new lease automatically takes effect when half of the current lease time has passed. To make the change take effect immediately, restart the ECS or log in to the ECS to cause the DHCP lease to automatically renew. CAUTION When IPv4/IPv6 dual-stack is enabled for the cluster, DHCP unlimited lease cannot be enabled for the selected node subnet.	365 days or 300 hours

6. Click Create Now.

Step 2: Create a CCE Cluster

Creating a CCE cluster

- 1. Log in to the CCE console and create a cluster.
 - Complete the network settings as follows: (For other configurations, see **Buying a CCE Cluster**.)
 - **VPC**: Select the created VPC **vpc-ipv6**.
 - **Default Node Subnet**: Select a subnet with IPv6 enabled.

- IPv6: Enable this function. After this function is enabled, cluster resources, including nodes and workloads, can be accessed through IPv6 CIDR blocks.
- Network Model: Select Tunnel network.
- Network Policies: This function is enabled by default to restrict the objects that can be accessed by pods. For details, see Configuring Network Policies to Restrict Pod Access.
- Container CIDR Block: A proper mask must be set for the container CIDR block. The mask determines the number of available nodes in the cluster. If the mask of the container CIDR block in the cluster is set improperly, there will be only a small number of available nodes in the cluster.

Figure 9-2 Configuring network settings

Chuster Natural	
Cluster Network	
VPC (2)	vpc-ipv6 (192.168.0.016) V C Create VPC C
	The setting cannot be modified after the cluster is created. Select a VPC to provide CIDR blocks for the master and worker nodes in your cluster.
Default Node Subnet (?)	submel-42ed (192 168 0.0/24 2407 c080.1110 166. <i>t</i> V Q Create Submet (?
	Available IPv4 Addresses in the Subnet: 250
	The setting cannot be modified after the cluster is created. Select a subnet in your VPC. Nodes in the cluster will use the IP addresses in this subnet by default. If needed, you can modify the node subnet during the creation of a node or node pool. Master ranks in a rank animated by CCC softguere works ranks.
	Creation an IRVeITING Dual-Stark Chester
IPV6 (?)	Voderbrit services can be accessed through an IPV-4IPV-6 dual stack network. To experience a dual-stack network smoothly, use CCE Turbo clusters.
Default Node Security	Auto generate Use existing
Group 💿	Two default security groups will be automatically created for your cluster, one for the master nodes and the other for the worker nodes. The security group for the master nodes is named (Cluster name)-ccc-control-(Random ID), and that of the worker nodes is named (Cluster name)-coce-control-(Random ID). Security Group Rules
Container Network	
Network Model	VPC network Tunnel network
	Suitable for services that require high performance.
	Interview a conserver of the second and the second
	The VPC routing table quota. performance.
	The setting cannot be modified after the cluster is created.Model framework used by the container network in the cluster.
Notwork Policion	
Network Folicies	This function allows for network control based on policies. Learn more
Container CIDR Block	Manually set Auto select How to plan CIDR blocks?
	▲ The value cannot be changed after creation. For cross-VPC passthrough networking, make sure the container CIDR block does not overlap with the VPC CIDR block to be accessed. Learn more
Service Network	
Service CIDR Block	
	Max. Services allowed by this CIDR block: 65,536
	The setting cannot be modified after the cluster is created. Configure an IP address range for ClusterIP Services in your cluster.
Request Forwarding	iptables IPVS Comparing iptables and IPVS
	The setting cannot be modified after the cluster is created. IPVS can be used to balance load between Services and their backend pods with high performance. IPVS is suitable for large-scale Services and pods.
	CentOS is prone to IPVS- or Commarck-related stability issues on a heavy-load network. Therefore, do not use CentOS as the operating system of cluster nodes when selecting IPVS. Vew Details
	When IPVS is used, if an ingress and a Service use the same ELB load balancer, the ingress cannot be accessed from the nodes and containers in the cluster.

2. Create a node.

The CCE console displays the nodes that support IPv6. You can directly select a node. For details, see **Creating a Node**.

After the creation is complete, access the cluster details page. Then, click the node name to go to the ECS details page and view the automatically allocated IPv6 address.

Step 3: Buy a Shared Bandwidth and Adding an IPv6 Address to It

By default, the IPv6 address can only be used for private network communication. If you want to use this IPv6 address to access the Internet or be accessed by IPv6 clients on the Internet, buy a shared bandwidth and add the IPv6 address to it.

If you already have a shared bandwidth, you can add the IPv6 address to the shared bandwidth without buying one.

Buying a Shared Bandwidth

- 1. Log in to the management console.
- 2. Click in the upper left corner of the management console and select a region and a project.
- 3. Choose Service List > Networking > Virtual Private Cloud.
- 4. In the navigation pane, choose **Elastic IP and Bandwidth > Shared Bandwidths**.
- 5. In the upper right corner, click **Buy Shared Bandwidth**. On the displayed page, configure parameters following instructions.

Table 9-5 Parameters

Paramete r	Description	Example Value
Billing Mode	Specifies the billing mode of a shared bandwidth. The billing mode can be:	Yearly/ Monthly
	• Yearly/Monthly: You pay for the bandwidth by year or month before using it. No charges will be incurred for the bandwidth during its validity period.	
	• Pay-per-use : You pay for the bandwidth based on the amount of time you use the bandwidth.	
Region	Specifies the desired region. Regions are geographic areas that are physically isolated from each other. The networks inside different regions are not connected to each other, so resources cannot be shared across different regions. For lower network latency and faster access to your resources, select the region nearest you.	AP- Singapore -
Billed By	Specifies the shared bandwidth billing factor.	Select Bandwidt h.
Bandwidt h	Specifies the shared bandwidth size in Mbit/s. The minimum bandwidth that can be purchased is 5 Mbit/s.	10
Name	Specifies the name of the shared bandwidth.	Bandwidth -001

Paramete r	Description	Example Value
Enterprise Project	When assigning the shared bandwidth, you can add the shared bandwidth to an enabled enterprise project.	default
	An enterprise project facilitates project-level management and grouping of cloud resources and users. The name of the default project is default .	
	For details about how to create and manage enterprise projects, see <i>Enterprise Management User Guide</i> .	
Required Duration	Specifies the required duration of the shared bandwidth to be purchased. Configure this parameter only in yearly/monthly billing mode.	2 months

6. Click Next.

Adding an IPv6 Address to a Shared Bandwidth

 On the shared bandwidth list page, locate the row containing the target shared bandwidth and choose More > Add Public IP Address in the Operation column.

Figure 9-3 Adding an IPv6 address to a shared bandwidth

Name	Status	Bandwidth (Mbit/s)	Billing Mo 7	Billed By 🐺	Public IP Address	Enterprise Project	Operation
	 Normal 	5	Pay-per-use	Bandwidth	View All(3)	default	Modify Bandwidth More 💌
							Add Public IP Address Remove Public IP Address

2. Add the IPv6 address to the shared bandwidth.

×

Figure 9-4 Adding an IPv4/IPv6 dual-stack network interface

After an EIP or 5 Mbit/s. The EIP or dua Yearly/monthl	dual-stack NIC IPv6 address Il-stack NIC IPv6 address cha y EIPs cannot be added to a	s is added to the sha arging mode will be ir shared bandwidth.	red bandwidth, the EIF nvalid and no extra tra	fic or dual-stack NIC IP	r6 address will use the sha s will be charged.	ared bandwidth
Shared Bandwidth						
	You can add 148 more p A maximum of 150 publi	ublic IP addresses to ic IP addresses can b	o the shared bandwidt be added to the shared	h. I bandwidth. Increase	e quota	
Public IP Address	EIP	Dual-stack NIC	CIPv6 address			
	All projects 🔹	All	EIP 🔻		Q Search by Tag	× C
	EIP	Status	EIP Type	Bandwidth (M	Associated Instance	Enterprise Proj
		Ø Bound	Dynamic BGP	1		default
		Ø Bound	Dynamic BGP	5		default

3. Click OK.

Verifying the Result

Log in to an ECS and ping an IPv6 address on the Internet to verify the connectivity. **ping6 ipv6.baidu.com** is used as an example here. The execution result is displayed in **Figure 9-5**.

Figure 9-5 Result verification

```
root@ecs-tang:~# ping6 ipv6.baidu.com

PING ipv6.baidu.com(2400:da00:2::29) 56 data bytes

64 bytes from 2400:da00:2::29: icmp_seq=1 ttl=42 time=45.6 ms

64 bytes from 2400:da00:2::29: icmp_seq=2 ttl=42 time=45.1 ms

64 bytes from 2400:da00:2::29: icmp_seq=3 ttl=42 time=44.8 ms

64 bytes from 2400:da00:2::29: icmp_seq=4 ttl=42 time=45.1 ms
```

9.3 Creating a Custom CCE Node Image

Custom CCE node images are created using the open-source tool **HashiCorp Packer** and the **open-source plug-in**. The cce-image-builder template is provided to help you quickly build images.

Packer is used to create custom container images. It offers builders, provisioners, and post-processors that can be flexibly combined to automatically create image files concurrently through JSON or HCL template files.

Packer has the following advantages:

1. Automatic build process: You can use Packer configuration files to specify and automate the build process.

- 2. High compatibility with cloud platforms: Packer can interconnect with most cloud platforms and various third-party plug-ins.
- 3. Easy-to-use configuration files: Packer configuration files are simple and intuitive to write and read. Parameter definitions are easy to understand.
- 4. Diverse image build functions: Common functional modules are supported. For example, the provisioner supports the shell module in remote script execution, the file module in remote file transfer, and the breakpoint module for process pauses.

Notes and Constraints

• Only the following OSs and kernels are supported:

OS	Kernel Version
CentOS 7.6	≥ 3.10.0-1160.15.2.el7.x86_64
Ubuntu 18.04	≥ 4.15.0-136-generic
Ubuntu 22.04	≥ 5.15.0-86-generic
EulerOS 2.9 (x86)	≥ 4.18.0-147.5.1.6.h1071.eulerosv2r9.x 86_64
EulerOS 2.9 (Arm)	≥ 4.19.90- vhulk2103.1.0.h1060.eulerosv2r9.aar ch64
Huawei Cloud EulerOS 2.0 (x86)	≥ 5.10.0-60.18.0.50.r1002_48.hce2.x86_ 64
Huawei Cloud EulerOS 2.0 (Arm)	≥ 5.10.0-60.18.0.50.r1002_48.hce2.aarc h64

- Suggestions on using CCE node images:
 - You are advised to use the default node images maintained by CCE.
 These images have passed strict tests and are updated in a timely manner. They provide better compatibility, stability, and security.
 - To use the custom image function in certain scenarios, you can use the node image IDs provided by CCE to create custom images. You cannot directly export an ECS public image or the image of a node that is currently running as a custom image.
- When customizing an image, exercise caution when modifying kernel parameters. Any improper kernel parameter modification will deteriorate the system running efficiency. For details about the reference values, see **Modifying Node Kernel Parameters**.

Modifying the following kernel parameters will affect the system performance: tcp_keepalive_time, tcp_max_tw_buckets, somaxconn, max_user_instances, max_user_watches, netdev_max_backlog, net.core.wmem_max, and net.core.rmem_max. To modify node kernel parameters, fully verify the modification in a test environment before applying the modification to the production environment.

Precautions

- Before you create an image, prepare:
 - An ECS executor: An ECS x86 server is used as the Linux executor. You are advised to select Huawei Cloud EulerOS 2.0 and bind an EIP to the server so that it can access the public network and be used to install Packer.
 - Authentication credentials: Obtain the AK/SK of the tenant or user with required permissions. For details, see How Do I Obtain an Access Key (AK/SK).
 - Security group: Packer creates a temporary ECS and uses a key pair to log in to the ECS using SSH. Ensure that **TCP:22** is enabled in the security group. For details, see Security Group Configuration Examples.
- When you create a custom node image, make sure:
 - You follow the instructions in this section to prevent unexpected problems.
 - You have the **sudo root** or **root** permissions required to log in to VMs created from base images.
- When the creation is complete:
 - The image creation process uses certain charging resources, including ECSs, EVS disks, EIPs, bandwidth, and IMS images. These resources are automatically released when the image is successfully created or fails to be created. Release the resources in time to ensure no charges are incurred unexpectedly.

Obtaining an Image ID

- **Step 1** Log in to the CCE console and click the cluster name to access the cluster console. In the navigation pane, choose **Nodes**.
- **Step 2** Click **Create Node** in the upper right corner and select **OS** to **Private image**.
- **Step 3** Click **View CCE base image information**. In the displayed dialog box, copy the image ID.

NOTICE

The image ID varies depending on the region. If the region is changed, obtain the image ID again.

Figure 9-6 Obtaining an Image ID

Selected specifications	General computing-plus c7.large 2 2 vCPUs 4 GIB AZ1
Container Engine	contained
	P Important: CCE 127 and later clusters do not support the Docker container engine. We recommend using Containerd, which provides a better user experience and more powerful features. Container Engine Description []
OS	Public image Private image
	Recommend image-by-packer-20230413160908 C
	The image must contain the cce=cce tag (which can be added in IMS). Qualifying tags are displayed. Learn how to create a private image [2] View CCE base image information
Node Name	tot=1056
	Enter 1 to 56 characters, starting with a lowercase letter and not ending with a hyphen (-). Only lowercase letters, digits, and hyphens (-) are allowed.

----End

Creating a Node Image

Step 1 Download cce-image-builder.

Log in to the ECS executor, download and decompress cce-image-builder. wget https://cce-north-4.obs.cn-north-4.myhuaweicloud.com/cce-image-builder/cce-image-builder.tgz

```
tar zvxf cce-image-builder.tgz
cd cce-image-builder/
```

NOTE

The cce-image-builder contains:

- turbo-node.pkr.hcl # Packer configuration template used for creating the image. For details about how to modify the template, see **Step 3**.
- scripts/* # CCE image creation preset in the template. Do not modify it. Otherwise, the image might become unavailable.
- user-scripts/* **# Custom package script directory preset in the template.** Take example.sh as an example. When you create a custom image, the image is automatically uploaded to the temporary server and executed.
- user-packages/* **# Custom package directory preset in the template**. Take **example.package** as an example. When you create a custom image, the image is automatically uploaded to **/tmp/example.package** in the temporary server.

Step 2 Install Packer.

Manually download and install **HashiCorp Packer**. For details, see the **official documentation**.

NOTE

The Packer version needs to be 1.10.0.

Install packer: (A Huawei Cloud EulerOS 2.0 executor is taken as an example.)

wget https://releases.hashicorp.com/packer/1.10.0/packer_1.10.0_linux_amd64.zip unzip packer_1.10.0_linux_amd64.zip cp -f packer /usr/bin/packer

Step 3 Define Packer template parameters.

The **cce-image-builder/turbo-node.pkr.hcl** file defines the process of building an image using Packer. For details, see **Packer Documentation**.

D NOTE

• Parameters of variables or variable

turbo-node.pkr.hcl defines the parameters required in the process of building an image. You can configure the parameters based on the live environment. For details, see **Table 1**.

• Parameter of packer

required_plugins defines the add-on dependency of Packer, including the add-on source and version range. When you run **packer init**, the add-on is automatically downloaded and initialized.

```
packer {
  required_plugins {
    huaweicloud = {
    version = ">= 1.0.4"
    source = "github.com/huaweicloud/huaweicloud"
    }
  }
}
```

• Parameter of **source**

The preceding defined variables are referred to automatically configure the parameters required for creating an ECS.

• Parameter of build

The scripts are executed from top to bottom. Common modules such as the file upload module and script execution shell module are supported. The corresponding scripts and files are stored in the **user-scripts** and **user-packages** directories, respectively, in **cce-image-builder**.

```
Example:
build {
 sources = ["source.huaweicloud-ecs.builder"]
# Example:
 provisioner "file" {
  source
           = "<source file path>"
  destination = "<destination file path>"
 }
 provisioner "shell" {
  scripts = [
    "<source script file: step1.sh>",
    "<source script file: step2.sh>"
]
}
 provisioner "shell" {
  inline = ["echo foo"]
 }
ļ
```

Step 4 Configure environment variables.

Configure the following environment variables on the executor: export REGION_NAME=xxx export IAM_ACCESS_KEY=xxx export IAM_SECRET_KEY=xxx export ECS_VPC_ID=xxx export ECS_NETWORK_ID=xxx export ECS_SECGRP_ID=xxx export CCE_SOURCE_IMAGE_ID=xxx export PKR_VAR_ecs_flavor=xxx

	Table 9	- 6 Vari	ables co	onfigui	ration
--	---------	-----------------	----------	---------	--------

Parameter	Description	Remarks
REGION_NAME	Region to which the project belongs	To obtain the region information, go to My Credentials .
IAM_ACCESS_K EY	Access key for user authentication	Apply for a temporary AK and delete it when the image is built successfully.
IAM_SECRET_K EY	Secret key for user authentication	Apply for a temporary SK and delete it when the image is built successfully.
ECS_VPC_ID	VPC ID	Used by the temporary ECS server, which must be the same as that of the executor
ECS_NETWORK _ID	Network ID of the subnet	Used by the temporary ECS server. It is recommended that the value be the same as that of the executor. It is not the subnet ID.
ECS_SECGRP_I D	Security group ID	Used by the temporary ECS. The public IP address of the executor must be allowed to pass through port 22 in the inbound direction of the security group to ensure that the executor can log in to the temporary ECS using SSH.
CCE_SOURCE_I MAGE_ID	Latest CCE node image ID	For details, see Obtaining an Image ID.
PKR_VAR_ecs_fl avor	Specifications of a temporary ECS	Enter a node flavor supported by CCE. The recommended flavor is 2 vCPUs and 4 GiB memory or higher. For details about flavor names, see A Summary List of x86 ECS Specifications and A Summary List of Kunpeng ECS Specifications.

Note: Retain the default values of other parameters. To change the values, refer to the description in the variable definition in **turbo-node.pkr.hcl** and configure the value using environment variables.

Use the ECS flavor variable **ecs_az** as an example. If no AZ is specified, select a random AZ. If you want to specify an AZ, configure an environment variable. The same applies to other parameters.

```
# export PKR_VAR_<variable name>=<variable value>
export PKR_VAR_ecs_az=xxx
```

Step 5 Customize scripts and files.

Compile scripts and files by referring to the file and shell modules defined by the **build** field in the **pkr.hcl** file, and store the scripts and files in the **user-scripts** and **user-packages** directories in **cce-image-builder**.

NOTICE

When customizing an image, exercise caution when modifying kernel parameters. Any improper kernel parameter modification will deteriorate the system running efficiency. For details about the reference values, see **Modifying Node Kernel Parameters**.

Modifying the following kernel parameters will affect the system performance: tcp_keepalive_time, tcp_max_tw_buckets, somaxconn, max_user_instances, max_user_watches, netdev_max_backlog, net.core.wmem_max, and net.core.rmem_max.

To modify node kernel parameters, fully verify the modification in a test environment before applying the modification to the production environment.

Step 6 Create a custom image.

After custom parameter settings, create an image. The creation will take 3 to 5 minutes.

make image

NOTE

In the encapsulation script **packer.sh**:

- Automatic access of hashicorp.com by Packer is disabled by default for privacy protection and security purposes.
 export CHECKPOINT_DISABLE=false
- The debugging detailed logs option is enabled by default for better visibility and traceability. The local Packer build logs packer_{timestamp}.log is specified so that the logs can be packed to the /var/log/ directory during build. If sensitive information is involved, remove the related logic. export PACKER_LOG=1 export PACKER_BUILD_TIMESTAMP=\$(date +%Y%m%d%H%M%S)

export PACKER LOG_PATH="packer_\$PACKER_BUILD_TIMESTAMP.log"

For details about Packer configuration, see **Configuring Packer**.

After the image is created, information similar to the following will display.

==> huaweicloud-ecs.builder: Setting a 15m0s timeout for the next provisioner
==> huaweicloud-ecs.builder: Provisioning with shell script: /tmp/packer-shell759174699
==> huaweicloud-ecs.builder: Setting a 15m8s timeout for the next provisioner
==> huaweicloud-ecs.builder: Uploading packer_20210530185050.log => /var/log/packer_20210530185050.log
huaweicloud-ecs.builder: packer_20210530185050.log 43.63 KiB / 43.50 KiB [====================================
==> huaweicloud-ecs.builder: Stopping server: 9c901ac9-37b5-40af-934e-7190e6fa088e
huaweicloud-ecs.builder: Waiting for server to stop: 9c901ac9-37b5-40af-934e-7190e6fa088e
==> huaweicloud-ecs.builder: Creating the image: image-by-packer-20210530185050
==> huaweicloud-ecs.builder: Waiting for image image-by-packer-20210530185050 to become available
huaweicloud-ecs.builder: Image: 64e940f4-d674-4ae1-89cc-299501581c59
==> huaweicloud-ecs.builder: Deleted temporary floating IP '494617cc-a7c9-442a-b3e8-3b90c2c3f804' (94.74.101.22)
==> huaweicloud-ecs.builder: Terminating the source server: 9c901ac9-37b5-40af-934e-7190e6fa088e
==> huaweicloud-ecs.builder: Deleting volume: bf769e29-e1fd-407b-bbec-79f353a3e671
==> huaweicloud-ecs.builder: Deleting temporary keypair: packer_60b36e0b-1f16-acc5-df04-d045aba70856
Build 'huaweicloud-ecs.builder' finished after 3 minutes 53 seconds.
==> Wait completed after 3 minutes 53 seconds
==> Builds finished. The artifacts of successful builds are:
> huaweicloud-ecs.builder: An image was created: 64e940f4-d674-4ae1-89cc-299501581c59
ISun May 30 18:54:45 CST 20211 packer.sh finish.



Clear the build files on the executor, mainly the authentication credentials in **turbo-node.pkr.hcl**.

- If the authentication credentials are temporary, directly release the executor.
- If they are built automatically, add post-processor in the configuration file to execute related operations.

----End

Common Issues

• When the packer is used to create an image, the latest Huawei Cloud ECS open-source plugin is automatically obtained from GitHub. However, this process may fail due to the network environment.

Start to packer(/usr/bin/packer) with [turbo-node.pkr.hcl]
Fri Dec 31 17-06:15 CST 2021] packer version checked: 1.7.8 [Fri Dec 31 17-06:15 CST 2021] start to init/build with [turbo-node.pkr.hcl] falled getting the "github.com/hawekicloud" plugin:
<pre>terior occurred: * could not get sha255 checksum file for github.com/huaweicloud/huaweicloud version 0.4.0. Is the file present on the release and correctly named 7 Get "http //github.com/huaweicloud/packer-plugin-huaweicloud/releases/download/v0.4.0/packer-plugin-huaweicloud_v0.4.0_SHA2565UMS": unexpected EDF</pre>
Error: no plugin installed for github.com/huaweicloud/huaweicloud >= 0.4.0
Did you run packer init for this project ?
[Fri Dec 31 17:08:16 CST 2021] packer.sh finish.

To solve the problem, apply either of the following methods:

 Create an executor in a region such as CN-Hong Kong with a better network performance and connect the executor to the original region, for example, CN North-Beijing 4, to build a custom image.

export REGION_NAME=cn-north-4

 Download the corresponding plugin and initialize it to the local add-on path.

To obtain the plugin releases, see **GitHub**.

If the executor is x86, and the downloaded plug-in is packer-pluginhuaweicloud_v1.0.4_x5.0_linux_amd64.zip, initialize the plug-in:

PLUGIN_PATH="\$HOME/.packer.d/plugins/github.com/huaweicloud/huaweicloud" mkdir -p \$PLUGIN_PATH unzip packer-plugin-huaweicloud_v1.0.4_x5.0_linux_amd64.zip -d /tmp/ cp /tmp/packer-plugin-huaweicloud_v1.0.4_x5.0_linux_amd64 \$PLUGIN_PATH/ sha256sum /tmp/packer-plugin-huaweicloud_v1.0.4_x5.0_linux_amd64 | awk '{print \$1}' >

\$PLUGIN_PATH/packer-plugin-huaweicloud_v1.0.4_x5.0_linux_amd64_SHA256SUM
ll \$PLUGIN_PATH/*

Then, run the **make image** command to create an image.

• If the error message "PublicIp type is invalid" is displayed, you need to check the EIP type in different environments, run **export PKR_VAR_eip_type='***xxx*', and create an image. For details about EIP types, see **Assigning an EIP**.

An example is as follows: export PKR_VAR_eip_type='5_bgp'

• If the error code **Ecs.0019** and the error message "Flavor xxxx is abandoned" are displayed, the flavor may be unavailable in the current AZ. Try again or change the flavor in the **turbo-node.pkr.hcl** file.

[Wed Jan 24 17:22:39 CST 2024] start to init/build with [turbo-node.pkr.hcl]
huaweicloud-ecs.builder: output will be in this color.
==> huaweicloud-ecs.builder: Loading availability zones
huaweicloud-ecs.builder: Availability zones: cn-north-7-ies-wlcbs cn-north-7a cn-north-7b cn-north-7c
huaweicloud-ecs.builder: Select cn-north-7b as the availability zone
==> huaweicloud-ecs.builder: Loading flavor: s6.xlarge.2
==> huaweicloud-ecs.builder: Creating temporary keypair: packer_65b0d6e0-6237-7a02-3923-04a9320698c2
==> huaweicloud-ecs.builder: Created temporary keypair: packer 65b0d6e0-6237-7a02-3923-04a9320698c2
huaweicloud-ecs.builder: the [677f546b-d669-44dc-81fa-cb5aId57b0fb] security groups will be used
==> huaweicloud-ecs.builder: Creating EIP
huaweicloud-ecs.builder: Created EIP: '2f91bb05-8525-4a00-be08-58e03065df9d' (100.95.149.26)
==> huaweicloud-ecs.builder: Launching server in AZ cn-north-7b
==> huaweicloud-ecs.builder: Deleted temporary public IP '2f91bb05-8525-4a00-be08-58e03065df9d' (100.95.149.26)
==> huaweicloud-ecs.builder: Deleting temporary keypair: packer_65b0d6e0-6237-7a02-3923-04a9320698c2
Build 'huawei <u>cloud-ecs.builder' errored after 9 seconds 268 milliseconds</u> {"status_code":400,"request_id":"fcb8b9elc43c120b274e8c8b
"error_code": <mark>'Ecs.0019"</mark> "error_message <mark>'</mark> :"Flavor s6.xlarge.2 is abandoned","encoded_authorization_message":""}
==> Wait completed after 9 seconds 268 milliseconds
==> Some builds didn't complete successfully and had errors:
> huaweicloud-ecs.builder: {"status_code":400,"request_id":"fcb8b9e1c43c120b274e8c8b55e2c487","error_code":"Ecs.0019","error_mess
or s6.xlarge.2 is abandoned","encoded_authorization_message":""}

• If the error message "no such host" is displayed, the current IAM domain name may fail to be resolved.

Configure the following environment variables on the host. *{IAM endpoint}* specifies the IAM domain name of the current region. export PKR_VAR_auth_url='*{IAM endpoint}*}

9.4 Executing the Pre- or Post-installation Commands During Node Creation

Background

When creating a node, use the pre- or -installation commands to install tools or perform security hardening on the node. This section provides guidance for you to correctly use the pre- or post-installation scripts. To use advanced installation scripts, store the scripts in OBS buckets to prevent problems such as excessive characters in the scripts. For details, see Using OBS Buckets to Implement Custom Script Injection During Node Creation.

Precautions

• Do not use pre- or post-installation scripts that take a long time to execute.

The pre-installation script has a 15-minute time limit, while the postinstallation script has a 30-minute time limit. If the node is not available within the designated time, the node reclaim process will be initiated. Therefore, do not use pre- or post-installation scripts that take a long time to execute.

• Do not directly use **reboot** in the script.

CCE executes the post-installation command after installing mandatory components on a node. The node will be available only after the post-installation command is executed. If you run **reboot** directly, the node may be restarted before its status is reported. As a result, it cannot reach the running state within 30 minutes, and a rollback due to timeout will be triggered. Therefore, do not use **reboot**.

If you need to restart a node, perform the following operations:

Run shutdown -r <time > in the script to delay the restart. For example, you can run shutdown -r 1 to delay the restart for 1 minute.

- After the node is available, manually restart it.

Procedure

- **Step 1** Log in to the CCE console. In the navigation pane, choose **Clusters**. Click the target cluster name to access the cluster console.
- **Step 2** Choose **Nodes** in the navigation pane, click the **Nodes** tab, click **Create Node** in the right corner, and configure the parameters.
- Step 3 In the Advanced Settings area, enter pre- or post-installation commands.

Post-installation Command iptables -A INPUT -p tcp --dport 80 -m limit --limit 25/minute -limit-burst 100 -j ACCEPT

For example, you can create iptables rules by running a post-installation command to allow a maximum of 25 TCP data packets to be addressed to port 80 per minute and allow a maximum of 100 data packets to be addressed to the port when the limit is exceeded to prevent DDoS attacks.

iptables -A INPUT -p tcp --dport 80 -m limit --limit 25/minute --limit-burst 100 -j ACCEPT

NOTE

The command example here is for reference only.

- **Step 4** After the configuration, enter the number of nodes to be purchased and click **Next: Confirm**.
- Step 5 Click Submit.

----End

9.5 Using OBS Buckets to Implement Custom Script Injection During Node Creation

Background

If you need to install some tools or perform custom security hardening on a node in advance, you need to inject some scripts when creating the node. CCE allows you to inject pre-installation and post-installation scripts when creating a node. However, this function has the following restrictions:

- The script characters are limited.
- The injected script content needs be frequently modified to meet various requirements and scenarios. However, CCE node pools have fixed scripts and do not support frequent modifications.

Solution

This section provides a simplified, scalable, easy-to-maintain best practice that combines CCE and OBS. It enables custom operations on CCE nodes.

The pre-installation and post-installation scripts are stored in an OBS bucket. When creating a node pool, the scripts directly retrieve the address of the corresponding OBS script and execute it. This approach eliminates the need to modify the configuration of the CCE node pool. If there are any new requirements, you can simply update the scripts in the OBS bucket.



Suggestions on Maintaining OBS Buckets

- If there is no OBS bucket dedicated for O&M, create an OBS bucket dedicated for O&M.
- Create a multi-level directory **tools/cce** in the bucket to represent CCE-specific tools for easy maintenance. You can also store other tool scripts in this directory later.

Precautions

- If the custom operation implemented by the script fails, the normal service running is affected. You are advised to add a check program at the end of the script. If the check fails, stop the kubelet process in the post-installation script to prevent services from being scheduled to the node. systemctl stop kubelet-monit systemctl stop kubelet
- Do not include sensitive information in the scripts to prevent information leakage.

Procedure

Step 1 Create an OBS bucket.

Step 2 Upload the pre-installation and post-installation scripts. The **pre_install.sh** and **post_install.sh** scripts are used as examples.

Figure 9-7 Uploading scripts

< ops-cce 🗇				
Overview	Objects / tools / cce 🗖			
Objects				
Metrics NEW	Objects Deleted Object	cts Fragments		
Permissions 💌	Objects are basic units of data s	torage. In OBS, files and folders are ti	reated as objects. Any file type can be	uploaded and managed in a bucl
Basic Configurations	You can use OBS Browser+ to move an object to any other folder in this bucket. Upload Object Create Folder Delete More +			
Domain Name Mgmt	Name J≡	Storage Class JΞ	Size JΞ	Encrypted JΞ
Cross-Region			-	
Replication	← Back			
Back to Source	post_install.sh	Standard	11 bytes	No
Data Processing 🔹	pre_install.sh	Standard	11 bytes	No
Inventories	•			

- **Step 3** Configure a read-only security policy for the scripts to ensure that the scripts can be downloaded without entering a password on the CCE nodes but cannot be downloaded from the Internet.
 - 1. Configure a policy for the **tools/cce** directory and select a read-only policy template.

Figure 9-8 Configuring an object policy

Objects Deleted (Objects Fragments					
bjects are basic units of o	data storage. In OBS, files and folders are treated	l as objects. Any file type can be i	uploaded and managed in a bucket. Learn mo	re		
u can use OBS Browser+ Upload Object	to move an object to any other folder in this bu Create Folder Delete More 💌	icket.				Enter an object name prefix Q
Name JΞ	Storage Class JΞ	Size ↓Ξ	Encrypted ↓≡	Restoration Status ↓Ξ	Last Modified ↓	Operation
Seck						
🗌 🔁 cce						Share Copy Path More 💌
						Delete Configure Object Policy

2. Modify the configuration information about the anonymous user, specified objects, and conditions.

Figure 9-9 Anonymous user

Configure Object Policy					
1) Select Template	2 Configure Policy	(3) Confirm Policy			
Policy View	Visual editor	JSON			
Policy Name	Read-only-8481				
Policy Content	🛛 Allow 📃 🤇	Include anonymous users	Include the specified objects	Include 5 actions	Conditions (optional)
	Principal 🕐	Current account Other	r account Anonymous user		
	Anonymous User				
	Anonymous oser	•			
		O Anonymous user selected. This may	pose potential security risks because any user can perform operation	itions allowed by this bucket policy without v	verifying their identity.
	User Policy	Include medified users	*		
		include specified (det)			

Figure 9-10 Specified objects

< Configure Object Po	olicy			
1) Select Template	— 2 Configure Poli	cy ③ Confirm Policy		
Policy View	Visual editor	ISON		
Policy Name	Read-only-8481			
Policy Content	Illow	Include anonymous users	Include 5 actions	Conditions (optional)
	Resource	Objects in bucket You can configure actions that are applicable to objects.		
		All objects Specified objects		
		Resource path:		
		○ ops-cce tools/cce/*.sh View object name prefixes		
		Add Object Name Prefix		
	Resource Policy	Include specified resources		

Figure 9-11 Conditions

(Configure Object Policy						
 Select Template 	🕗 Configure Pol	licy (3) Confirm Policy				
Policy View	Visual editor	JSON				
Policy Name	Read-only-8481					
Policy Content	Allow	Include anonymous users	Include the specified objects	Include 5 actions	Conditions: 2	
	Conditions required	for this policy to take effect. A condition is	expressed as a key-value pair. View configura-	tion examples		
	Conditional Operator		Key	Value	Operation	
	StringEquals	*	UserAgent +	ccePrePosticistali	Delete	
	IpAddress	×	Sourcelp *	100.0.0/8,10.0.0.0/8,172.16.0.0/12,192.1	68.0.0/16 Delete	
	C.Add Condition					

Two conditions are set: **UserAgent** and **Sourcelp**.

- UserAgent functions in the similar way as a key. The specified User-Agent request header and the corresponding key value must be carried during access.
- **Sourcelp** is used to prevent access from external networks. Set this parameter to

100.0.0/8,10.0.0/8,172.16.0.0/12,192.168.0.0/16,214.0.0.0/8, among which **100.0.0/8** and **214.0.0.0/8** are private IP ranges, among whom **10.0.0.0/8, 172.16.0.0/12**, and **192.168.0.0/16** are commonly used.

For details about how to configure an OBS policy, see **Configuring an Object Policy** and **Bucket Policy Parameters**.

Step 4 Configure the pre-installation script and post-installation script when creating a node pool on CCE.

Enter the following scripts in the **Advanced ECS Settings** on the **Create Node Pool** page:

Pre-installation Command	curl -H "User-Agent: ccePrePostInstall" https://ops-cce.obs.cn-north-4.myhuaweicloud.com/tools/cce/pre_install.sh -o /tmp/pre_install.sh & bash -x /tmp/pre_install.sh > /tmp/pr	
		196/1,000
Post-installation Command	curl -H "User-Agent: ccePrePostInstall" https://ops-cce.obs.cn-north-4.myhuaweicloud.com/tools/cce/post_install.sh -o /Imp/post_install.sh && bash -x //mp/post_install.sh > //mp/post_	
		200/1,000

In the scripts below, the **curl** command is run to download **pre_install.sh** and **post_install.sh** from OBS to the **/tmp** directory, and then **pre_install.sh** and **post_install.sh** are executed.

Pre-installation script:

curl -H "User-Agent: ccePrePostInstall" https://ops-cce.obs.ap-southeast-1.myhuaweicloud.com/tools/cce/ pre_install.sh -o /tmp/pre_install.sh && bash -x /tmp/pre_install.sh > /tmp/pre_install.log 2>&1

Post-installation script:

curl -H "User-Agent: ccePrePostInstall" https://ops-cce.obs.ap-southeast-1.myhuaweicloud.com/tools/cce/ post_install.sh -o /tmp/post_install.sh && bash -x /tmp/post_install.sh > /tmp/post_install.log 2>&1

NOTE

- The actual value of **User-Agent** must be configured based on the OBS bucket policy.
- The bucket address in the link must be configured based on site requirements.

----End

9.6 Connecting to Multiple Clusters Using kubectl

Background

The kubectl command line tool relies on the kubeconfig configuration file to locate the necessary authentication information to select a cluster and communicate with its API server. By default, kubectl uses the **\$HOME/.kube/ config** file as the credential for accessing the cluster.

When working with CCE clusters on a daily basis, it is common to manage multiple clusters simultaneously. However, this can make using the kubectl command line tool to connect to clusters cumbersome, as it requires frequent switching of the kubeconfig file during routine O&M. This section introduces how to connect to multiple clusters using the same kubectl client.

NOTE

The file used to configure cluster access is called the kubeconfig file, but it does not mean that the file name is **kubeconfig**.

Solution

When performing O&M on Kubernetes clusters, it is often necessary to switch between multiple clusters. The following shows some typical solutions for cluster switchover:

- **Solution 1**: Specify --**kubeconfig** of kubectl to select the kubeconfig file used by each cluster and use aliases to simplify commands.
- Solution 2: Combine clusters, users, and credentials in multiple kubeconfig files into one configuration file and run kubectl config use-context to switch clusters.

Compared with solution 1, this solution requires manual configuration of the kubeconfig file, which is relatively complex.



Figure 9-12 Using kubectl to connect to multiple clusters

Prerequisites

- You have a Linux VM with the kubectl command line tool installed. The kubectl version must match the cluster version. For details, see Install Tools.
- The VM where kubectl is installed must be able to access the network of each cluster.

kubeconfig File Structure

kubeconfig is the configuration file of kubectl. You can download it on the cluster details page.

Status	 Running 	Access Cluster example Through kubect
Cluster Scale	50 nodes	
Created	Mar 15, 2022 00:16:29 GM	You need to download kubect1 and its configuration file, copy the file to your client, and configure kubect1. After the configuration is complete, you can use kubect1 to access your Kubernetes clusters.
Enterprise Project	default	C Download kubectl
		Go to the Kubernetes version release page to download kubectl corresponding to the cluster version or a later version. If you have installed kubectl, skip this step.
Networking Conf	iguration	1. Run the following command to check whether kubectl is installed:
Network Model	VPC network	kubectl version
VPC	vpc-demo 🗠	Davelard the luther offer
Subnet	subnet-test	Click have to download the subjecting life.
Container CIDR Bloc	k 10.0.0/16	Install and configure kubectl.
Service CIDR Block	10.247.0.0/16	The following operations use the Linux environment as an example. For details, see Install Tools.
Forwarding	iptables	• 1. Copy kubect1 and its configuration file to the /home directory on your client. If kubectl has been installed, you only need to copy
		the kubeconfig file.
		• 2. Log in to your client, and configure kubect1. If you have installed kubectl, skip this step.
Connection Infor	mation	1 of thoma
Private IP	https://192.168.5.161:54	2. chmod +x kubecti
Public IP	Bind	3. mv -f kubectl /usr/local/bin
Custom SAN	🖉	3. Log in to your client and configure the kubeconfig file.
kubectl	Learn more	1. cd /home
Authentication Mode	X.509 certificate Downle	2. mkdir -p \$HOME/.kube
		3. mv -f kubeconfig.json \$HOME/.kube/config
		4. Switch the kubect1 access mode based on application scenarios.
Other		Run this command to enable intra-VPC access:
Billing Mode	Pay-per-use	kubecti config use-context internal 🗇
Resource Tag	- 2	

The content of the kubeconfig file is as follows:

```
[
    "kind": "Config",
    "apiVersion": "v1",
    "preferences": {},
    "clusters": [{
        "name": "internalCluster",
        "cluster": {
            "server": "https://192.168.0.85:5443",
            "server": "https://192.168.0.85:5443",
```
```
"certificate-authority-data": "LS0tLS1CRUULIE..."
  }
}, {
   "name": "externalCluster",
   "cluster": {
     "server": "https://xxx.xxx.xxx.sxx:5443",
     "insecure-skip-tls-verify": true
  }
}],
"users": [{
   "name": "user",
   "user": {
      "client-certificate-data": "LSOtLS1CRUdJTiBDRVJ...",
     "client-key-data": "LS0tLS1CRUdJTiBS..."
  }
}],
"contexts": [{
   "name": "internal",
   "context": {
      "cluster": "internalCluster",
      "user": "user"
}, {
   "name": "external",
   "context": {
      "cluster": "externalCluster",
      "user": "user"
  }
}],
 'current-context": "external"
```

It mainly consists of three sections.

- **clusters**: describes the cluster information, mainly the access address of the cluster.
- users: describes information about the users who access the cluster. It
 includes the client-certificate-data and client-key-data certificate files.
- contexts: describes the configuration contexts. You switch between contexts to access different clusters. A context is associated with user and cluster, that is, it defines which user accesses which cluster.

The preceding kubeconfig defines the private network address and public network address of the cluster as two clusters with two different contexts. You can switch the context to use different addresses to access the cluster.

Solution 1: Specify Different kubeconfig Files in Commands

- **Step 1** Log in to the VM where kubectl is installed.
- **Step 2** Download the kubeconfig files of the two clusters to the **/home** directory on the kubectl client. The following names are taken as examples.

Cluster Name	kubeconfig File Name
Cluster A	kubeconfig-a.json
Cluster B	kubeconfig-b.json

Step 3 Make kubectl access cluster A by default and move the **kubeconfig-a.json** file to **\$HOME/.kube/config**.

cd /home mkdir -p \$HOME/.kube mv -f *kubeconfig-a.json* \$HOME/.kube/config

Step 4 Move the **kubeconfig-b.json** file of cluster B to **\$HOME/.kube/config-test**. mv -f *kubeconfig-b.json* \$HOME/.kube/config-test

The name of the **config-test** file can be customized.

Step 5 Add --kubeconfig to specify the credential used by the kubectl commands when accessing cluster B. (There is no need to add --kubeconfig when running kubectl commands to access cluster A, because kubectl can access cluster A by default.) For example, run the following command to check the nodes in cluster B: kubectl --kubeconfig=\$HOME/.kube/config-test get node

If you frequently use a long command, the preceding method can be inconvenient. To simplify the command, you can use aliases. For example: alias ka='kubectl --kubeconfig=\$HOME/.kube/config' alias kb='kubectl --kubeconfig=\$HOME/.kube/config-test'

In the preceding information, **ka** and **kb** can be custom aliases. When running the kubectl command, you can directly enter **ka** or **kb** to replace **kubectl**. The -- **kubeconfig** parameter is automatically added. For example, the command for checking nodes in cluster B can be simplified as follows:

kb get node

----End

Solution 2: Combine the kubeconfig Files of the Two Clusters Together

The following steps walk you through the procedure of modifying the kubeconfig files and accessing multiple clusters.

This example configures only the public network access to the clusters. If you want to access multiple clusters over private networks, retain the **clusters** field and ensure that the clusters can be accessed over private networks. Its configuration is similar to that described in this example.

- **Step 1** Download the kubeconfig files of the two clusters and delete the lines related to private network access, as shown in the following figure.
 - Cluster A:

ł

```
"kind": "Config"
"apiVersion": "v1",
"preferences": {},
"clusters": [ {
   "name": "externalCluster",
"cluster": {
      "server": "https://119.xxx.xxx.xxx:5443",
      "insecure-skip-tls-verify": true
   }
}],
"users": [{
   "name": "user",
   "user": {
      "client-certificate-data": "LS0tLS1CRUdJTxM...",
      "client-key-data": "LS0tLS1CRUdJTiB....'
   }
}],
"contexts": [{
   "name": "external",
```

.

```
"context": {
          "cluster": "externalCluster",
          "user": "user"
      }
   }],
   "current-context": "external"
}
Cluster B:
{
   "kind": "Config"
   "apiVersion": "v1",
   "preferences": {},
   "clusters": [ {
"name": "externalCluster",
      "cluster": {
          "server": "https://124.xxx.xxx.xxx:5443",
          "insecure-skip-tls-verify": true
      }
   }],
   "users": [{
      "name": "user",
      "user": {
          "client-certificate-data": "LS0tLS1CRUdJTxM...",
"client-key-data": "LS0rTUideUdJTiB...."
      }
   }],
   "contexts": [{
"name": "external",
      "context": {
          "cluster": "externalCluster",
          "user": "user"
      }
   }],
   "current-context": "external"
}
```

The preceding files have the same structure except that the **client-certificatedata** and **client-key-data** fields of **user** and the **clusters.cluster.server** field are different.

Step 2 Modify the name field as follows:

```
Cluster A:
{
   "kind": "Config",
   "apiVersion": "v1",
   "preferences": {},
   "clusters": [ {
"name": "Cluster-A",
      "cluster": {
         "server": "https://119.xxx.xxx.xxx:5443",
         "insecure-skip-tls-verify": true
      }
   }],
   "users": [{
      "name": "Cluster-A-user",
      "user": {
         "client-certificate-data": "LS0tLS1CRUdJTxM...",
         "client-key-data": "LS0tLS1CRUdJTiB...."
      }
   }],
   "contexts": [{
"name": "Cluster-A-Context",
      "context": {
         "cluster": "Cluster-A",
"user": "Cluster-A-user"
      }
  }],
```

}

"current-context": "Cluster-A-Context"

```
Cluster B:
{
   "kind": "Config",
   "apiVersion": "v1",
   "preferences": {},
   "clusters": [ {
      "name": "Cluster-B",
"cluster": {
         "server": "https://124.xxx.xxx.xxx:5443",
         "insecure-skip-tls-verify": true
     }
  }],
   "users": [{
      "name": "Cluster-B-user",
      "user": {
         "client-certificate-data": "LS0tLS1CRUdJTxM...",
         "client-key-data": "LS0rTUideUdJTiB...."
     }
  }],
   "contexts": [{
"name": "Cluster-B-Context",
      "context": {
         "cluster": "Cluster-B",
         "user": "Cluster-B-user"
     }
  }],
   "current-context": "Cluster-B-Context"
}
```

Step 3 Combine these two files.

The file structure remains unchanged. Combine the contents of **clusters**, **users**, and **contexts** as follows:

```
"kind": "Config",
"apiVersion": "v1",
"preferences": {},
"clusters": [ {
   "name": "Cluster-A",
   "cluster": {
      "server": "https://119.xxx.xxx.xxx:5443",
      "insecure-skip-tls-verify": true
  }
},
{
   "name": "Cluster-B",
   "cluster": {
      "server": "https://124.xxx.xxx.xxx:5443",
      "insecure-skip-tls-verify": true
  }
}],
"users": [{
   "name": "Cluster-A-user",
   "user": {
      "client-certificate-data": "LS0tLS1CRUdJTxM...",
      "client-key-data": "LS0tLS1CRUdJTiB...."
   }
},
{
   "name": "Cluster-B-user",
   "user": {
      "client-certificate-data": "LS0tLS1CRUdJTxM...",
      "client-key-data": "LS0rTUideUdJTiB...."
   }
}],
"contexts": [{
```

```
"name": "Cluster-A-Context",
    "context": {
        "cluster": "Cluster-A",
        "user": "Cluster-A-user"
    }
},
{
    "name": "Cluster-B-Context",
    "context": {
        "cluster": "Cluster-B",
        "user": "Cluster-B-user"
    }
}],
"current-context": "Cluster-A-Context"
```

```
}
```

Step 4 Run the following command to copy the combined file to the kubectl configuration path:

mkdir -p \$HOME/.kube

mv -f kubeconfig.json \$HOME/.kube/config

```
Step 5 Run the kubectl command to check whether the two clusters can be accessed.
```

```
# kubectl config use-context Cluster-A-Context
Switched to context "Cluster-A-Context".
# kubectl cluster-info
Kubernetes control plane is running at https://119.xxx.xxx.5443
CoreDNS is running at https://119.xxx.xxx.5443/api/v1/namespaces/kube-system/services/coredns:dns/
proxy
To further debug and diagnose cluster problems, use 'kubectl cluster-info dump'.
```

```
# kubectl config use-context Cluster-B-Context
Switched to context "Cluster-B-Context".
# kubectl cluster-info
Kubernetes control plane is running at https://124.xxx.xxx.5443
CoreDNS is running at https://124.xxx.xxx:5443/api/v1/namespaces/kube-system/services/coredns:dns/
proxy
```

To further debug and diagnose cluster problems, use 'kubectl cluster-info dump'.

If you frequently use a long command, the preceding method can be inconvenient. To simplify the command, you can use aliases. For example:

alias ka='kubectl config use-context Cluster-A-Context;kubectl' alias kb='kubectl config use-context Cluster-B-Context;kubectl'

In the preceding information, **ka** and **kb** can be custom aliases. When running the kubectl command, you can directly enter **ka** or **kb** to replace **kubectl**. You need to switch the context and then run the kubectl command. For example:

```
# ka cluster-info
Switched to context "Cluster-A-Context".
Kubernetes control plane is running at https://119.xxx.xxx:5443
CoreDNS is running at https://119.xxx.xxx:5443/api/v1/namespaces/kube-system/services/coredns:dns/
proxy
```

To further debug and diagnose cluster problems, use 'kubectl cluster-info dump'.

----End

9.7 Selecting a Data Disk for the Node

When a node is created, a data disk is attached by default for a container runtime and kubelet. For details, see **Data Disk Space Allocation**. The data disk used by

the container runtime and kubelet cannot be detached, and the default capacity is 100 GiB. To cut costs, you can adjust the disk capacity to the minimum of 20 GiB or reduce the disk capacity attached to a node to the minimum of 10 GiB.

NOTICE

Adjusting the size of the data disk used by the container runtime and kubelet may incur risks. You are advised to evaluate the capacity adjustment and then perform the operations described in this section.

- If the disk capacity is too small, the image pull may fail. If different images need to be frequently pulled on the node, you are not advised to reduce the data disk capacity.
- Before a cluster upgrade, the system checks whether the data disk usage exceeds 95%. If the usage is high, the cluster upgrade may be affected.
- If Device Mapper is used, the disk capacity may be insufficient. You are advised to use the OverlayFS or select a large-capacity data disk.
- For dumping logs, application logs must be stored in a separate disk to prevent insufficient storage capacity of the dockersys volume from affecting service running.
- After reducing the data disk capacity, you are advised to install the npd add-on in the cluster to detect disk usage. If the disk usage of a node is high, resolve this problem by referring to What If the Data Disk Capacity Is Insufficient?

Notes and Constraints

- Only clusters of v1.19 or later allow reducing the capacity of the data disk used by container runtimes and kubelet.
- Only the EVS disk capacity can be adjusted. (Local disks are available only when the node specification is **disk-intensive** or **Ultra-high I/O**.)

Selecting a Data Disk

When selecting a data disk, consider the following factors:

- During image pull, the system downloads the image package (the .tar package) from the image repository, and decompresses the package. Then it deletes the package but retain the image file. During the decompression of the .tar package, the package and the decompressed image file coexist. Reserve the capacity for the decompressed files.
- Mandatory add-ons (such as everest and coredns) may be deployed on nodes during cluster creation. When calculating the data disk size, reserve about 2 GiB storage capacity for them.
- Logs are generated during application running. To ensure stable application running, reserve about 1 GiB storage capacity for each pod.

For details about the calculation formulas, see **OverlayFS** and **Device Mapper**.

OverlayFS

By default, the container engine and container image storage capacity of a node using the OverlayFS storage driver occupies 90% of the data disk capacity (you

are advised to retain this value). All the 90% storage capacity is used for dockersys partitioning. The calculation methods are as follows:

- Capacity for storing container engines and container images requires 90% of the data disk capacity by default.
 - Capacity for dockersys volume (in the /var/lib/docker directory) requires 90% of the data disk capacity. The entire container engine and container image capacity (need 90% of the data disk capacity by default) are in the /var/lib/docker directory.
- Capacity for storing temporary kubelet and emptyDir requires 10% of the data disk capacity.

On a node using the OverlayFS, when an image is pulled, the .tar package is decompressed after being downloaded. During this process, the .tar package and the decompressed image file are stored in the dockersys volume, occupying about twice the actual image storage capacity. After the decompression is complete, the .tar package is deleted. Therefore, during image pull, after deducting the storage capacity occupied by the system add-on images, ensure that the remaining capacity of the dockersys volume is greater than twice the actual image storage capacity. To ensure that the containers can run stably, reserve certain capacity in the dockersys volume for container logs and other related files.

When selecting a data disk, consider the following formula:

Capacity of dockersys volume > Actual total image storage capacity x 2 + Total system add-on image storage capacity (about 2 GiB) + Number of containers x Available storage capacity for a single container (about 1 GiB log storage capacity for each container)

NOTE

If container logs are output in the **json.log** format, they will occupy some capacity in the dockersys volume. If container logs are stored on persistent storage, they will not occupy capacity in the dockersys volume. Estimate the capacity of every container as required.

Example:

Assume that the node uses the OverlayFS and the data disk attached to this node is 20 GiB. According to **the preceding methods**, the capacity for storing container engines and images occupies 90% of the data disk capacity, and the capacity for the dockersys volume is 18 GiB (20 GiB x 90%). Additionally, mandatory add-ons may occupy about 2 GiB storage capacity during cluster creation. If you deploy a .tar package of 10 GiB, the package decompression takes 20 GiB of the dockersys volume's storage capacity. This, coupled with the storage capacity occupied by mandatory add-ons, exceeds the remaining capacity of the dockersys volume. As a result, the image pull may fail.

Device Mapper

By default, the capacity for storing container engines and container images of a node using the Device Mapper storage driver occupies 90% of the data disk capacity (you are advised to retain this value). The occupied capacity includes the dockersys volume and thinpool volume. The calculation methods are as follows:

• Capacity for storing container engines and container images requires 90% of the data disk capacity by default.

- Capacity for the dockersys volume (in the /var/lib/docker directory) requires 20% of the capacity for storing container engines and container images.
- Capacity for the thinpool volume requires 80% of the container engine and container image storage capacity.
- Capacity for storing temporary kubelet and emptyDir requires 10% of the data disk capacity.

On a node using the Device Mapper storage driver, when an image is pulled, the .tar package is temporarily stored in the dockersys volume. After the .tar package is decompressed, the image file is stored in the thinpool volume, and the package in the dockersys volume will be deleted. Therefore, during image pull, ensure that the dockersys partition space and thinpool space are sufficient, and note that the former is smaller than the latter. To ensure that the containers can run stably, reserve certain capacity in the dockersys volume for container logs and other related files.

When selecting a data disk, consider the following formulas:

- Capacity for dockersys volume > Temporary storage capacity of the .tar package (approximately equal to the actual total image storage capacity) + Number of containers x Storage capacity of a single container (about 1 GiB log storage capacity must be reserved for each container)
- Capacity for thinpool volume > Actual total image storage capacity + Total add-on image storage capacity (about 2 GiB)

NOTE

If container logs are output in the **json.log** format, they will occupy some capacity in the dockersys volume. If container logs are stored on persistent storage, they will not occupy capacity in the dockersys volume. Estimate the capacity of every container as required.

Example:

Assume that the node uses the Device Mapper and the data disk attached to this node is 20 GiB. According to **the preceding methods**, the container engine and image storage capacity occupies 90% of the data disk capacity, and the disk usage of the dockersys volume is 3.6 GiB. Additionally, the storage capacity of the mandatory add-ons may occupy about 2 GiB of the dockersys volume during cluster creation. The remaining storage capacity is about 1.6 GiB. If you deploy a .tar image package larger than 1.6 GiB, the storage capacity of the dockersys volume is insufficient for the package to be decompressed. As a result, the image pull may fail.

What If the Data Disk Capacity Is Insufficient?

Solution 1: Clearing images

Perform the following operations to clear unused images:

- Nodes that use containerd
 - a. Obtain local images on the node. crictl images -v
 - b. Delete the unnecessary images by image ID. crictl rmi {Image ID}

- Nodes that use Docker
 - a. Obtain local images on the node. docker images
 - b. Delete the unnecessary images by image ID. docker rmi {*}Image ID*}

NOTE

Do not delete system images such as the **cce-pause** image. Otherwise, the pod creation may fail.

Solution 2: Expanding the disk capacity

Expand the data disk capacity as required. For details, see **Expanding the Storage Space**.

9.8 Implementing Cost Visualization for a CCE Cluster

Background

The billing information of CCE is displayed based on the entire service by default. The costs of different clusters are not shown.

Solution

Add the **CCE-Cluster-ID** tag to the resources used in the clusters, obtain the costs by searching for this tag in cost center, and then analyze the costs by cluster to improve efficiency and cut costs.

Notes and Constraints

Generally, tags appear on the **Cost Tags** page 24 hours after resource expenditures are generated.

You can filter or group cost data by tag only after the tags are activated. If you activate the tags, they will be used to organize your resource costs generated thereafter.

Procedure

Step 1 Activate the CCE-Cluster-ID tag.

In **Cost Center**, choose **Cost Allocation** > **Cost Tags** in the navigation pane. Select **Tag** from the drop-down list for the filter criteria, search for **CCE-Cluster-ID**, and click **Activate** in the **Operation** column.

		•	•				
	Cos	t Tags					🕞 Quick Link
•	6	For more information, see Whh 1. Before using cost tags, you i generated cost data. 2. Using predefined tags is rec will appear on this page 24 ho 3. After you activate your tags, 4. If there are no tags to activa	at Are Cost Tags? or Whi need to plan how they w commended. They can be ours after their associated , it can take up to a full d ate, consider shortening t	h Services Can Use Tag II be used based on yo activated on this page resources have genera ay before the most rece he tag length or remov	is for Cost Analysis? ur services. After you activate the tags, immediately after being created. Alter ted expenditures. Learn how to add tag int tag information is displayed. ing the colon from tag keys and values	Cost Center starts applyi natively, you can use expi 3s and how to predefine !	ing them to any newly enditure tags, which tags.
	•	Activate Deactivate	Tag: CCE-Cluster-ID	Add filter			x Q
æ		Tag J⊟			Tag Source 🝞	Status 🏹	Operation
		CCE-Cluster-ID			Expenditure tags	Deactivated	Activate
	•	e cos	Cost Tags For more information, see Wh 1. Before using cost tags, you generated cost data. 2. Using predefined tags is re- will appear on this page 24 h 3. After you activate you tags 4. If there are no tags to activa Activate Deactivate Activate Deactivate CEC-Cluster-ID	Cost Tags for more information, see What Are Cost Tags? or Which I. Before using cost tags, you need to plan how they wi generated cost data. Using predefined tags is recommended. They can be will appear on this page 24 hours after their associated 3. After you activate your tags, it can take up to a full d. 4. If there are no tags to activate, consider shortening th Activate Deactivate Tag: CCE-Cluster-ID CCE-Cluster-ID	Cost Tags for more information, see What Are Cost Tags? or Which Services Can Use Tag 1. Before using cost tags, you need to plan how they will be used based on yo generated cost data. 2. Using predefined tags is recommended. They can be activated on this page will appear on this page 24 hours after their associated resources have genera 3. After you activate your tags, it can take up to a full day before the most rect 4. If there are no tags to activate, consider shortening the tag length or remov Activate Deactivate Tag: CCE-Cluster-ID Add filter Tag JE CCE:Cluster-ID	Cost Tags For more information, see What Are Cost Tags? or Which Services Can Use Tags for Cost Analysis? I. Before using cost tags, you need to plan how they will be used based on your services. After you activate the tags, generated cost data. Using predefined tags is recommended. They can be activated on this page immediately after being created. After will appear on this page 24 hours after their associated resources have generated expenditures. Learn how to additate your tags. It can take up to a full day before the most recent tag information is displayed. A. If there are no tags to activate, consider shortening the tag length or removing the colon from tag keys and values Activate Deactivate Tag: CCE-Cluster-ID Add filter Tag JE Tag Source T Expenditure tags	Cost Tags I be note information, see What Are Cost Tags? or Which Services Can Use Tags for Cost Analysis? 1. Before using cost tags, you need to plan how they will be used based on your services. After you activate the tags, Cost Center starts applying enerated cost data. 2. Using predefined tags is recommended. They can be activated on this page immediately after being created. Alternatively, you can use exp. will appear on this page 24 hours after their associated resources have generated expenditures. Learn how to add tags and how to predefine 3. After you activate your tags, it can take up to a full day before the most recent tags information is displayed. 4. If there are no tags to activate, consider shortening the tag length or removing the colon from tag keys and values. 8 Activate Deactivate Tag: CCE-Cluster-ID Add filter CCE-Cluster-ID Expenditure tags Deactivated

Figure 9-13 Activating the cost tag

The following information will be displayed.

Figure 9-14 Activation succeeded

Activate	Deactivate	Tag: CCE-Cluster-ID	Add filter			x Q
Tag ↓≣				Tag Source 🖓	Status 🏹	Operation
CCE-Cl	uster-ID			Expenditure tags	Activated	Deactivate

Step 2 Add the tag to resources used in a cluster.

Resources used in a cluster include master nodes, worker nodes, storage resources (such as EVS disks, SFS file systems, and OBS buckets), and network resources (such as load balancers and EIPs). By default, the **CCE-Cluster-ID** tag is added to the worker nodes.

• Add the tag to a cluster.

On the CCE console, click the name of the target cluster to access the cluster console. On the **Cluster Information** page, add the resource tag in the **Other** area.

I ⁽²⁾ Overview	VPC	vpc-coe 🖸		Events are retained for 30 days and will be then extremetically cleared. Th	a maximum time ranne for an event muory is 15 days
Kubernetes Resources	Subnet	subnet-coe		 Litera di constructiva de subjetaria na persona accontancen y accura. 	Check Date _ Cost Date
R Workloads	Default Pod Subnet	subnet-coe-container			Start Date - End Date
Gervices & Ingresses		Add Pod		×	Event Remarks
Storage		goto Net Edit Tag		<u>^</u>	-
Configurations And Se	IPv4 Service CIDR Block	10.247.0	If users need to use the same to	ag to identify multiple cloud resources, that is, all services can use the same tag.	TraceID-570319817155.11ee.00db.0255ac1002
Q Policies	Forwarding	iptables	it is recommended to create a p	redefined tag in TMS. View predefined tags	
Templates	Default Node Security Group	cce-test-	CCE-Cluster-ID	= 07da5e02-9/9/-11ed-bbae-0255ac Add C	TraceID:5793189a-7155-11ee-99db-0255ac100
Custom Resources			Max. resource tags: 20		TraceID:57f1b5c4-7155-11ee-99db-0255ac1002
Cluster	Connection Information			OK Cancel	TraceID:57ae760d-7155-11ee-99db-0255ac100
Å, Nodes	Private IP ht	ttps://192.166.0.191.9449 Er		Uct 23, 2023 11:37:09 Gh11*0	TraceID:579318a7-7155-11ee-99db-0255ac100
C Settings	EIP -	Bind		Oct 19, 2023 18:46:45 GMT+0 Creation result: 1 succeeded, 0 failed	-
88 Add-ons	custom SAN	onfigure		Oct 19, 2023 16:46:45 GMT+0 Node(s) created successfully	Create node
08.M	Certificate Authentication X	509 certificate Download		Oct 19, 2023 16 46 45 GMT+0 Node kubernetes installed success	Install node kubernetes
Monitoring Ce New				Oct 19, 2023 16:46:05 GMT+0 Begin to install node kubernetes	Install node kubernetes
Logging New	Other			Total Records: 53 (2 3 4 5 6)	
[9] Alarm assistant New	Billing Mode Pay-pe	ar-use			
	Resource Tag - Add	1			
	Deletion Protection Enable	0			
	Description - 🖉				

• Add the tag to an EVS disk.

On the EVS console, click the name of the target disk to go to the details page. On the **Tags** tab, add a tag.

< pvc-7350146b-114f-47b6-bebc-b1e917f89819

Sur	nmary	Servers	Backups	Snapshots	Tags	
	Add Tag	You can ad	dd 9 more tags. A	tag is a pair of key a	nd value. For hierarchical	management, use both keys and values. For common mar
	Key					Value
	CCE-Clust	er-ID				07da5e02-9f9f-11ed-bbae-0255ac1002c5

• Add the tag to an OBS bucket.

On the OBS console, click the name of the target bucket to go to the details page. Choose **Basic Configurations** > **Tagging** and add a tag.

< pvc-00f35905-063a-4db2 □								
Overview								
Files		Tagging						
Metrics NEW		You can add tags to parallel file systems to facilitate classification and management. Learn more						
Permissions	•	Add Tag You can add 9 more tags. A tag is a pair of key and value. For hierard	chical management, use both keys and values. For					
Basic Configurations		Key	Value					
Lifecycle Rules		CCE-Cluster-ID	07da5e02-9f9f-11ed-bbae-0255ac1002c5					
Event Notification								
Tagging								

• Add the tag to an SFS Turbo file system.

On the SFS console, click the name of the target SFS Turbo file system to go to the details page. On the page displayed, click the **Tags** tab and add a tag.

< cost-sf	-turbo	
Basic Info	Tags	
() We would	much appreciate if you could complete our questionnaire on Scalable File System. Your feedback will hel	p us provide a better user experience.
Add Tag	You can add 9 more tags.	
Key		Value
CCE-Cluste	r-ID	07da5e02-9f9f-11ed-bbae-0255ac1002c5

• Add the tag to a load balancer.

On the ELB console, click the name of the target load balancer to go to the details page. On the page displayed, click the **Tags** tab and add a tag.

Summary	Listeners	Monitoring	Access Logs	Tags	
Add Ta	ag You can a	dd 9 more tags. A ta	g is a pair of key and v	lue. For hierarchical management, use both keys and	values. For common
Key				Value	
CCE-Clu	ister-ID			07da5e02-9f9f-11ed-bbae-0255a	ac1002c5

• Add the tag to an EIP.

On the EIP console, click the name of the target EIP to go to the details page. Click the **Tags** tab and add a tag.

Sum	nmary	Bandwidth	Tags	
	Add Tag	You can add 9	more tags. A tag is a pair of key and value. For hierarchical	management, use both keys and values. For cor
	Key			Value
	CCE-Clust	er-ID		07da5e02-9f9f-11ed-bbae-0255ac1002c5

Step 3 Analyze the cluster's costs.

On the **Cost Analysis** page, choose **CCE-Cluster-ID** and the ID of the desired cluster from the drop-down list.

Figure 9-15 Analyzing costs



Click OK.

Figure 9-16 Viewing the cluster's costs

Service Type	Accrued Total	Forecasted Total **	Aug 2023	
Total cost(¥)	53.08		53.08	
Elastic Cloud Serve	30.44		30.44	
Scalable File Servic	17.50		17.50	
Elastic Volume Ser	5.14		5.14	
Virtual Private Clou	0.00		0.00	
Object Storage Ser	0.00		0.00	
Elastic Load Balanc	0.00		0.00	

----End

9.9 Creating a CCE Turbo Cluster Using a Shared VPC

Shared VPC Overview

A shared VPC allows you to share your VPC resources with other accounts through the Resource Access Manager (RAM) service. For example, tenant A can share its VPC and subnets with tenant B. After tenant B accepted the sharing, tenant B can view the shared subnets and the shared VPC to which the shared subnets belong. Tenant B can use the shared subnets and VPC to create resources, such as CCE Turbo clusters. For details, see **VPC Sharing Overview**.

Application Scenarios

An enterprise organizes accounts in an orderly and centralized manner based on its organization structure or service form. Resources are managed in a unified manner and shared with other members to avoid repeated configurations. Unified security and O&M management makes it easy to configure and audit security policies.

For example, an enterprise IT account, the resource owner, creates a VPC and subnets and shares multiple subnets with other accounts.

- Account A is an enterprise service account and uses the shared subnet 1 to create resources.
- Account B is an enterprise service account and uses the shared subnet 2 to create resources.

Notes and Constraints

- Only CCE Turbo clusters support shared VPCs.
- Clusters created using a shared VPC do not support shared load balancers and NAT gateways.
- Clusters created using a shared VPC do not support SFS, OBS, and SFS Turbo storage volumes.
- If a CCE Turbo cluster has been created using a shared VPC, the owner of the shared VPC should not turn off the VPC sharing. Otherwise, the CCE Turbo cluster will malfunction.

Procedure

After account A shares a VPC with account B, account B can select the shared VPC and shared subnets when creating a CCE Turbo cluster.

Step 1 (For account A) Use RAM to create a shared VPC and specify account B as the resource user. For details, see **Creating a Resource Share**.

After the resource sharing is created, RAM sends an invitation to account B. Account B can access and use the shared resources only after accepting the invitation.

Step 2 (For account B) Log in to the CCE console and create a CCE Turbo cluster.

294

Select the VPC shared by account A when configuring network for the cluster. For details about other configurations, see **Buying a CCE Cluster**.

Network Setting	S
Cluster Network	
VPC 🔏 🕜	vpc-turbo (192.168.0.0/16) Shared VPC V C Create VPC C
	Select a VPC to provide CIDR blocks for the master and worker nodes and pods in your cluster. Shared VPC: The VPC is shared by another tenant to the current tenant. View Details 🕜
Subnet 🔏 🕜	subnet-47ac (192 168 64 0/18) Shared Subnet V C
	Available Subnet IP Addresses: 16,167
	Select a subnet in your VPC. Nodes in the cluster will use the IP addresses in the subnet. Master nodes in a cluster are hosted and maintained by CCE. You only need to configure for worker nodes.

Figure 9-17 Selecting a shared VPC

----End

9.10 Protecting a CCE Cluster Against Overload

Cluster overload occurs when a Kubernetes cluster's compute, storage, or network resources exceed its processing capacity, leading to exhaustion of key control plane components (like etcd and kube-apiserver) or worker nodes. This can severely degrade cluster performance or even cause operational failures. To prevent this, proactive overload protection is essential. Policies like overload control and LIST request optimization help maintain the stability of core services, ensuring that sudden spikes in loads do not result in service disruptions. This section explores the causes and impacts of cluster overload, explains CCE cluster overload control, and provides best practices for maintaining stability.

Causes of Cluster Overload

Cluster overload affects stability and service continuity within a Kubernetes environment. Common causes include:

- Resource request exceeding cluster capacity: Core components on the cluster control plane include etcd and kube-apiserver. etcd is a backend database that stores all cluster data. kube-apiserver acts as the entry to the control plane and processes requests. kube-apiserver caches cluster data to lessen the burden on etcd. Other core components in the cluster also cache various resources and monitor changes to these resources. When the demand for compute, network, and storage resources surpasses the cluster capacity, these components become heavily loaded. If this load exceeds a certain threshold, the cluster becomes overloaded.
- Lots of data queries (such as multiple LIST requests or a single request retrieving large amounts of data): If a client obtains pod data using field selectors, but kube-apiserver cache does not contain the requested information, the client must fetch the full pod data from etcd, since etcd cannot get data by field. Once retrieved, kube-apiserver deserializes the pod data into structured objects in memory, traverses each pod to match the requested fields, and returns the obtained serialized results. When many concurrent queries occur, resource utilization of each component increases

sharply, leading to issues such as etcd latency spikes, OOM errors in kubeapiserver, and control loop imbalances. Consequently, the entire cluster becomes overloaded.



Figure 9-18 Example of a large amount of data obtained from a client

Impacts of Cluster Overload

When a cluster experiences overload, Kubernetes API response delay increase, and resource utilization on control plane nodes rises, affecting both the control plane and related services. Below are key areas impacted:

- **Kubernetes resource management**: Operations such as creating, deleting, updating, or obtaining Kubernetes resources may fail.
- **Kubernetes leader election**: Overload can prevent a leader node lease renewal from completing on time. If the lease expires, the leader node loses its leadership role, triggering re-election. This can cause temporary service interruptions, task migrations, scheduling delays, and fluctuations in cluster performance.
- Cluster management failures: Severe overload may make a cluster unavailable, preventing management operations like node creation or deletion.

CCE Overload Control

• **Overload control**: CCE clusters have supported overload control since v1.23, which reduces the number of LIST requests outside the system when the control plane experiences high resource usage pressure. To use this function, enable overload control for your clusters. For details, see **Enabling Overload Control for a Cluster**.

- **Optimized processes on LIST requests**: Starting from CCE clusters of v1.23.8r0 and v1.25.3-r0, processes on LIST requests have been optimized. Even if a client does not specify the **resourceVersion** parameter, kube-apiserver responds to requests based on its cache to avoid additional etcd queries and ensure that the response data is up to date. Additionally, namespace indexes are now added to the kube-apiserver cache. This allows clients to request specified resources in a namespace without needing to fetch full data for that namespace. This effectively reduces the response delay and control plane memory overhead.
- **Refined traffic limit policy on the server**: The API Priority and Fairness (APF) feature is used to implement fine-grained control on concurrent requests. For details, see **API Priority and Fairness**.

Configuration Suggestions

When running services in Kubernetes clusters, the cluster performance and availability are influenced by various factors, including the cluster scale, number and volume of resources, and resource access. CCE has optimized cluster performance and availability based on cloud native practices and has developed suggestions to protect against cluster overload. You can use these suggestions to ensure that your services run stably and reliably over the long term.

Category	Suggestion	Billing
Cluster	Keep the cluster version up to date.	N/A
	Enable overload control.	N/A
	Change the cluster scale.	The larger the cluster management scale, the higher the price. For details, see CCE Price Calculator .
	Control data volume of resources.	N/A
	Use multiple clusters.	The costs vary with the number of clusters and the cluster management scale. For details, see CCE Price Calculator .

Category	Suggestion	Billing
O&M	Enable observability.	 If Prometheus is used to monitor metrics of components on the controller plane nodes, the monitoring center will report related metrics to AOM. If the metrics reported to AOM are within the basic container metrics, there will be no extra costs. If the metrics are beyond the basic container metrics, you will be billed based on the number of reported metrics, retention period, and amount of data dumped. For details, see AOM Price Calculator. After log collection is enabled, you will be billed for what you use. For details, see LTS Price Calculator.
	Clear unused resources.	N/A
Application	Optimize the client access mode.	N/A

Keeping the Cluster Version Up to Date

As the CCE cluster version evolves, new overload protection features and optimizations are regularly introduced. It is recommended that you promptly upgrade your clusters to the latest version. For details, see **Cluster Upgrade Overview**.

Enabling Overload Control

After overload control is enabled, concurrent LIST requests outside the system will be dynamically controlled based on the resource demands received by master nodes to ensure the stable running of the master nodes and the cluster.

For details, see Enabling Overload Control for a Cluster.

Changing the Cluster Scale

If the resource usage on the master nodes in a cluster remains high for a long time, for example, the memory usage is greater than 85%, it is recommended that you promptly increase the cluster management scale. This will prevent the cluster from becoming overloaded during sudden traffic surges. For details, see **Changing a Cluster Scale**.

- The cluster performance improves as the control plane node specifications become higher and the management scale of the cluster increases.
- The CCE cluster management scale is the maximum number of nodes that a cluster can manage. It is used as a reference during service deployment planning, and the actual quantity of nodes in use may not reach the limit. The actual scale depends on various factors, including the type, quantity, and size of resource objects in the cluster, as well as the amount of external access to the cluster control plane.

Controlling Data Volume of Resources

When the resource data volume in a cluster is too large, it can negatively impact etcd performance, including data read and write latency. Additionally, if the data volume of a single type of resource is too large, the control plane consumes a significant number of resources when a client requests all the resources. To avoid these issues, it is recommended that you keep both the etcd data volume and the data volume of a single type of resources under control.

Cluster Scale	50 Nodes	200 Nodes	1000 Nodes	2000 Nodes
Total etcd data capacity	500 MiB	1 GiB	4 GiB	8 GiB
etcd data volume of a single type of resources	50 MiB	100 MiB	400 MiB	800 MiB

Table 9-7 Recommended maximum etcd data volume for different cluster scales

Using Multiple Clusters

The Kubernetes architecture has a performance bottleneck, meaning that the scale of a single cluster cannot be expanded indefinitely. If your cluster has 2,000 worker nodes, it is necessary to split the services and deploy them across multiple clusters. If you encounter any issues with splitting a cluster, submit a service ticket for technical support.

Enabling Observability

Observability is crucial for maintaining the reliability and stability of clusters. By using monitoring, alarms, and logs, administrators can gain a better understanding of the clusters' performance, promptly identify any issues, and take corrective action in a timely manner.

Monitoring configurations

• You can check the monitoring information about master nodes on the **Overview** page of the CCE cluster console.

Figure 9-19 Viewing master node monitoring information

Master Nodes ⑦						
 Overload control not enable 	bled. Overloaded master r	nodes will not reject	external traffic.			Enable
Select a property or enter a	a keyword.					0
Name 😝	Status 🔶	AZ ⊜	Overload Level Overload Level	U Usage ⇔	Memory Usage	Operation
master-1	O Running	AZ3	Normal	60.2%		Monitor

 You can use Prometheus to monitor the metrics of master node components, especially the memory usage, resource quantity, QPS, and request latency of kube-apiserver. For details, see Monitoring Metrics of Master Node Components Using Prometheus.

Alarm configurations

Alarms are an additional feature of monitoring. Alarms are generated to administrators in a timely manner when a cluster experiences malfunctions, allowing for prompt resolution of any issues. You can configure alarms for metrics such as memory usage, resource quantity, QPS, and request latency of kubeapiserver as needed. For details, see **Configuring Custom Alarms on CCE**.

NOTE

Monitoring metrics such as resource quantity, QPS, and request latency do not have a fixed boundary between normal and abnormal metrics due to variations in service scenarios. As a result, these metrics are considered normal as long as they do not impact service stability. Typical alarm thresholds cannot be defined. To address this, you can observe metric data when services are running stably and configure appropriate alarm thresholds based on the normal fluctuation range of resource usage. Alternatively, you can use the changes of metric data in a unit of time as the alarm detection object.

Logging configurations

Kubernetes logs allow you to locate and rectify faults. The kube-apiserver component logs contain details about client requests, such as the request source, processing time, and reasons for any exceptions. These logs are useful for tracing the source of issues and analyzing problems related to overload. For details, see **Collecting Control Plane Component Logs**.

Clearing Unused Resources

To prevent a large number of pending pods from consuming extra resources on the control plane, it is recommended that you promptly clear up Kubernetes resources that are no longer in use, such as ConfigMaps, Secrets, and PVCs.

Category	Description	Recommended Solution
Number of Services in a namespace	In a Kubernetes cluster, kubelet transforms the Service information specified in the cluster into environment variables and then injects them into the pod that is running on the node. This enables the pod to access these environment variables and locate the Service, facilitating communication with it. However, if there are numerous Services defined in a namespace, kubelet may have to inject a larger number of environment variables into the pod, which could potentially delay the pod's startup process. In severe cases, the pod may even fail to start.	Do not configure these environment variables and set enableServiceLinks in podSpec to false . For details, see Accessing the Service.
Total number of Services in a cluster	If the number of Services increases, the number of kube- proxy network rules increases, affecting network performance.	It is recommended that the total number of Services be less than or equal to 10,000. It is recommended that the total number of LoadBalancer Services be less than or equal to 500.
Maximum number of endpoints in a Service	When a Service is associated with a large number of endpoints, each time the endpoints are updated, a large amount of data is synchronized between the control plane component kube-apiserver and kube-proxy on the node, which may cause significant pressure on network and system resources. NOTE Kubernetes provides the EndpointSlices feature to efficiently update endpoints. This feature is enabled by default in clusters of v1.19 and later versions, and enters the stable state in clusters of v1.21.	It is recommended that the number of backend pods of a single Service endpoint be less than or equal to 3000.

 Table 9-8 Resource limits

Category	Description	Recommended Solution
Total number of endpoints of all Services	If a Service has an excessive number of endpoints, the API server will have to handle a larger amount of data, potentially leading to a high workload on the API server and a decline in network performance.	It is recommended that the total number of endpoints associated with all Services be less than or equal to 64,000.
Number of pending pods	When there is a high number of pods waiting to be scheduled in a cluster, new pods may remain in the pending state for an extended period, causing delays in their allocation to suitable nodes. If a pod cannot be scheduled for a prolonged period, the scheduler generates events at regular intervals, leading to an excessive number of events.	It is recommended that the total number of pending pods be less than or equal to 10,000.

Optimizing the Client Access Mode

• To avoid frequent LIST queries, it is best to use the client cache mechanism when retrieving cluster resource data multiple times. It is recommended that you communicate with clusters using informers and listers. For details, see the **Go documentation**.

If a LIST query must be used, you can:

- Obtain needed data from the kube-apiserver cache first and avoid making additional queries on etcd data. For clusters earlier than v1.23.8r0 or v1.25.3-r0, you can set **resourceVersion** to **0**. In clusters of v1.23.8r0, v1.25.3-r0, and later versions, CCE has improved data retrieval and ensured that the cached data is up to date. By default, you can access the required data from the cache.
- Accurately define the query scope to avoid retrieving irrelevant data and using unnecessary resources. For example:
 # client-go Code example for obtaining pods in a specified namespace k8sClient.CoreV1().Pods("*<your-namespace>*").List(metav1.ListOptions{})
 # kubectl Command example for obtaining pods in a specified namespace kubectl get pods -n *<your-namespace>*
- Use the more efficient Protobuf format instead of the JSON format. By default, Kubernetes returns objects serialized to JSON with content type **application/json**. This is the default serialization format for the API. However, clients may request the more efficient Protobuf representation of these objects for better performance. For details, see **Alternate representations of resources**.

Helpful Links

- Which Resource Quotas Should I Pay Attention To When Using CCE?
- For details about how to properly plan the VPC network and container network in a cluster, see **Planning CIDR Blocks for a Cluster**.
- For details about how to ensure workload stability and reliability, see **Recommended Configurations for Workloads**.
- For details about how to prevent and avoid high-risk operations on a cluster, see **High-Risk Operations**.

9.11 Managing Costs for a Cluster

The key to optimizing cluster costs is to maximize the utilization of cluster resources and minimize unnecessary expenses. It is important to note that cost optimization goes beyond just reducing resources; it also involves finding a balance between cost optimization and cluster reliability. This section provides a summary of the best practices for cluster cost optimization, which will help you efficiently manage cluster expenses and enhance overall efficiency.

Using Appropriate Cluster Configurations

Before setting up a cluster, it is important to assess the resource needs of your applications. This will help you choose the appropriate cluster type, node instance type, and cluster billing mode, all of which can contribute to building a cost-effective cluster.

Selecting a Cluster Type

CCE provides many cluster types. You can select a proper one based on your service characteristics. The following table lists the differences between these clusters.

Categor y	CCE Standard	CCE Turbo	CCE Autopilot
Descripti on	An enterprise-level container service on Kubernetes	Hardware-software synergy for extra performance	A Kubernetes- compatible serverless container service
Manage d object	Clusters, nodes, and workloads	Clusters, nodes, and workloads	Fully hosted nodes
Cluster scale	On-demand adjustment	On-demand adjustment	Automatic adjustment
Node	Multiple flavors available, custom node creation or deletion	Multiple flavors available, custom node creation or deletion	Automatic, flexible node allocation for containers

Categor y	CCE Standard	CCE Turbo	CCE Autopilot
Comput e	Heterogeneous compute including x86, Arm, and NPUs	Heterogeneous compute including x86, Arm, and NPUs	Heterogeneous compute including x86, Arm, and NPUs
Scheduli ng	Proprietary Volcano for various scheduling policies and improved task execution efficiency	Hybrid scheduling for improved cluster resource utilization	Intelligent scheduling for starting containers in seconds and automatic load balancing
Network	VPC network overlaid with container network	VPC network and container network flattened into one layer, so there is zero performance loss	VPC network and container network flattened into one layer, so there is zero performance loss
Security	Container network access control based on network policies	Kata Containers that allow containers to run inside lightweight VMs	Dedicated cloud native OS that allows automatic vulnerability fixing

For details, see Comparison Between Cluster Types.

Selecting a Node Flavor

ECSs come in various flavors, each offering different computing and storage capabilities. Typically, higher specifications (such as CPU and memory) and specialized features (like GPUs and NPUs) result in higher costs per node. It is important to configure stable, cost-effective ECSs that align with the specific needs of your services.

Selecting a Billing Mode for a Node

Different services have varying resource usage periods and stability requirements. To achieve cost-effectiveness, you can choose the appropriate billing mode based on the service characteristics.

Billing Mode	Description
Yearly/Monthly	Yearly/Monthly is a prepaid mode in which you pay for a service before using it. Your bill is generated based on the required duration you specify in the order. The longer the subscription period, the higher the discount. Yearly/Monthly billing is a good option for long-term, stable services.

Billing Mode	Description
Pay-per-use	Pay-per-use is a postpaid billing mode. You pay as you go and just pay for what you use. The prices are calculated by the second but billed every hour. Pay-per-use billing allows you to flexibly adjust resource usage. You neither need to prepare for resources in advance, nor end up with excessive or insufficient preset resources.
	It is a good option for scenarios where there are sudden traffic bursts, such as e-commerce promotions.
Spot pricing	Spot pricing is a postpaid billing mode. The prices are adjusted gradually based on long-term trends in supply and demand for spot instance capacity. The resource usage is calculated by the second but billed every hour.
	You need to set a maximum price you are willing to pay for a spot instance. If inventory resources are insufficient or the market price rises above your maximum price, the spot instance will be reclaimed.
	NOTICE Spot instances are ideal for stateless, cost-sensitive applications that can tolerate interruptions. A spot instance is not recommended for workloads that need to run for a long time or that require high stability.

For details, see **Billing Items**.

Clearing Idle Resources in a Timely Manner

It is a good option to identify and clear idle cloud services or resources in clusters in a timely manner, such as unused ECSs, EVS disks, OBS buckets, ELB load balancers, and EIPs.

Optimizing Resource Configuration for a Workload

Setting resource requests and limits too high leads to resource wastage, while setting them too low affects workload stability. By properly configuring resource requests and limits, cluster resource utilization can be improved, resulting in reduced costs.

Configuring Proper Resource Requests and Limits

To ensure that your workloads have enough resources and to avoid wasting resources due to excessive requests, it is important to configure appropriate requests and limits.

You can optimize the configuration of your existing applications using the resource specification recommendation feature provided by CCE. This feature uses the Volcano Scheduler to analyze historical CPU and memory usage data of your workloads and provide recommended requests. You can directly apply these recommended values to your applications. When making changes to the

configuration, the recommended value will be set as the default for a request. As for the limit, it will be determined by taking the larger value between the request multiplied by 1.2 and the current limit. You have the option to include additional configurations based on this value.

However, note that the recommended values generated based on historical data may not be completely accurate. They should be used as a reference only. It is important to adjust the resource requests and limits based on the actual running status of your services.

Managing Quotas for a Namespace

Quota management sets limits on the total number of resources that teams and users can use when they share cluster resources. These resources include the number of objects of a specific type created in a namespace, as well as the total number of compute resources like CPUs and memory used by these objects.

This approach help minimize unnecessary resource overhead.

For details, see Configuring Resource Quotas.

Configuring Auto Scaling for a Cluster

CCE provides auto scaling in seconds. It automatically adjusts compute resources based on preset policies and your service needs to ensure that the number of cloud servers or containers increases or decreases with service load. This ensures stable, healthy services, improves cluster resource utilization, and reduces costs.

For details, see Workload Scaling Rules.

Enabling Auto Scaling for an Application

CCE provides auto scaling for applications. This feature enables applications that experience traffic surges or periodic peak and off-peak hours to automatically adjust compute resources.

Application scaling on demand (HPA)

Application auto scaling helps dynamically adjust compute resources based on service requirements and policies. It enables quick scale-out during peak hours and scale-in during off-peak hours, optimizing resource utilization and reducing costs.

The following table	lists the auto scal	ing approaches suppo	orted by CCE.

Policy Name	Description
HPA	Scales Deployments based on metrics like CPU usage and memory usage. For details, see Horizontal Pod Autoscaling . HPA adds cooldown time windows and scaling thresholds for applications based on Kubernetes HPA.
	HPA applies to scenarios where services experienced fluctuating traffic, many services are deployed, and frequent scaling is required.

Policy Name	Description
CronHPA	Scales Deployment periodically (daily, weekly, monthly, or yearly at a specific time). CronHPA applies to scenarios where the application resource usage changes periodically.

Burst scaling (interconnected with CCI)

In short-term high-load scenarios, besides HPA, pods in a CCE cluster can be scaled to CCI. There is no need to create new nodes, reducing resource consumption caused by scale-out. The CCE Cloud Bursting Engine for CCI add-on must be installed in the cluster. For details, see **Elastic Scaling of CCE Pods to CCI**.

Scaling a Node

Application auto scaling helps dynamically adjust the number of pods based on workload metrics. If there are not enough cluster resources and new pods cannot run properly, you can add more nodes to the cluster.

For details, see Node Scaling Rules.

The following table lists auto scaling policies that you can select.

Policy Name	Description
Manual scaling	You can manually scale in or out nodes in a node pool. If the resources of the selected flavor are insufficient or the quota is insufficient, the scale-out will fail.
Auto scaling	CCE Cluster Autoscaler automatically scales in or out nodes in a cluster based on the pod scheduling status and resource usage. It supports multiple scaling modes, such as multi-AZ, multi-pod-specifications, metric triggering, and periodic triggering, to meet the requirements of different node scaling scenarios.
	• Scale-out: Autoscaler checks all unscheduled pods every 10 seconds and selects a node pool that meets the requirements for scale-out based on the policy you set.
	• Scale-in: Autoscaler scans all nodes every 10 seconds. If the number of pod requests on a node is less than the custom scale-in threshold (in percentage), Autoscaler will check whether pods on the current node can be migrated to other nodes.

Optimizing Application Scheduling

During cloud native progress, it is important to strike a balance between performance and service quality. This can be achieved by carefully considering the service deployment solutions and architectures. Depending on your specific service scenarios, you can choose an appropriate scheduling solution to optimize resource utilization and manage costs efficiently.

Using Cloud Native Hybrid Deployment

If your service meets the following requirements, it is recommended that you use this capability:

- Nodes are deployed in different clusters. They cannot share compute resources with each other, resulting in an increase in resource fragments.
- The node flavors are not ideal for applications that undergo frequent changes. At first, the node flavors match the application requirements, resulting in a high resource allocation rate. However, as the applications evolve, their resource demands change, causing a significant difference in the ratio of requested resources to node flavors. This leads to a decrease in the allocation rate of node resources and an increase in compute resource fragments.
- There are a large number of reserved resources. Online services experience daily peaks and troughs. To ensure service performance and stability, users apply for resources based on peak usage, which may result in many idle resources in the cluster during certain times.
- Online and offline services are deployed in separate Kubernetes clusters, and resources cannot be shared between them at different time. This means that during off-peak hours for online services, the resources cannot be used by offline services.

The following table lists cloud native hybrid deployment features that can help you improve resource utilization, reduce costs, improve efficiency in the scenarios mentioned earlier.

Feature	Description
Dynamic resource oversubscription	Based on the types of online and offline jobs, Volcano is used to optimize cluster resource utilization by using the requested but unused resources (the difference between the requested and used resources) for resource oversubscription and hybrid deployment.
	For details, see Dynamic Resource Oversubscription .
CPU burst	CPU burst is an elastic traffic limiting mechanism that allows temporarily exceeding the CPU limit to reduce the long-tail response time of services and improve the quality of latency-sensitive services. For details, see CPU Burst .
Guaranteed egress network bandwidth	The egress network bandwidth used by online and offline services is balanced to ensure enough network bandwidth for online services.
	For details, see Guaranteed Egress Network Bandwidth.

Enabling Resource Usage-based Scheduling

The Volcano Scheduler is used to improve cluster resource usage. It provides bin packing, descheduling, node pool affinity, and load-aware scheduling policies.

Scheduling Policy	Description
Bin packing	Bin packing is an optimization algorithm that aims to reduce cluster resource fragments. After bin packing is enabled for cluster workloads, the scheduler preferentially schedules pods to nodes with high resource allocation. This reduces resource fragments on each node and improves cluster resource utilization. For details, see Bin Packing .
Desiste dulla a	The Malance Cale data are provided and the today
Descheduling	The Volcano Scheduler can remove pods that do not meet the configured policies and reschedule them according to those policies. This helps balance the cluster loads and minimize resource fragmentation.
	For details, see Descheduling .
Node pool affinity	When it comes to scenarios like node pool replacement and rolling node upgrade, it becomes necessary to replace an old node pool with a new one. To prevent the node pool replacement from affecting services, it is recommended that you enable soft affinity, which allows for scheduling service pods to the new node pool. For details, see Node Pool Affinity .
Load-aware scheduling	Volcano Scheduler offers CPU and memory load- aware scheduling for pods and preferentially schedules pods to the node with the lightest load to balance node loads. This prevents an application or node failure due to heavy loads on a single node. For details, see Load-aware Scheduling .

Enabling Priority-based Scheduling and Preemption

A pod priority indicates the importance of a pod relative to other pods. Volcano supports pod PriorityClasses in Kubernetes. After PriorityClasses are configured, the scheduler preferentially schedules high-priority pods. When cluster resources are insufficient, the scheduler will proactively evict low-priority pods to make it possible to schedule pending high-priority pods. For details, see **Priority-based Scheduling and Preemption**.

The following table lists the types of priority-based scheduling and preemption supported by CCE.

Scheduling Type	Description
Priority-based scheduling	The scheduler preferentially guarantees the running of high-priority pods, but will not evict low-priority pods that are running. Priority-based scheduling is enabled by default and cannot be disabled.
Priority-based preemption	When cluster resources are insufficient, the scheduler will proactively evict low-priority pods to make it possible to schedule pending high-priority pods.

Sharing GPUs

GPU virtualization allows for the separation of compute and GPU memory, optimizing the utilization of GPUs. CCE GPU virtualization leverages the proprietary xGPU virtualization technology to dynamically separate GPU memory and compute. This virtualization solution offers greater flexibility compared to static allocation. While ensuring maximum service stability, you have the freedom to define the number of GPUs to be used, thereby enhancing GPU utilization.

For details, see **GPU Virtualization**.

Enabling AI Performance-based Scheduling

In AI and big data collaborative scheduling scenarios, Volcano Dominant Resource Fairness (DRF) and group can be used to improve training performance and resource utilization.

DRF

DRF is a scheduling algorithm based on the dominant resource of a container group. DRF can be used to enhance the service throughput of a cluster, shorten the overall service execution time, and improve training performance. It is suitable for batch AI training and big data jobs. For details, see **DRF**.

In actual services, limited cluster resources are often allocated to multiple users. Each user has the same rights to obtain resources, but the number of resources they need may be different. It is crucial to fairly allocate resources to each user. A common scheduling algorithm is the max-min fairness share, which allocates resources to meet users' minimum requirements as far as possible and then fairly allocates the remaining resources. The rules are as follows:

- Resources are allocated in order of increasing demand.
- No source gets a resource share larger than its demand.
- Sources with unsatisfied demands get an equal share of the resource.

Gang

Gang scheduling meets the scheduling requirements of "All or nothing" in the scheduling process and avoids the waste of cluster resources caused by arbitrary scheduling of pods. It is mainly used in scenarios that require multi-process collaboration, such as AI and big data scenarios. Gang scheduling effectively resolves pain points such as resource waiting or deadlocks in distributed training

jobs, thereby significantly improving the utilization of cluster resources. For details, see **Gang**.

The Gang scheduler algorithm checks whether the number of scheduled pods in a job meets the minimum requirements for running the job. If yes, all pods in the job will be scheduled. If no, the pods will not be scheduled.

It is applicable to the scenarios where multi-process collaboration is required.

- Al scenarios typically involve complex processes. Data ingestion, data analysts, data splitting, trainers, serving, and logging which require a group of containers to work together are suitable for container-based Gang scheduling.
- Multi-thread parallel computing communication scenarios under MPI computing framework are also suitable for Gang scheduling because primary and secondary processes need to work together.
- Containers in a pod group are highly correlated, and there may be resource contention. The overall scheduling allocation can effectively resolve deadlocks.

If cluster resources are insufficient, Gang scheduling can significantly improve the utilization of cluster resources.

Enabling NUMA Affinity Scheduling

When working with high-performance computing (HPC), real-time applications, or memory-intensive workloads that require frequent communication between CPUs, accessing nodes across non-uniform memory access (NUMA) in a cloud native environment can lead to decreased system performance due to increased latency and overhead. Volcano's NUMA affinity scheduling policy resolves the issue by scheduling pods to the worker node that requires the least number of cross-NUMA nodes. This reduces data transmission overheads, optimizes resource utilization, and enhances overall system performance.

Volcano targets to lift the limitation to make scheduler NUMA topology aware so that:

- Pods are not scheduled to the nodes that NUMA topology does not match.
- Pods are scheduled to the most suitable node for NUMA topology.

For details, see NUMA Affinity Scheduling.

Configuring Application Scaling Priority Policies

With application scaling priority policies, you have precise control over the scaling priorities of pods on different types of nodes, allowing for optimized resource management. The application scaling priority policies include the following aspects:

- Scale-out: Volcano schedules new pods in a cluster based on preset node priority for scale-out.
- Scale-in: When a workload is specified, Volcano scores the workload based on preset node priority to determine pod deletion sequence during scale-in.

If the default scaling priority policy is applied, pods will be scheduled first to yearly/monthly nodes during scale-out, followed by pay-per-use nodes and virtual-kubelet nodes (scaling pods to CCI). During scale-in, pods are deleted sequentially from virtual-kubelet nodes (scaling pods to CCI), pay-per-use nodes, and yearly/

monthly nodes. You can adjust the scaling priority policies based on your service scenarios. For details, see **Application Scaling Priority Policies**.

Establishing Resource and Cost Monitoring

Cost Insights uses in-house cost profile algorithms to split costs by department, cluster, namespace, or application, based on your bills and cluster resource usage. It allows you to analyze cluster costs and resource usage and identify resource waste for cost optimization. For details, see **Cost Insights**.

10 Networking

10.1 Planning CIDR Blocks for a Cluster

Before creating a cluster on CCE, determine the number of VPCs, number of subnets, container CIDR blocks, and Services for access based on service requirements.

This topic describes the addresses in a CCE cluster in a VPC and how to plan CIDR blocks.

Notes and Constraints

To access a CCE cluster through a VPN, ensure that the VPN does not conflict with the VPC CIDR block where the cluster resides and the container CIDR block.

Basic Concepts

• VPC CIDR Block

Virtual Private Cloud (VPC) enables you to provision logically isolated, configurable, and manageable virtual networks for cloud servers, cloud containers, and cloud databases. You can configure IP address ranges, subnets, security groups, and more in a VPC, and assign EIPs and bandwidth to build your service system.

• Subnet CIDR Block

A subnet is a network that manages ECS network planes. It supports IP address management and DNS. ECSs in a subnet are assigned IP addresses from this subnet.



Figure 10-1 VPC CIDR block architecture

By default, ECSs in all subnets of the same VPC can communicate with one another, but ECSs in different VPCs cannot.

You can create a VPC peering connection to enable ECSs in different VPCs to communicate with each other.

• Container (Pod) CIDR Block

Pod is a Kubernetes concept. Each pod has an IP address.

When creating a cluster on CCE, you can specify the pod (container) CIDR block. The container CIDR block cannot overlap with the VPC CIDR block or subnet CIDR block. For example, if the subnet CIDR block is 192.168.0.0/16, the container CIDR block of the cluster cannot be 192.168.0.0/18 or 192.168.64.0/18 because these addresses are covered by 192.168.0.0/16.

• Container Subnet (Only for CCE Turbo Clusters)

In a CCE Turbo cluster, a container is assigned an IP address from the CIDR block of a VPC. The container subnet can overlap with the subnet CIDR block. Note that the subnet you select determines the maximum number of pods in the cluster.

• Service CIDR Block

Service is also a Kubernetes concept. Each Service has an address. When creating a cluster on CCE, you can specify the Service CIDR block. Similarly, the Service CIDR block cannot overlap with the VPC CIDR block, subnet CIDR block, or container CIDR block. The Service CIDR block can be used only within a cluster.

Single-VPC Single-Cluster Scenarios

CCE Clusters: include clusters in VPC network model and container tunnel network model. **Figure 10-2** shows the CIDR block planning of a cluster.

- VPC CIDR Block: specifies the VPC CIDR block where the cluster resides. The size of this CIDR block affects the maximum number of nodes that can be created in the cluster.
- Subnet CIDR Block: specifies the subnet CIDR block where the node in the cluster resides. The subnet CIDR block is included in the VPC CIDR block.

Different nodes in the same cluster can be allocated to different subnet CIDR blocks.

- Container CIDR block: cannot overlap with the VPC CIDR block or subnet CIDR block.
- Service CIDR block: cannot overlap with the VPC CIDR block, subnet CIDR block, or container CIDR block.

Figure 10-2 Network CIDR block planning in single-VPC single-cluster scenarios (CCE cluster)



Figure 10-3 shows the CIDR block planning for a **CCE Turbo cluster** (Cloud Native Network 2.0).

- VPC CIDR Block: specifies the VPC CIDR block where the cluster resides. The size of this CIDR block affects the maximum number of nodes that can be created in the cluster.
- Subnet CIDR Block: specifies the subnet CIDR block where the node in the cluster resides. The subnet CIDR block is included in the VPC CIDR block. Different nodes in the same cluster can be allocated to different subnet CIDR blocks.
- Container Subnet CIDR Block: The container subnet is included in the VPC CIDR block and can overlap with the subnet CIDR block or even be the same as the subnet CIDR block. Note that the container subnet size determines the maximum number of containers in the cluster because IP addresses in the VPC are directly allocated to containers. Set a larger IP address segment for the container subnet to prevent insufficient container IP addresses.
- Service CIDR Block: cannot overlap with the subnet CIDR block or the container CIDR block.



Figure 10-3 Network CIDR block planning in single-VPC single-cluster scenarios (CCE Turbo cluster)

Single-VPC Multi-Cluster Scenarios

VPC network model

Pod packets are forwarded through VPC routes. CCE automatically configures a routing table on the VPC routes to each container CIDR block. The network scale is limited by the VPC route table. Figure 10-4 shows the CIDR block planning of the cluster.

- VPC CIDR Block: specifies the VPC CIDR block where the cluster resides. The size of this CIDR block affects the maximum number of nodes that can be created in the cluster.
- Subnet CIDR Block: The subnet CIDR block in each cluster cannot overlap with the container CIDR block.
- Container CIDR Block: If multiple VPC network model clusters exist in a single VPC, the container CIDR blocks of all clusters cannot overlap because the clusters use the same routing table. In this case, if the node security group allows container CIDR block from the peer cluster, pods in one cluster can directly access pods in another cluster through the pod IP addresses.
- Service CIDR Block: can be used only in clusters. Therefore, the Service CIDR blocks of different clusters can overlap, but cannot overlap with the subnet CIDR block and container CIDR block of the cluster.



Figure 10-4 VPC network - multi-cluster scenario

Tunnel network model

Though at some cost of performance, the tunnel encapsulation enables higher interoperability and compatibility with advanced features (such as network policy-based isolation), meeting the requirements of most applications. Figure 10-5 shows the CIDR block planning of the cluster.

- VPC CIDR Block: specifies the VPC CIDR block where the cluster resides. The size of this CIDR block affects the maximum number of nodes that can be created in the cluster.
- Subnet CIDR Block: The subnet CIDR block in each cluster cannot overlap with the container CIDR block.
- Container CIDR Block: The container CIDR blocks of all clusters can overlap. In this case, pods in different clusters cannot be directly accessed through pod IP addresses. Services are needed for accessing pods in different clusters. The LoadBalancer Services are recommended.
- Service CIDR Block: can be used only in clusters. Therefore, the Service CIDR blocks of different clusters can overlap, but cannot overlap with the subnet CIDR block and container CIDR block of the cluster.



Figure 10-5 Tunnel network - multi-cluster scenario

Cloud Native 2.0 network model (CCE Turbo Clusters)

In this mode, container IP addresses are allocated from the VPC CIDR block. ELB passthrough networking is supported to direct access requests to containers. Security groups and multiple types of VPC networks can be bound to deliver high performance.
- VPC CIDR Block: specifies the VPC CIDR block where the cluster resides. In a CCE Turbo cluster, the CIDR block size affects the total number of nodes and containers that can be created in the cluster.
- Subnet CIDR Block: There is no special restriction on the subnet CIDR blocks in CCE Turbo clusters.
- Container Subnet: The CIDR block of the container subnet is included in the VPC CIDR block. Container subnets in different clusters can overlap with each other or overlap with the subnet CIDR block. However, you are advised to stagger the container CIDR blocks of different clusters and ensure that the container subnet CIDR blocks have sufficient IP addresses. In this case, if the elastic network interface security group of the cluster allows the container CIDR block of the peer cluster, pods in different clusters can directly access each other through IP addresses.
- Service CIDR Block: can be used only in clusters. Therefore, the Service CIDR blocks of different clusters can overlap, but cannot overlap with the subnet CIDR block and container subnet CIDR block of the cluster.



Figure 10-6 Cloud Native 2.0 network - multi-cluster scenario

Clusters using different networks

When a VPC contains clusters created with different network models, comply with the following rules when creating a cluster:

- VPC CIDR Block: In this scenario, all clusters are located in the same VPC CIDR block. Ensure that there are sufficient available IP addresses in the VPC.
- Subnet CIDR Block: Ensure that the subnet CIDR block does not overlap with the container CIDR block. Even in some scenarios (for example, coexistence with CCE Turbo clusters), the subnet CIDR block can overlap with the container (subnet) CIDR block. However, this is not recommended.
- Container CIDR Block: Ensure that the container CIDR blocks of clusters in **VPC network model** do not overlap.
- Service CIDR Block: The Service CIDR blocks of all clusters can overlap, but cannot overlap with the subnet CIDR block and container CIDR block of the cluster.

Cross-VPC Cluster Interconnection

If VPCs cannot communicate with each other, a VPC peering connection is used to ensure communication between VPCs. When two VPC networks are

interconnected, you can configure the packets to be sent to the peer VPC in the route table. For details, see **VPC Peering Connection Overview**.

Clusters using VPC networks

To allow clusters that use VPC networks to access each other across VPCs, add routes to the two ends of the VPC peering after a VPC peering connection is created.



Figure 10-7 VPC network - VPC interconnection scenario

When creating a VPC peering connection between containers across VPCs, pay attention to the following points:

- The VPC to which the clusters belong must not overlap. In each cluster, the subnet CIDR block cannot overlap with the container CIDR block.
- The container CIDR blocks of clusters at both ends cannot overlap, but the Service CIDR blocks can.
- If the request end cluster uses the VPC network, check whether the node security group in the destination cluster allows the container CIDR block of the request end cluster. If yes, pods in one cluster can directly access pods in another cluster through the pod IP address. Similarly, if nodes running in the clusters at the two ends of the VPC peering connection need to access each other, the node security group must allow the VPC CIDR block of the peer cluster.
- You need to add routes for accessing the peer network CIDR block to the VPC routing tables at both ends. For example, you need to add a route for accessing the CIDR block of VPC 2 to the route table of VPC 1, and add a route for accessing VPC 1 to the route table of VPC 2.
 - Add the VPC CIDR block of the peer cluster: After the route of the VPC CIDR block is added, a pod in a cluster can access another cluster node.
 For example, the pod can access the port of a NodePort Service.
 - Add peer container CIDR block: After the route of the container CIDR block is added, a pod can directly access pods in another cluster through the container IP addresses.

Figure 10-8 Adding the peer container CIDR block to the local route on the VPC console

< rtb-vpc-373896-1		G Feedback
Summary Associated Subnets		
Name <u></u> D bit 1187 354 455 555 3984450 Ф Decorption - <u>2</u>	Add Route Route Table : rtb-ypc-337386-1(Default)	×
Routes Date: Add Route Replicate Route: Q Learn 1 Destination (P) (P) (P) (P)	Destination Next Hop Type Next Hop © Description 172 158.018 VPC pown • •	Enter a destination Q C
✓ Local	OK Carcol	s instance communication within a VPC Mindify Delete

Clusters using tunnel networks

To allow clusters that use tunnel networks to access each other across VPCs, add routes to the two ends of the VPC peering after a VPC peering connection is created.





Pay attention to the following:

- The VPCs of the peer clusters must not overlap.
- The container CIDR blocks of all clusters can overlap, so do the Service CIDR blocks.
- If the request end cluster uses the tunnel network, check whether the node security group in the destination cluster allows the VPC CIDR block (including the node subnets) of the request end cluster. If yes, nodes in one cluster can access nodes in another cluster. However, pods in different clusters cannot be directly accessed using pod IP addresses. Access between pods in different clusters requires Services. The LoadBalancer Services are recommended.
- The VPC CIDR block route of the peer cluster must be added to the VPC routing tables of both ends. For example, you need to add a route for accessing the CIDR block of VPC 2 to the route table of VPC 1, and add a route for accessing VPC 1 to the route table of VPC 2. After the route of the VPC CIDR block is added, the pod can access another cluster node, for example, accessing the port of a NodePort Service.

Figure 10-10 Adding the subnet CIDR block of the peer cluster node to the local route on the VPC console



Clusters using Cloud Native 2.0 networks (CCE Turbo clusters)

After creating a VPC peering connection, add routes of the VPC peering connection to both ends so that the two VPCs can communicate with each other. Pay attention to the following:

- The VPCs of the clusters at the two ends must not overlap.
- If the request end cluster uses the Cloud Native 2.0 network, check whether the elastic network interface security group (named in the format of *{Cluster name}-cce-eni-{Random ID}*) of the destination cluster allows the VPC CIDR block (including the node subnets and container CIDR block) of the request end cluster. If yes, pods in one cluster can directly access pods in another cluster through the pod IP addresses. Similarly, if nodes in the clusters at the two ends of the VPC peering need to access each other, allow the VPC CIDR block of the peer cluster in the node security group (named in the format of *{Cluster name}-cce-node-{Random ID}*).
- The VPC CIDR block route of the peer cluster must be added to the VPC routing tables of both ends. For example, you need to add a route for accessing the CIDR block of VPC 2 to the route table of VPC 1, and add a route for accessing VPC 1 to the route table of VPC 2. After the route of the VPC CIDR block is added, the pod can access pod IP addresses or nodes in another cluster.

Clusters using different networks

If clusters using different networks need to communicate with each other across VPCs, every one of them may serve as the request end or destination end. Pay attention to the following:

- The VPC CIDR block to which the cluster belongs cannot overlap with the VPC CIDR block of the peer cluster.
- Cluster subnet CIDR blocks cannot overlap with the container CIDR blocks.
- Container CIDR blocks in different clusters cannot overlap with each other.
- If pods or nodes in different clusters need to access each other, the security groups of the clusters on both ends must allow the corresponding CIDR blocks based on the following rules:
 - If the request end cluster uses the VPC network, the node security group of the destination cluster must allow the VPC CIDR block (including the node subnets and container CIDR block) of the request end cluster.

- If the request end cluster uses the tunnel network, the node security group of the destination cluster must allow the VPC CIDR block (including the node subnets) of the request end cluster.
- If the request end cluster uses the Cloud Native 2.0 network, the elastic network interface security group and node security group of the destination cluster must allow the VPC CIDR block (including node subnets and container CIDR block) of the request end cluster.
- The VPC CIDR block route of the peer cluster must be added to the VPC routing tables of both ends. For example, you need to add a route for accessing the CIDR block of VPC 2 to the route table of VPC 1, and add a route for accessing VPC 1 to the route table of VPC 2. After the route of the VPC CIDR block is added, the pod can access another cluster node, for example, accessing the port of a NodePort Service.

If a cluster uses the VPC network, the VPC routing tables at both ends must contain its container CIDR block. After the container CIDR block route is added, the pod can directly access pods in another cluster through the container IP addresses.

VPC-IDC Scenarios

Similar to the VPC interconnection scenario, some CIDR blocks in the VPC are routed to the IDC. The pod IP addresses of CCE clusters cannot overlap with the addresses within these CIDR blocks. To access the pod IP addresses in the cluster in the IDC, configure the route table to the private line VBR on the IDC.

10.2 Selecting a Network Model

CCE uses proprietary, high-performance container networking add-ons to support the tunnel, Cloud Native 2.0, and VPC network models.

After a cluster is created, the network model cannot be changed.

• **Tunnel network**: The container network is an overlay tunnel network on top of a VPC network and uses the VXLAN technology. This network model is applicable when there is no high requirements on performance. VXLAN encapsulates Ethernet packets as UDP packets for tunnel transmission. Though at some cost of performance, the tunnel encapsulation enables higher interoperability and compatibility with advanced features (such as network policy-based isolation), meeting the requirements of most applications.



Figure 10-11 Container tunnel network

• VPC network: The container network uses VPC routing to integrate with the underlying network. This network model is applicable to performanceintensive scenarios. The maximum number of nodes allowed in a cluster depends on the route quota in a VPC network. Each node is assigned a CIDR block of a fixed size. VPC networks are free from tunnel encapsulation overhead and outperform container tunnel networks. In addition, as VPC routing includes routes to node IP addresses and container network segment, container pods in the cluster can be directly accessed from outside the cluster.



Figure 10-12 VPC network

• **Cloud Native Network 2.0**: The container network deeply integrates the elastic network interface capability of VPC, uses the VPC CIDR block to allocate container addresses, and supports passthrough networking to pods through load balancers.



Figure 10-13 Cloud Native 2.0 network

The following table lists the differences between the network models.

Dimensio n	Tunnel Network	VPC Network	Cloud Native Network 2.0
Applicatio n scenarios	 Low requirements on performance: As the container tunnel network requires additional VXLAN tunnel encapsulation, it has about 5% to 15% of performance loss when compared with the other two container network models. Therefore, the container tunnel network applies to the scenarios that do not have high performance requirements, such as web applications, and middle-end and back-end services with a small number of access requests. Large-scale networking: Different from the VPC network that is limited by the VPC route quota, the container tunnel network does not have any restriction 	 High performance requirements: As no tunnel encapsulation is required, the VPC network model delivers the performance close to that of a VPC network when compared with the container tunnel network model. Therefore, the VPC network model applies to scenarios that have high requirements on performance, such as AI computing and big data computing. Small- and medium-scale networks: Due to the limitation on VPC route tables, it is recommended that the number of nodes in a cluster be less than or equal to 1000. 	 High performance requirements: Cloud Native 2.0 networks use VPC networks to construct container networks, eliminating the need for tunnel encapsulation or NAT when containers communicate. This makes Cloud Native 2.0 networks ideal for scenarios that demand high bandwidth and low latency, such as live streaming and e-commerce flash sales. Large-scale networking: Cloud Native 2.0 networks support a maximum of 2,000 ECS nodes and 100,000 pods.

 Table 10-1
 Network model comparison

Dimensio n	Tunnel Network	VPC Network	Cloud Native Network 2.0
	on the infrastructure. In addition, the container tunnel network controls the broadcast domain to the node level. The container tunnel network supports a maximum of 2000 nodes.		
Core technolog y	OVS	IPVLAN and VPC route	VPC ENI/sub-ENI
Applicable clusters	CCE standard cluster	CCE standard cluster	CCE Turbo cluster
Container network isolation	Kubernetes native NetworkPolicy for pods	No	Pods support security group isolation.
Interconne cting pods to a load balancer	Interconnected through a NodePort	Interconnected through a NodePort	Directly interconnected using a dedicated load balancer Interconnected using a shared load balancer through a NodePort
Managing container IP addresses	 Separate container CIDR blocks needed Container CIDR blocks divided by node and dynamically added after being allocated 	 Separate container CIDR blocks needed Container CIDR blocks divided by node and statically allocated (the allocated CIDR blocks cannot be changed after a node is created) 	Container CIDR blocks divided from a VPC subnet (You do not need to configure separate container CIDR blocks.)

Dimensio n	Tunnel Network	VPC Network	Cloud Native Network 2.0
Network performan ce	Performance loss due to VXLAN encapsulation	No tunnel encapsulation, and cross-node traffic forwarded through VPC routers (The performance is so good that is comparable to that of the host network, but there is a loss caused by NAT.)	Container network integrated with VPC network, eliminating performance loss
Networkin g scale	A maximum of 2000 nodes are supported.	Suitable for small- and medium-scale networks due to the limitation on VPC route tables. It is recommended that the number of nodes be less than or equal to 1000. Each time a node is added to the cluster, a route is added to the VPC route tables (including the default and custom ones). Evaluate the cluster scale that is limited by the VPC route tables before creating the cluster. For details about route tables, see Constraints .	A maximum of 2000 nodes are supported. In a cloud-native network 2.0 cluster, containers' IP addresses are assigned from VPC CIDR blocks, and the number of containers supported is restricted by these blocks. Evaluate the cluster's scale limitations before creating it.

NOTICE

- 1. The scale of a cluster that uses the VPC network model is limited by the custom routes of the VPC. Therefore, you need to estimate the number of required nodes before creating a cluster.
- 2. By default, VPC routing network supports direct communication between containers and hosts in the same VPC. If a peering connection policy is configured between the VPC and another VPC, the containers can directly communicate with hosts on the peer VPC. In addition, in hybrid networking scenarios such as Direct Connect and VPN, communication between containers and hosts on the peer end can also be achieved with proper planning.

10.3 Enabling Cross-VPC Network Communications Between CCE Clusters

Application Scenarios

Because services in different VPCs cannot communicate with each other, CCE clusters are unable to communicate across VPCs. To resolve this, a VPC peering connection can be established between two VPCs with different CIDR blocks. This allows clusters in one VPC to access clusters or other services in the other VPC.





To enable cross-VPC access, clusters with different network models must communicate with each other across different CIDR blocks. For example, if the local VPC CIDR block of a cluster is 172.16.0.0/16 and the peer VPC CIDR block is 172.17.0.0/16, the routing tables at both ends should be configured as shown in the below table.

Cluster Network	VPC Route Table Configura	ation at Both Ends
Model	Local VPC Route Table of a Cluster	Peer VPC Route Table
Container tunnel network	The peer VPC CIDR block 172.17.0.0/16 must be added to the destination IP address.	The cluster VPC CIDR block 172.16.0.0/16 must be added to the destination IP address.
VPC network	The peer VPC CIDR block 172.17.0.0/16 must be added to the destination IP address.	The cluster VPC CIDR block 172.16.0.0/16 must be added to the destination IP address.
	The container CIDR block of the cluster, for example, 10.0.0.0/16, must be added to the destination IP address.	The container CIDR block of the cluster, for example, 10.0.0.0/16, must be added to the destination IP address.

Cluster Network	VPC Route Table Configura	ation at Both Ends
Model	Local VPC Route Table of a Cluster	Peer VPC Route Table
Cloud Native 2.0 network (for CCE Turbo clusters)	The peer VPC CIDR block 172.17.0.0/16 must be added to the destination IP address.	The cluster VPC CIDR block 172.16.0.0/16 must be added to the destination IP address.

Step 1: Create a VPC Peering Connection

- **Step 1** Log in to the VPC peering connection console.
- **Step 2** In the upper right corner of the page, click **Create VPC Peering Connection**. The **Create VPC Peering Connection** page is displayed.
- **Step 3** Configure the parameters following instructions.

For details about the parameters, see **Table 10-2**.

Figure 10-15 Creating a VPC peering connection

Basic Configuration

Region	
VPC Peering Connection Name	peering-AB
Description (Optional)	0/255 1⁄2
Local VPC Settings	
Local VPC	pc-A v Q
Local VPC CIDR Block 172	2.16.0.0/16
Peer VPC Settings	
Account	My account Another account (?)
Peer Project	ou select My account , the project is filled in by default.
Peer VPC v	pc-B v
Peer VPC CIDR Block 172	2.17.0.0/16

Parameter	Description	Example Value
VPC Peering Connection Name	Mandatory. Enter a name for the VPC peering connection. The name can contain a maximum of 64 characters, including letters, digits, hyphens (-), and underscores (_).	peering-AB
Local VPC	Mandatory. VPC at one end of the VPC peering connection. You can select one from the drop-down list.	vрс-А
Local VPC CIDR Block	CIDR block of the selected local VPC	vpc-A CIDR block: 172.16.0.0/16
Account	 Mandatory. My account: The local and peer VPCs are from the same account. Another account: The local and peer VPCs are from different accounts. 	Current account
Peer Project	The system fills in the corresponding project by default because Account is set to My account . For example, if vpc-A and vpc-B are in account A and region A, the system fills in the correspond project of account A in region A by default.	None
Peer VPC	This parameter is mandatory if Account is set to My account. VPC at the other end of the VPC peering connection. You can select one from the drop-down list.	vpc-В
Peer VPC CIDR Block	CIDR block of the selected peer VPC If the local and peer VPCs have overlapping CIDR blocks, the VPC peering connection may not take effect.	vpc-B CIDR block: 172.17.0.0/16

|--|

Parameter	Description	Example Value
Description	This parameter is optional. Enter the description of the VPC peering connection in the text box as required.	Use peering-AB to connect vpc-A and vpc-B.

Step 4 Click Create Now.

A dialog box for adding routes is displayed.

Step 5 In the displayed dialog box, click Add Now. On the displayed page about the VPC peering connection details, go to Step 2: Add Routes for the VPC Peering Connection to add a route.

----End

Step 2: Add Routes for the VPC Peering Connection

Step 1 In the lower part of the VPC peering connection details page, click **Add Route**.

The **Add Route** dialog box is displayed.

Figure 1	10-16 Adding	routes fo	or the VPC	peering	connection
----------	--------------	-----------	------------	---------	------------

VPC	vpc-A	~	
Route Table	(rtb-vpc-A(Default)	~	Q View Route Table
Destination	172.17.0.0/16	×	
Next Hop	peering-AB(bdd2c6b7-5903-4915	-a63d-e143730	
Description			
Add a route for To enable commu forward and return	or the other VPC nications between VPCs connected by a n routes to the route tables of the VPCs.	0/255 // VPC peering cor earn more	nnection, you need to add
Add a route fo To enable commu forward and return	or the other VPC nications between VPCs connected by a noutes to the route tables of the VPCs.	0/255 // VPC peering cor earn more	nection, you need to add
Add a route for To enable commu forward and return VPC Route Table	or the other VPC nications between VPCs connected by a noutes to the route tables of the VPCs. vpc-B rtb-vpc-B(Default)	0/255 // VPC peering cor earn more	nnection, you need to add
Add a route for To enable commu forward and return VPC Route Table Destination	or the other VPC nications between VPCs connected by a norutes to the route tables of the VPCs. vpc-B rtb-vpc-B(Default) 172.16.0.0/16	0/255 // VPC peering cor Learn more	nection, you need to add
Add a route for To enable commu forward and return VPC Route Table Destination Next Hop	or the other VPC nications between VPCs connected by a notes to the route tables of the VPCs. vpc-B (rtb-vpc-B(Default)) 172.16.0.0/16 peering-AB(bdd2c6b7-5903-4915)	0/255 ×	nection, you need to add
Add a route for To enable commu forward and return VPC Route Table Destination Next Hop Description	or the other VPC nications between VPCs connected by a n routes to the route tables of the VPCs. vpc-B rtb-vpc-B(Default) 172.16.0.0/16 peering-AB(bdd2c6b7-5903-4915	0/255 × VPC peering cor earn more × × -a63d-e143730	nnection, you need to add

Step 2 Add routes to the VPC route tables following instructions.

 Table 10-3 describes the parameters.

Table 10-3 Parameters

Parameter	Description	Example Value
VPC	Select a VPC that is connected by the VPC peering connection.	vpc-A

Parameter	Description	Example Value
Route Table	Select the route table of the VPC. The route will be added to this route table.	rtb-vpc-A (Default route table)
	Each VPC comes with a default route table to control the outbound traffic from the subnets in the VPC. In addition to the default route table, you can create a custom route table and associate it with the subnets in the VPC. Then, the custom route table controls outbound traffic of the subnets.	
	• If there is only the default route table in the drop-down list, select the default route table.	
	• If there are both default and custom route tables in drop-down list, select the route table associated with the subnet connected by the VPC peering connection.	
Destination	IP address in the VPC at the other end of the VPC peering connection. The value can be a VPC CIDR block, subnet CIDR block, or ECS IP address.	vpc-B CIDR block: 172.17.0.0/16
Next Hop	The default value is the current VPC peering connection. You do not need to specify this parameter.	peering-AB
Description	Supplementary information about the route. This parameter is optional. The route description can contain a maximum of 255 characters and cannot contain angle brackets (< or >).	Route from vpc-A to vpc-B
Add a route for the other VPC	If you select this option, you can also add a route for the other VPC connected by the VPC peering connection. To enable communications between VPCs connected by a VPC peering connection, you need to add forward and return routes to the route tables of the VPCs.	Select this option.
VPC	By default, the system selects the other VPC connected by the VPC peering connection. You do not need to specify this parameter.	vpc-В

Parameter	Description	Example Value
Route Table	Select the route table of the VPC. The route will be added to this route table.	rtb-vpc-B (Default route table)
	Each VPC comes with a default route table to control the outbound traffic from the subnets in the VPC. In addition to the default route table, you can also create a custom route table and associate it with the subnets in the VPC. Then, the custom route table controls outbound traffic of the subnets.	
	• If there is only the default route table in the drop-down list, select the default route table.	
	• If there are both default and custom route tables in drop-down list, select the route table associated with the subnet connected by the VPC peering connection.	
Destination	IP address in the VPC at the other end of the VPC peering connection. The value can be a VPC CIDR block, subnet CIDR block, or ECS IP address.	vpc-A CIDR block: 172.16.0.0/16
Next Hop	The default value is the current VPC peering connection. You do not need to specify this parameter.	peering-AB
Description	Supplementary information about the route. This parameter is optional.	Route from vpc-B to vpc-A.
	The route description can contain a maximum of 255 characters and cannot contain angle brackets (< or >).	

NOTICE

If the cluster in the local VPC uses a VPC network, clients within the same VPC can directly access pods in the cluster through the container IP addresses, thanks to the built-in connectivity features of the VPC networks. For details, see **Cross-VPC Cluster Interconnection**. If the client resides in a different VPC, repeat the previous steps to add the cluster container CIDR block to the route tables on both ends as the **destination**. In this example, the container CIDR block is **10.0.0.0/16**.

Step 3 Click OK.

You can check the routes in the route list.

----End

Follow-Up Operations

If a cluster needs to access services in other VPCs, **it is important to verify if those cloud services permit access outside the VPC**. This may involve adding a trustlist or security group to enable access to certain services. In the case of a cluster using the VPC network model, you must allow the container CIDR block to pass the destination ends.

For example, if a cluster using the VPC network model needs to access an ECS in a different VPC, you must allow the VPC CIDR block where the cluster is located and its container CIDR block to pass through the ECS security group. This ensures that nodes and containers in the cluster can access the ECS.

10.4 Implementing Network Communications Between Containers and IDCs Using VPC and Direct Connect

Application Scenarios

With VPC and Direct Connect, the container CIDR block (172.56.0.0/16) and IDC CIDR block (10.1.123.0/24) can communicate with each other in the cluster using the VPC network model.

Figure 10-17 Example network topology



Table 10-4 Address information

Networking	CIDR Block
User's IDC network	10.1.123.0/24
Remote and local gateways (addresses for interconnection)	Huawei Cloud: 10.0.0.1/30 User: 10.0.0.2/30
VPC	192.168.0.0/16
Container CIDR block	172.56.0.0/16

Prerequisites

An IDC is available, and the Direct Connect service has been applied for.

Procedure

Step 1 Create a connection.

1. Log in to the management console, click \bigcirc in the upper left corner, and

select the desired region and project. Click = at the upper left corner and choose **Networking** > **Direct Connect** in the expanded list.

- 2. In the navigation pane on the left of the console, choose **Direct Connect** > **Connections**. On the displayed page, click **Create Connection**.
- 3. Enter the equipment room details and select the Direct Connect location and port based on Table 10-5.

Parameter	Description
Billing Mode	Specifies how you are charged. Currently, only Yearly/ Monthly is supported.
Region	Specifies the region where the connection is deployed. You can change the region in the upper left corner of the console.
Connection Name	Specifies the name of your connection.
Location	Specifies the Direct Connect location where your leased line can be connected to.
Carrier	Specifies the carrier that provides the leased line.
Port Type	Specifies the type of the port used by the connection. There are four types of ports: 1GE, 10GE, 40GE, and 100GE.
Leased Line Bandwidth	Specifies the bandwidth of the connection in the unit of Mbit/s. This is the bandwidth of the leased line you have purchased from the carrier.
Equipment Room Address	Specifies the address of your equipment room. The address must be specific to the floor your equipment room is on, for example, XX Equipment Room, XX Building, No. XX, Huajing Road, Pudong District, Shanghai.
Tag	Identifies the connection. A tag consists of a key and a value. You can add 10 tags to a connection. NOTE If a predefined tag has been created on TMS, you can directly select the corresponding tag key and value. For details about predefined tags, see Predefined Tag Overview .
Description	Provides supplementary information about the connection.

Table 10-5 Parameters required for creating a cloud connection

Parameter	Description
Required Duration	Specifies how long the connection will be used.
Auto-renew	Specifies whether to automatically renew the connection to ensure service continuity.
	It is recommended that you set the auto-renewal period to be the same as the required duration. If the required duration is three months, the system automatically renews the subscription for every three months.
Enterprise Project	Centrally manages cloud resources and members by project.

- 4. Click Confirm Configuration.
- 5. Confirm the order and click **Pay Now**.
- 6. Click **Confirm**.

Step 2 Create a virtual gateway.

 Choose Direct Connect > Virtual Gateways, and click Create Virtual Gateway on the right. Add the VPC CIDR block and the container CIDR block in the VPC network model.

Figure 10-18 Creating a virtual gateway

Create Virtual	Gateway	×
* Name	vgw-8fcd	
★ Enterprise Project	-Select	
★ Attach To	VPC Enterprise Router	
* VPC	vpc-ipv6 0077f842-46dc V Q Create VPC	
★ Local Subnet ⑦	192.168.0.0/16,172.56.0.0/16	
	4	
Tag	It is recommended that you use TMS's predefined tag function to add the same tag to different cloud resources. View predefined tags Q	
	Tag key Tag value	
	You can add 20 more tags.	
Description		
	0/128 2	
	Cancel	ок

Parameter	Description	
Name	Specifies the virtual gateway name.	
Enterprise Project	Centrally manages cloud resources and members by project.	
Attach To	Select VPC .	
VPC	Specifies the VPC you want to access using the connection.	
Local Subnet	Specifies the CIDR blocks of subnets in the VPC to connect to the on-premises network.	
	In this example, the cluster uses the VPC network model. Enter the VPC CIDR block (192.168.0.0/16) and container CIDR block (172.56.0.0/16). For clusters using the container tunnel network and Cloud Native 2.0 Network models, you only need to enter the VPC CIDR block.	
Description	Provides supplementary information about the virtual gateway.	
	The description can contain a maximum of 128 characters.	

Table 10-6 Virtual gateway parameters

2. Click **OK**.

When the virtual gateway status changes **Normal**, the virtual gateway has been created.

Step 3 Create a virtual interface.

- 1. Choose **Direct Connect** > **Virtual Interfaces**, and click **Create Virtual Interface** on the right.
- 2. Configure the parameters based on Table 10-7.

* Virtual Interface Owner	● Current account O Another account O
* Region	CN East-Shanghai1
	Select the region where your VPC resides.
* Name	vif-cfdb
* Virtual Interface Priority	Preferred Standard
	If virtual interfaces are associated with one connection, load is balanced among virtual interfaces with the same priority, while virtual interfaces with different priorities are working in active/standby pairs.
* Connection	✓ Q Create Connection
	Bandwidth: Mbit/s
Gateway	Virtual gateway Global DC gateway
* Virtual Gateway	C Create Virtual Gateway
* VLAN	25 ③
	Enter a value from 0 to 3,999 based on your network plan. A value of 0 indicates that the connection does not use VLAN. In this case, only one virtual interface can be created. VLAN IDs of the devices used in the on-premises data center and on the cloud must be the same.
* Bandwidth (Mbit/s)	100 Enable Rate Limiting Learn more
	Multiple virtual interfaces share the bandwidth of the connection. Select a value based on service traffic. The maximum value is the bandwidth of the connection.
* Enterprise Project	default v Q (2) Create Enterprise Project
Tag	It is recommended that you use TMS's predefined tag function to add the same tag to different cloud resources. View predefined tags Q
	Tag key Tag value
	You can add 20 more tags,

Figure 10-19 Creating a virtual interface

Table 10-7 Parameters required for creating a virtual interface

Parameter	Description	
Region	Specifies the region where the connection is deployed. You can change the region in the upper left corner of the console.	
Name	Specifies the virtual interface name.	
	You can enter 1 to 64 characters.	
Connection	Specifies the connection you use to connect your data center to the cloud.	
Virtual Gateway	Specifies the virtual gateway that the virtual interface connects to.	
VLAN	Specifies the VLAN of the virtual interface.	
	Configure the VLAN if you create a connection on your own.	
	The VLAN for a hosted connection will be allocated by the carrier or partner. You do not need to configure the VLAN.	
Bandwidth	Specifies the bandwidth that can be used by the virtual interface in the unit of Mbit/s. The bandwidth cannot exceed that of the connection.	
Enterprise Project	Centrally manages cloud resources and members by project.	

Parameter	Description	
Local Gateway	Specifies the IP address for connecting to the cloud. In this example, the IP address is 10.0.0.1/30 .	
Remote Gateway	Specifies the IP address for connecting to your on- premises network.	
	The remote gateway must be in the same IP address range as the local gateway. Generally, a subnet with a 30-bit mask is recommended.	
	In this example, the IP address is 10.0.0.2/30 .	
Remote Subnet	Specifies the subnets of your on-premises network. If multiple remote subnets are available, use commas (,) to separate them.	
	In this example, the IP address is 10.1.123.0/24 .	
Routing Mode	Specifies the routing mode. Two options are available, Static and BGP .	
	If there are two or more connections, select BGP routing.	
BGP ASN	Specifies the ASN of the BGP peer.	
	This parameter is mandatory when you select BGP routing.	
BGP MD5 Authentication Key	Specifies the password used to authenticate the BGP peer using MD5.	
	This parameter is mandatory when BGP routing is selected, and the parameter values on both gateways must be the same.	
	The key contains 8 to 255 characters and must contain at least two types of the following characters:	
	 Uppercase letters 	
	- Lowercase letters	
	– Digits	
	- Special characters ~!, .:;"(){}[]/@#\$ %^&*+\ =	
Description	Provides supplementary information about the virtual interface.	

- 3. Click **Create Now**. When the status of the virtual interface changes to **Normal**, the virtual interface has been created.
- 4. Ping the IP address of a server in the VPC from your data center to test network connectivity.

Now your environment can connect to the cloud and access the desired VPC.

NOTE

After creating a virtual interface, configure your devices and security group rules to allow access on and off the cloud.

Step 4 Test the connectivity.

- 1. Run the **traceroute** command to check whether the IDC host can communicate with the container.
 - a. If the route is normal, Direct Connect has a return route.
 - b. If the IDC route to the cloud gateway of Direct Connect is abnormal, check whether the route settings at both ends of Direct Connect are correct.
- 2. If the IP address cannot be tracerouted, try the ping or telnet operation. Before pinging the address, ensure that the ICMP policy has been enabled for the security group if the target is an ECS.

----End

10.5 Enabling a CCE Cluster to Resolve Domain Names on Both On-Premises IDCs and Huawei Cloud

10.5.1 Solution Overview

Background

Microservices are increasingly used to deploy applications. When microservices access each other, they need to resolve domain names.

When you have on-premises IDCs with internal domain names configured, and you have deployed containerized applications on both these IDCs and cloud, you need to enable the containers and nodes in CCE clusters to resolve domain names of both the IDC and cloud.

Suppose you have reconstructed one of your applications using microservices. You run the application management backend in a CCE cluster, deploy the content moderation service in the on-premises IDC, and use the image recognition service on Huawei Cloud. The VPC where CCE resides is connected to the IDC through a private line. Figure 10-20 shows the deployment architecture.

When a user accesses this application, the following interaction is involved between different microservices:

- The CCE cluster uses the Huawei Cloud DNS server, by default, to access the image recognition service.
- The CCE cluster uses the internal DNS server of the IDC to access the content moderation service deployed in the IDC.

In this case, the CCE cluster must be able to use both the Huawei Cloud DNS server and the internal DNS server of the IDC. If the DNS server on the CCE node points to that of the IDC, the domain name of Huawei Cloud cannot be resolved. If the IP address of the IDC internal domain name is added to the **hosts** file, the configuration of the CCE node needs to be updated in real time when the IDC internal service IP changes. This is difficult to implement and may cause the CCE node to be unavailable.

D NOTE

The content moderation and image recognition services are used only as examples.





This section provides a solution for CCE clusters to resolve domain names of both on-premises IDCs and Huawei Cloud.

Solution 1: Using a DNS Endpoint for Cascading Resolution

You can use the VPC endpoint service to create a DNS endpoint cascaded with the IDC DNS server, so that nodes and containers in the CCE cluster can use the IDC DNS server for domain name resolution.

- If the Huawei Cloud domain name needs to be resolved, the request is forwarded to the DNS endpoint, and the Huawei Cloud DNS server is used to resolve the address and return the result.
- If the IDC domain name needs to be resolved, the IDC DNS server directly resolves the address and returns the result.

Figure 10-21 Accessing both the Huawei Cloud domain name and external domain name (for nodes)



For domain name resolution in a container, you can set the DNS policy to **ClusterFirst** when creating a pod. In this way, the domain name resolution requests of the container are directly sent to CoreDNS.

- If a cluster-internal domain name needs to be resolved, CoreDNS directly returns the resolution result.
- If an external domain name needs to be resolved, CoreDNS forwards the request to the IDC DNS server for resolution.

Figure 10-22 Accessing both the Huawei Cloud domain name and external domain name (for containers)



Solution 2: Changing the CoreDNS Configurations

Set the DNS policy to **ClusterFirst** when creating a pod so that the domain name resolution requests of containers are directly sent to CoreDNS.

- If a cluster-internal domain name needs to be resolved, CoreDNS directly returns the resolution result.
- If an external domain name needs to be resolved, CoreDNS forwards the request to the IDC DNS server for resolution.
- If a container accesses a Huawei Cloud internal domain name, the domain name is resolved by the internal DNS server of Huawei Cloud.

Figure 10-23 Domain name resolution in solution 2



Solution Comparison

Solution	Advantage	Disadvantage
Using the DNS endpoint for cascading resolution	External domain names can be resolved for containers and nodes in a CCE cluster.	An external DNS server is required to forward the requests for resolving internal domain names of Huawei Cloud, resulting in performance loss.
Changing the CoreDNS configuration	No external DNS server is required to forward the requests for resolving internal domain names of Huawei Cloud. Therefore, there is no performance loss.	 External domain names cannot be resolved on CCE cluster nodes. The configuration will be lost if CoreDNS is upgraded or rolled back, and you need to reconfigure CoreDNS.

10.5.2 Solution 1: Using a DNS Endpoint for Cascading Resolution

Prerequisites

The VPC where the CCE cluster is deployed has been connected to the on-premises IDC through a private line (Direct Connect) or other channels. The IDC can access the IP addresses in the VPC CIDR block and CCE cluster container CIDR block. For details about how to create a Direct Connect connection, see **Getting Started** with Direct Connect.

Procedure

Step 1 Create a DNS endpoint in the VPC where the CCE cluster is deployed.

- 1. Access the VPC Endpoint page on the network console.
- 2. Click **Buy VPC Endpoint** in the upper right corner.
- 3. Select the DNS service and VPC where the CCE cluster is deployed.

★ Billing Mode	Pay-per-use (?)
* Service Category	Cloud services Find a service by name
* Service List	
	Name
	0
	0
	com4.dns
	5 Total Records: 8 < 1 2
	Currently selected: com.myhuaweicloud.cn-north-4.dns
Private Domain Name	Create a Private Domain Name (?)
* VPC	• C
* Subnet	C View subnets

Figure 10-24 Creating a DNS endpoint

4. Click Next and then Submit.

After the creation is complete, you can view the IP address of the DNS endpoint on its details page.

Figure 10-25 IP address of the DNS endpoint

Summary Access Control Tags ID ID VPC VPC1 VPC Endpoint Service Name ID Private IP Address ID			
ID VPC VPC VPC1 VPC Endpoint Service Name	mmary Access Cont	rol Tags	
ID VPC VPC VPC1 VPC Endpoint Service Name П Private IP Address			
VPC VPC1 VPC Endpoint Service Name ☐ ☐	ID		
VPC Endpoint Service Name	VPC	VPC1	
Private IP Address	VPC Endpoint Service Name		D
	Private IP Address		
Private Domain Name			

Step 2 Configure cascading on the IDC DNS server.

NOTE

The configuration varies depending on the DNS server. The following configurations are used only as example.

BIND (a commonly used DNS server software) is used for the following demonstration.

In this step, you configure the DNS server to forward the requests of resolving Huawei Cloud internal domain names to the DNS endpoint created in the previous step.

For example, in BIND, you can add the lines marked in red to the **/etc/ named.conf** file. **192.168.0.203** is the IP address of the DNS endpoint created in **Step 1**.

```
options {
listen-on port 53 { any; };
listen-on-v6 port 53 { ::1; };
```

```
directory "/var/named";
dump-file "/var/named/data/cache_dump.db";
statistics-file "/var/named/data/named_stats.txt";
memstatistics-file "/var/named/data/named_mem_stats.txt";
recursing-file "/var/named/data/named.recursing";
secroots-file "/var/named/data/named.secroots";
allow-query { any; };
forward first;
forwarders { 192.168.0.203; };
.....
```

Step 3 Change the DNS configuration of the node in the CCE cluster.

You can use either of the following methods to change the settings.

Method 1:

};

Change the settings after the node is created.

- 1. Log in to the worker node of the CCE cluster.
- In the /etc/resolv.conf file, change the value of nameserver to the IP address of the IDC DNS server. # vi /etc/resolv.conf nameserver 10.0.0.190
- 3. Run the following command to lock the **resolv.conf** file to prevent it from being automatically updated by Huawei Cloud.

```
chattr +i /etc/resolv.conf
```

For details about how to configure DNS, see **Configuring DNS**.

Method 2:

Change the DNS settings of the VPC subnet where the CCE cluster resides. In this way, the IP address of the specified DNS server is used in the **/etc/resolv.conf** file for newly created worker nodes.

Before using this method, ensure that the node can use the IDC DNS server to resolve the Huawei Cloud internal domain name. Otherwise, the node cannot be created. You are advised to commission the DNS server before you change the DNS settings of the VPC subnet.

Figure 10-26 Subnet DNS settings

Gateway and DNS Information						
DHCP	Enabled		Gateway			
DNS Server Address	2	0	DHCP Lease Time	365 days 🖉	?	

Step 4 Configure the workload DNS policy.

When creating a workload, you can set **dnsPolicy** to **ClusterFirst** in the YAML file for domain name resolution in containers. This is also the default configuration in Kubernetes. For details about how to configure DNS for workloads, see **DNS Configuration**. apiVersion: v1 kind: Pod metadata:

```
namespace: default
name: dns-example
spec:
containers:
- name: test
image: nginx
dnsPolicy: ClusterFirst
```

----End

Verification

After the configuration is complete, run the **dig** command on the cluster node. The command output shows that the IP address of the server is **10.0.0.190**, which indicates that the domain name is resolved by the IDC DNS server.

dig cce.ap-southeast-1.myhuaweicloud.com

```
; <<>> DiG 9.9.4-61.1.h14.eulerosv2r7 <<>> cce.ap-southeast-1.myhuaweicloud.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 24272
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1
;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
;; QUESTION SECTION:
;cce.ap-southeast-1.myhuaweicloud.com. IN A
;; ANSWER SECTION:
cce.ap-southeast-1.myhuaweicloud.com. 274 IN A 100.125.4.16
;; Query time: 4 msec
;; SERVER: 10.0.0.190#53(10.0.0.190)
```

;; WHEN: Tue Feb 23 19:16:08 CST 2021 ;; MSG SIZE rcvd: 76

Access a domain name of Huawei Cloud from the cluster node. The following output shows that the domain name is resolved to the corresponding IP address.

ping cce.ap-southeast-1.myhuaweicloud.com
PING cce.ap-southeast-1.myhuaweicloud.com (100.125.4.16) 56(84) bytes of data.

Create a pod to access the Huawei Cloud domain name. The following output shows that the domain name can be resolved.

```
# kubectl run -i --tty --image tutum/dnsutils dnsutils --restart=Never --rm /bin/sh If you do not see a command prompt, try pressing Enter.
# ping cce.ap-southeast-1.myhuaweicloud.com
PING cce.ap-southeast-1.myhuaweicloud.com (100.125.4.16) 56(84) bytes of data.
```

10.5.3 Solution 2: Changing the CoreDNS Configurations

Prerequisites

The VPC where the CCE cluster is deployed has been connected to the on-premises IDC through a private line (Direct Connect) or other channels. The IDC can access the IP addresses in the VPC CIDR block and CCE cluster container CIDR block. For details about how to create a Direct Connect connection, see **Getting Started** with Direct Connect.

}

Procedure

CoreDNS configurations are stored in the ConfigMap named **coredns**. You can find this ConfigMap in the kube-system namespace. Run the following command to view the default configurations.

kubectl get configmap coredns -n kube-system -oyaml

kind: ConfigMap apiVersion: v1 metadata: name: coredns namespace: kube-system selfLink: /api/v1/namespaces/kube-system/configmaps/coredns uid: d54ed5df-f4a0-48ec-9bc0-3efc1ac76af0 resourceVersion: '21789515' creationTimestamp: '2021-03-02T09:21:55Z' labels: app: coredns k8s-app: coredns kubernetes.io/cluster-service: 'true' kubernetes.io/name: CoreDNS release: cceaddon-coredns data: Corefile: |-.:5353 { bind {\$POD_IP} cache 30 errors health {\$POD_IP}:8080 kubernetes cluster.local in-addr.arpa ip6.arpa { pods insecure upstream /etc/resolv.conf fallthrough in-addr.arpa ip6.arpa loadbalance round_robin prometheus {\$POD_IP}:9153 forward . /etc/resolv.conf reload

The preceding example shows that all CoreDNS configurations are defined in **Corefile**. By default, resolution requests of any domain name that does not belong to the Kubernetes cluster are directed to the DNS server specified by **forward**. In **forward**. **/etc/resolv.conf**, the first period (.) indicates all domain names, and **/etc/resolv.conf**. indicates the DNS server of the node.

If a specific external domain name needs to be resolved, you can add an extra configuration item. For example, if you want to forward the requests of resolving the domain name **content.internal** to the DNS server whose IP address is 10.0.0.190, perform the following operations to add configurations in **Corefile**.

- **Step 1** Log in to the CCE console and click the cluster name to access the cluster console.
- Step 2 In the navigation pane, choose Add-ons. Then, click Edit under CoreDNS.
- Step 3 Add a stub domain in the Parameters area. A stub domain is a key-value pair. The key is a DNS suffix domain name, and the value is a DNS IP address or a group of DNS IP addresses. In this example, set the key-value pair to content.internal --10.0.0.190.

Parameters

Stub Domain	A domain name server for a custom domain name in key-value pair. The key is a suffix of DNS domain name, and the value is one or more DNS addresses. For example, "acme.local – 1.2.3.4,6.7.8.9" means that DNS requests with the ".acme.local" suffix are forwarded to a DNS server lis 1.2.3.4,6.7.8.9.			
	⊖ content internal 10.0.0.190			
-	Add			

Step 4 Click OK.

Step 5 In the navigation pane, choose **ConfigMaps and Secrets**. In the **kube-system** namespace, view the coredns configuration data to check whether the update is successful.

< cce-test *	Cluster: coe-test Is OCE cluster /	Namespace: kube-system	/ ConfigMaps		🕼 Quick Links	Create ConfigMap Create from YAM	IL.
Cluster Information	ConfigMaps Secrets						
B Resources		View ConfigMon Data		×			
Nodes	Delete	view Configmap Data				Enter a name. Q C	
Workloads	ConfigMap Name	Key	Value		Created	Operation	
Networking	controller-leader-election-h		.:5363 (73 days ago	View YAML Update Delete	
Storage	cluster-autoscaler-priority-e		bind (\$PCO_IP) cache 30		93 days ago	View YAML Update Delete	
ConfigMaps and Secrets	coredns		health (\$P00_1P):8080 kubernetes cluster.local in-addr.arpa ip5.arpa [106 days ago	View YAML Update Delete	
Custom Resources			pods insecure fallthrough in-addr.arpa ip6.arpa				
Namespaces	everest-driver-th-config) loadbalance round_robin		106 days ago	View YAML Update Delete	
Ø OSM	cluster-versions	Corefie	prometheus (\$PCD_IP):9153 forward . /etc/resolv.conf (106 days ago	View YAML Update Delete	
Node Scaling	scheduler-config) maland		106 days ago	View YAML Update Delete	
Workload Scaling	9a37566a.coe.io		ready (\$FCD_IF):8081		106 days ago	View YAML Update Delete	
Add-ons 🔇	kube-root-ca.ort		content. internal:5353 (106 days ago	View YAML Update Delete	
Charts			bind (\$POD_IP) errors				
Cluster Upgrade 🛛 懀	cotension-apiserver-authen		cache 30 forward . 10.0.0.190		106 days ago	View YAML Update Delete	
Logging OBT)				
Container Intelligent Analysis OBT			Close	_			0

Corresponding Corefile content:

```
.:5353 {
  bind {$POD_IP}
  cache 30
  errors
  health {$POD_IP}:8080
  kubernetes cluster.local in-addr.arpa ip6.arpa {
     pods insecure
     fallthrough in-addr.arpa ip6.arpa
  }
  loadbalance round_robin
  prometheus {$POD_IP}:9153
  forward . /etc/resolv.conf {
     policy random
  }
  reload
  ready {$POD_IP}:8081
content.internal:5353 {
  bind {$POD_IP}
  errors
  cache 30
  forward . 10.0.0.190
}
```

For details about other CoreDNS configurations, see **Customizing DNS Service**.

----End

Verification

Create a pod to access the IDC domain name. The following output shows that the domain name can be resolved.

web-terminal-568c6566df-c9jhl:~# kubectl run -i --tty --image tutum/dnsutils dnsutils --restart=Never -rm /bin/sh If you don't see a command prompt, try pressing enter. # ping www.content.internal PING www.content.internal (10.0.1.80) 56(84) bytes of data. 64 bytes from 10.0.1.80: icmp_seq=1 ttl=64 time=1.08 ms 64 bytes from 10.0.1.80: icmp_seq=2 ttl=64 time=0.337 ms

Access a Huawei Cloud domain name. The following output shows that the domain name can be resolved.

ping cce.ap-southeast-1.myhuaweicloud.com PING cce.ap-southeast-1.myhuaweicloud.com (100.125.4.16) 56(84) bytes of data.

Access the IDC domain name on the cluster node. The domain name cannot be pinged, indicating that the CoreDNS configurations do not affect the domain name resolution of the node.

10.6 Implementing Sticky Session Through Load Balancing

Concepts

Sticky sessions ensure continuity and consistency when you access applications. If a load balancer is deployed between a client and backend servers, connections may be forwarded to different servers for processing. Sticky sessions can resolve this issue. After sticky session is enabled, requests from the same client will be continuously distributed to the same backend server through load balancing.

For example, in most online systems that require user identity authentication, a user needs to interact with the server for multiple times to complete a session. These interactions require continuity. If sticky session is not configured, the load balancer may allocate certain requests to different backend servers. Since user identity has not been authenticated on other backend servers, interaction exceptions such as a user login failure may occur.

Therefore, select a proper sticky session type based on the application environment.

OSI Layer	Listener Protocol and Networking	Sticky Session Type	Scenarios Where Sticky Sessions Become Invalid
Layer 4	TCP- or UDP- compliant Services	Source IP address : The source IP address of each request is calculated using the consistent hashing algorithm to obtain a unique hashing key, and all backend servers are numbered. The system allocates the client to a particular server based on the generated key. This allows requests from the same IP address are forwarded to the same backend server.	 Source IP addresses of the clients have changed. Requests from the clients exceed the session stickiness duration.
Layer 7	HTTP- or HTTPS- compliant ingresses	 Load balancer cookie: The load balancer generates a cookie after receiving a request from the client. All subsequent requests with the cookie will be routed to the same backend server. Application cookie: The application deployed on the backend server generates a cookie after receiving the first request from the client. All subsequent requests with the same cookie will be routed to the same backend server. 	 If requests sent by the clients do not contain a cookie, sticky sessions will not take effect. Requests from the clients exceed the session stickiness duration.

Table 10-8 Sticky session types

NOTE

When creating a load balancer, configure sticky sessions by setting **kubernetes.io/elb.lb-algorithm** to **ROUND_ROBIN** or **kubernetes.io/elb.lb-algorithm** to **LEAST_CONNECTIONS**. If you set **kubernetes.io/elb.lb-algorithm** is to **SOURCE_IP**, source IP address-based sticky sessions are supported. In this case, you do not need to configure sticky sessions again.

Layer 4 Sticky Sessions for Services

In Layer 4 mode, source IP address-based sticky sessions can be enabled, where hash routing is performed based on the client IP address.
Enabling Layer 4 Sticky Session in a CCE Standard Cluster

In a CCE standard cluster, to enable source IP address-based sticky session for a Service, ensure the following conditions are met:

- 1. Service Affinity of the Service must be set to Node-level, where the externalTrafficPolicy value of the Service must be Local.
- 2. Anti-affinity has been enabled on the backend applications of the Service to prevent all pods from being deployed on the same node.

Procedure

Step 1 Create an Nginx workload.

```
Set the number of pods to 3 and configure podAntiAffinity.
kind: Deployment
apiVersion: apps/v1
metadata:
 name: nginx
 namespace: default
spec:
 replicas: 3
 selector:
  matchLabels:
   app: nginx
 template:
  metadata:
   labels:
     app: nginx
  spec:
   containers:
     - name: container-0
      image: 'nginx:perl'
      resources:
       limits:
         cpu: 250m
        memory: 512Mi
       requests:
         cpu: 250m
         memory: 512Mi
    imagePullSecrets:
     - name: default-secret
   affinity:
     podAntiAffinity:
                                # Pod anti-affinity
      requiredDuringSchedulingIgnoredDuringExecution:
       - labelSelector:
          matchExpressions:
            - key: app
             operator: In
             values:
              - nginx
         topologyKey: kubernetes.io/hostname
```

Step 2 Create a LoadBalancer Service, for example, using an existing load balancer. The following shows an example YAML file for configuring source IP address-based sticky sessions:

apiVersion: v1 kind: Service metadata: name: svc-example namespace: default annotations: kubernetes.io/elb.class: union kubernetes.io/elb.id: ***** kubernetes.io/elb.id: ***** kubernetes.io/elb.b-algorithm: ROUND_ROBIN # Weighted round robin allocation policy kubernetes.io/elb.session-affinity-mode: SOURCE_IP # Enable source IP address-based sticky session.

```
spec:

selector:

app: nginx

externalTrafficPolicy: Local # Node level Service affinity

ports:

- name: cce-service-0

targetPort: 80

nodePort: 32633

port: 80

protocol: TCP

type: LoadBalancer
```

- **Step 3** Check whether the Layer 4 sticky session function is enabled.
 - 1. Log in to the ELB console, locate the row containing the target load balancer, and click the listener name.
 - 2. Check whether the sticky session function is enabled in the backend server group.

Figure 10-27 Enabling Layer 4 sticky session

C Exercic cost searce in cost searce in cost of the second second cost of the second second cost of the second sec					
Eði					
5e7bba5 6255 4351 bacb-aatar13734433 🗇					
isabled					
inabled ()					
'cluster_id'.1db856136-5a2c-11e1-a2db-0255ac1000cff,'tservice_jdf.'%20465237-db82-4775-88d9- 3855949cf701/'attention'.1Attention! It is auto-generated by CCE service, do not modify!"] 🖉					
Manifold Barbard Same Channe Barbard Samer Group					
CP					
oad balancing					
Healthy Configure					
5e ini 33					

----End

Enabling Layer 4 Sticky Session in a CCE Turbo Cluster

In a CCE Turbo cluster, enabling source IP address-based sticky session for a Service relies on the load balancer type.

- When a dedicated load balancer is used, passthrough networking is allowed between the load balancer and pods, and pods function as the backend server group of the load balancer. Therefore, you do not need to configure Service affinity or application anti-affinity when enabling source IP address-based sticky session for the Service.
- If a shared load balancer is used, sticky session cannot be enabled.

Procedure

• For dedicated load balancers

The following shows an example YAML file for configuring source IP addressbased sticky sessions for a Service that uses an existing load balancer: apiVersion: v1 kind: Service metadata: name: svc-example namespace: default annotations:

```
kubernetes.io/elb.class: performance
  kubernetes.io/elb.id: ****
  kubernetes.io/elb.lb-algorithm: ROUND_ROBIN
                                                    # Weighted round robin allocation policy
  kubernetes.io/elb.session-affinity-mode: SOURCE_IP  # Enable source IP address-based sticky
session.
spec:
 selector:
  app: nginx
                                # In CCE Turbo clusters, Service affinity does not need to be
 externalTrafficPolicy: Cluster
configured if a dedicated load balancer is used.
 ports:
   - name: cce-service-0
   targetPort: 80
   nodePort: 32633
   port: 80
   protocol: TCP
 type: LoadBalancer
```

Verify that the Layer 4 sticky session function is enabled.

- a. Log in to the ELB console, locate the row containing the target load balancer, and click the listener name.
- b. Check whether the sticky session function is enabled in the backend server group.

Figure 10-28 Enabling Layer 4 sticky session

<	C Ekistic Load Balancer / Load balancer (ccs-b-8045327-db02-4775-8698-838004Hic7ft0) / Lissenar (big_TCP_80)						
Su	mmary Monitoring	Tags					
	Basic Information				Edit		
	Name	k8s_TCP_80 🖉	ID	55e7bba5 6255-4351-bacb-aa6a13734433 🗇			
	Frontiend Protocol/Port	TCP/80	Listen to All Ports	Disabled			
	Access Control (3)	All IP addresses Configure	Transfer Client IP Address	Enabled (9)			
	Created	Aug 19, 2024 20 02 35 GMT+08 00	Description	["cluster_id":/bbf5636.5a2c-11ef-a2d6.0255ac1000cf","service_id":?92445327-db62-4775-06d8- 83850949cf70","attention":"AttentionI It is auto-generated by CCE service, do not modify["]. Z			
	Modification Protection (3)	Enabled Configure					
	Advanced Settings						
	Idle Timeout (s)	300					
	Default Backend Server	WewWdd Backend Server Change Backend Server G					
	Backend Server Group	k8s_default_ngrxx-80_TCP-80	Backend Protocol	TCP			
	IP Address Type	Type Source IP address	Forwarding Mode	Load balancing			
	Load Balancing Algorithm	00 mm	Health Check	O Healthy Configure			
	Sticky Session	Enabled					

Layer 7 Sticky Sessions for Ingresses

In Layer 7 mode, sticky sessions can be enabled using HTTP cookies or application cookies.

Enabling Layer 7 Sticky Session in a CCE Standard Cluster

To enable cookie-based sticky session on an ingress, ensure the following conditions are met:

- 1. Service Affinity of the ingress must be set to Node-level, where the externalTrafficPolicy value of the Service must be Local.
- 2. Anti-affinity must be enabled for the ingress workload to prevent all pods from being deployed on the same node.

Procedure

Step 1 Create an Nginx workload.

Set the number of pods to 3 and configure podAntiAffinity.

kind: Deployment apiVersion: apps/v1 metadata: name: nginx namespace: default spec: replicas: 3 selector: matchLabels: app: nginx template: metadata: labels: app: nginx spec: containers: - name: container-0 image: 'nginx:perl' resources: limits: cpu: 250m memory: 512Mi requests: cpu: 250m memory: 512Mi imagePullSecrets: - name: default-secret affinity: podAntiAffinity: # Pod anti-affinity $required {\sf DuringSchedulingIgnoredDuringExecution}:$ - labelSelector: matchExpressions: - key: app operator: In values: - nginx topologyKey: kubernetes.io/hostname

Step 2 Create a Service for the workload. This section uses a NodePort Service as an example.

Configure sticky sessions during the creation of a Service. An ingress can access multiple Services, and each Service can have different sticky sessions. apiVersion: v1 kind: Service metadata: name: nginx namespace: default annotations: kubernetes.io/elb.lb-algorithm: ROUND_ROBIN # Weighted round robin allocation policy kubernetes.io/elb.session-affinity-mode: HTTP_COOKIE # HTTP cookie kubernetes.io/elb.session-affinity-option: '{"persistence_timeout":"1440"}' # Session stickiness duration, in minutes. The value ranges from 1 to 1440. spec: selector: app: nginx ports: - name: cce-service-0 protocol: TCP port: 80 targetPort: 80 nodePort: 32633 # Custom node port type: NodePort externalTrafficPolicy: Local # Node level Service affinity

```
You can also select APP_COOKIE.
```

NOTICE

Only shared load balancers support application cookie-based sticky sessions. For details, see What Are the Relationships Between Load Balancing Algorithms and Sticky Session Types?

apiVersion: v1 kind: Service
metadata:
name: nginx
namespace: default
annotations:
kubernetes io/elb.lb-algorithm: ROUND_ROBIN# Weighted round robin allocation policy
kubernetes io/elb sesion-affinity-mode: APP COOKIE # Select APP COOKIE
kubernetes io/elb session-affinity-ontion: '("ann cookie name"."test"): # Application cookie name
sner
salactor
ann nainy
app. ngmx
- Halle, CC-service-0
port: 80
nodePort: 32633 # Custom node port
type: NodePort
external IratticPolicy: Local # Node level Service attinity

Step 3 Create an ingress and associate it with the Service. The following uses an existing load balancer as an example. For details about how to automatically create a load balancer, see **Creating a LoadBalancer Ingress Using kubectl**.

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
 name: ingress-test
 namespace: default
 annotations:
  kubernetes.io/elb.class: union
  kubernetes.io/elb.port: '80'
  kubernetes.io/elb.id: *****
spec:
 rules:
 - host: 'www.example.com'
  http:
   paths:
    - path: '/'
     backend:
      service:
       name: nginx # Service name
       port:
         number: 80
     property:
      ingress.beta.kubernetes.io/url-match-mode: STARTS_WITH
     pathType: ImplementationSpecific
 ingressClassName: cce
```

Step 4 Verify that the Layer 7 sticky session function is enabled.

- 1. Log in to the ELB console, locate the row containing the target load balancer, and click the listener name.
- 2. Click the **Forwarding Policies** tab, click the backend server group name, and check whether sticky session is enabled for it.

C Estate Load Balancer / Load balancer (cos-b-116+603 cdt27-6533648-602050001%) / Useren Summary Porwarding Policies Monitoring Tags	Backend server group Summary Backend S	k8s_default_nginx-03297-80_HTTP-80 C		
You can click Sort and drag and drop the forwarding policies to adjust their priorities. A smaller number	Basic Information			
All Forwards (Polary Set) C Search by forwards policy saves by artical Forcey: 1 Forwards policy saves bits, artical, right, 5655627E © Iturerry If Polity Petersection /	Name Backend Protocol Forwarding Mode Load Balancing Algorithm Slow Start Removed Protection Modification Protection 🕐	ske_antuut rygen 2029 AD J1179 AD H1TP Laast baardoog Weghted sound rabin Disated Disated Coasted Coasted	ID Backend Server Group Type VPC Sticky Session IP Address Type Description	HINITING-2114-805-6496 32260-1008027 \mathcal{O} Type Load balancer collen Type Load balancer collen Double Double PM4 (altimutor''Ado generated by CCE server, de net motify", "da
Proof / Lowell / Default forwarding policy If HTTP 80	Health Check Health Check Associated Resources Load balancer	Disabled	Listeners	Mb_JITT00

Figure 10-29 Enabling Layer 7 sticky session

```
----End
```

Enabling Layer 7 Sticky Session in a CCE Turbo Cluster

Enable cookie-based sticky session on the ingress.

- When a dedicated load balancer is used, passthrough networking is allowed between the load balancer and pods, and pods function as the backend server group of the load balancer. Therefore, you do not need to configure Service affinity or application anti-affinity when enabling cookie-based sticky session for the ingress.
- If a shared load balancer is used, sticky session cannot be enabled.

Procedure

- For dedicated load balancers
 - a. Create a Service for the workload. In a CCE Turbo cluster, the ingresses that use a dedicated load balancer must interconnect with ClusterIP Services.

Configure sticky sessions during the creation of a Service. An ingress can access multiple Services, and each Service can have different sticky sessions.

```
apiVersion: v1
     kind: Service
     metadata:
      name: nginx
      namespace: default
      annotations:
       kubernetes.io/elb.lb-algorithm: ROUND_ROBIN
                                                        # Weighted round robin allocation policy
       kubernetes.io/elb.session-affinity-mode: HTTP_COOKIE  # HTTP cookie
       kubernetes.io/elb.session-affinity-option: '{"persistence_timeout":"1440"}' # Session
     stickiness duration, in minutes. The value ranges from 1 to 1440.
     spec:
      selector:
       app: nginx
      ports:
        - name: cce-service-0
         protocol: TCP
         port: 80
         targetPort: 80
         nodePort: 0
      type: ClusterIP
b. Create an ingress and associate it with the Service. The following uses an
```

b. Create an ingress and associate it with the Service. The following uses an existing load balancer as an example. For details about how to automatically create a load balancer, see Creating a LoadBalancer Ingress Using kubectl.

apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
name: ingress-test
namespace: default
annotations:
kubernetes.io/elb.class: performance
kubernetes.io/elb.port: '80'
kubernetes.io/elb.id: *****
spec:
rules:
- host: 'www.example.com'
http:
paths:
- path: '/'
backend:
service:
name: nginx # Service name
port:
number: 80
property:
ingress.beta.kubernetes.io/url-match-mode: STARTS_WITH
pathType: ImplementationSpecific
ingressClassName: cce

- c. Verify that the Layer 7 sticky session function is enabled.
 - i. Log in to the ELB console, locate the row containing the target load balancer, and click the listener name.
 - ii. Click the **Forwarding Policies** tab, click the backend server group name, and check whether sticky session is enabled for it.

Figure 10)-30 Enabl	ing Layer	7 sticky	session
-----------	-------------------	-----------	----------	---------

C ExetcLoad Balancer / Load Latencer (cost-6416e406c0174903-6464-062000019) / Leterer Backend server group k8s_default_nginx-402297-80_HTTP-40_C Summary Forwarding Policies Monitoring Taps				
	Summary Backend S	ervers		
You can click Sort and drag and drop the forwarding policies to adjust their priorities. A smaller number	Basic Information			
Add Forwarding Policy Sort	Name	k8s_default_nginx-03297-80_HTTP-80	ID	119#019-c211-4665-9466-33266-1049637
Q. Search by forwarding policy name by default.	Forwarding Mode	Load balancing	VPC	Type Load balancer cookie Time 1440 min
Priority 1 Forwarding policy name k8s_default_nginx_6666cd76 3 Running	Load Balancing Algorithm	Weighted round robin	Sticky Session	Enabled
H	Slow Start	Disabled	IP Address Type	IP _V 4
Path Prefix match /	Removal Protection	Disabled	Description	{"attention":"Auto-generated by CCE service, do not modify!","clus
	inclusion Projection (j)	ENNURY		
Priority * (Lowest) / Default forwarding policy	Health Check			
н	Health Check	Disabled		
HTTP-80	Associated Resources			
	Load balancer	cce-lb-414e4d0d-c027-4553-b548-40b25b300d79	Listeners	k0s_HTTP_00

10.7 Obtaining the Client Source IP Address for a Container

When using containers, clients may communicate with them through multiple proxy servers. However, this can cause issues with transferring the clients' source IP addresses to the containers' services. This section describes how to effectively obtain the client source IP address from a container based on different network solutions provided by CCE clusters.

Description



Figure 10-31 Obtaining the source IP addresses from the containers

The method for obtaining source client IP addresses may vary with the network settings. The following shows some typical network configurations and their corresponding solutions:

- Ingress Layer-7 forwarding: When accessing an application at Layer-7, the client's source IP address is automatically saved in the X-Forwarded-For field of the HTTP header. No additional configurations are needed to obtain the client's source IP address.
- **Service Layer-4 forwarding**: The method and principle for obtaining the source IP addresses will depend on the type of Services being used.
 - LoadBalancer Service: A load balancer is used as the traffic entry. Both shared and dedicated load balancers are supported.
 - For a shared load balancer, you need to enable the function of obtaining client IP addresses on the listeners.
 - By default, the function of obtaining client IP addresses is enabled for a dedicated load balancer listener.
 - NodePort Service: Container ports are mapped to node ports, which are used as the entry for external services. The capability of obtaining client source IP addresses depends on the service affinity.
 - In a cluster-level service affinity for a NodePort Service, traffic is forwarded within the cluster, which means the backend containers of the Service cannot access the client's source IP address.
 - In a node-level service affinity for a NodePort Service, the traffic can directly reach the container without any forwarding. This allows the backend containers of the Service to obtain the source IP address of the client.

D NOTE

If Istio is used, you can obtain the source IP address by referring to How Do I Obtain the Actual Source IP Address of a Client After a Service Is Added into Istio?

Scenarios in Which Source IP Address Can Be Obtained

Due to network model differences, CCE does not allow obtaining source IP addresses in some scenarios, as listed in **Table 10-9**. "-" in the table indicates that this scenario does not exist.

Level-1 Category	Level-2 Category	Load Balancer Type	VPC and Container Tunnel Network Models	Cloud Native 2.0 Network Model (CCE Turbo Clusters)	Reference
Layer-7	ELB	Shared	Supported	Supported	LoadBalan
forwarding (ingress)		Dedicated	Supported	Supported	cer ingress
	Nginx (connectin	Shared	Supported	Not supported	Nginx Ingress
	g to the NGINX Ingress Controller add-on)	Dedicated	Supported	Supported	This function is enabled by default. No other configurati on is required.
Layer-4 forwarding (Service)	LoadBalanc er	Shared	Supported	Not supported (supported by workloads that use hostNetwo rk)	LoadBalan cer
		Dedicated	Supported	Supported	

 Table 10-9
 Scenarios in which source IP addresses can be obtained

Level-1 Category	Level-2 Category	Load Balancer Type	VPC and Container Tunnel Network Models	Cloud Native 2.0 Network Model (CCE Turbo Clusters)	Reference
	NodePort	-	Supported	Not supported (supported by workloads that use hostNetwo rk)	NodePort

LoadBalancer Ingress

For the ELB Ingresses (using HTTP- or HTTPS-compliant), the function of obtaining the source IP addresses of the client is enabled by default. No other operation is required.

A client source IP address is placed in the **X-Forwarded-For** field of the HTTP header by the load balancer. The format is as follows:

X-Forwarded-For: <*client-source-IP-address>*, <*proxy-server-1-IP-address>*, <*proxy-server-2-IP-address>*, ...

The first IP address obtained from the **X-Forwarded-For** field is the client source IP address.

Nginx Ingress

NOTE

In Cloud Native 2.0 networks (for CCE Turbo clusters), **if a shared load balancer is used**, **source IP addresses cannot be obtained when an ingress is associated with the NGINX Ingress Controller add-on**. For details, see **Scenarios in Which Source IP Address Can Be Obtained**. To obtain the source IP address, uninstall the NGINX Ingress Controller add-on and use a dedicated load balancer during reinstallation.

- For an Nginx ingress that uses a dedicated load balancer, transparent transmission of source IP addresses is enabled by default. This means that you can easily obtain the source IP address of the client without any additional configurations.
- For an Nginx ingress that uses a shared load balancer, take the following steps to obtain the source IP address of the client:
- **Step 1** Take the Nginx workload as an example. Before configuring the source IP address, obtain the access logs. **nginx-c99fd67bb-ghv4q** indicates the pod name. kubectl logs nginx-c99fd67bb-ghv4q

Information similar to the following is displayed:

10.0.0.7 - - [17/Aug/2023:01:30:11 +0000] "GET / HTTP/1.1" 200 19 "http://114.114.114.114:9421/"

"Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/115.0.0.0 Safari/537.36 Edg/115.0.1901.203" "**100.125.**.****"

100.125.**.** specifies the CIDR block of the load balancer, indicating that the traffic is forwarded through the load balancer.

Step 2 Enable Transfer Client IP Address. This operation is required only when shared load balancers are used. For dedicated load balancers, source IP addressbased transparent transmission is enabled by default.

Procedure for clusters of v1.23.17-r0, v1.25.12-r0, v1.27.9-r0, v1.28.7-r0, v1.29.3-r0, and later versions

- 1. Log in to the CCE console and click the cluster name to access the cluster console.
- 2. In the navigation pane, choose **Add-ons**. In the right pane, locate **NGINX Ingress Controller** and click **Manage**.
- 3. Click **Edit** under the installed instance and enable the function of obtaining the client IP address in the load balancer configuration.

Load Balancer	Shared v -Select- v Q Create Load Balancer C Ny shared load balancers in VPC vpc-default-flexus where the cluster resides are supported. Constraints
	Obtain Client IP Address Not configured V
	If backend services have been configured when you enable or disable the function of obtaining a client IP address, traffic to the listener will be interrupted. Exercise caution when performing this operation.

4. Click OK.

D NOTE

If backend services have been configured when you enable or disable the function of obtaining a client IP address, traffic will be interrupted. Exercise caution when performing this operation.

Procedure for clusters earlier than v1.23.17-r0, v1.25.12-r0, v1.27.9-r0, v1.28.7-r0, and v1.29.3-r0

- 1. Click in the upper left corner of the management console and select a region and a project.
- 2. Choose Service List > Networking > Elastic Load Balance.
- 3. On the **Elastic Load Balance** page, click the name of the target load balancer.
- 4. Click the **Listeners** tab, locate the row containing the target listener, and click **Edit**. If modification protection exists, disable the protection on the basic information page of the listener and try again.
- 5. Enable Transfer Client IP Address.

Figure 10-32 Enabling the function

Edit	
★ Name	k8s_TCP_88
Frontend Protocol	ТСР
Frontend Port	88
* Access Control	All IP addresses • ⑦
★ Transfer Client IP Address	O If you enable this option, servers cannot be backend servers and clients at the same time

Step 3 Access the workload again and view the new access log.

```
10.0.0.7 - - [17/Aug/2023:02:43:11 +0000] "GET / HTTP/1.1" 304 0 "http://114.114.114.114.9421/"
"Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/115.0.0.0
Safari/537.36 Edg/115.0.1901.203" "124.**.**"
```

The source IP address of the client is obtained.

----End

You can enable WAF for the load balancers used by Nginx Ingress Controllers in clusters, but different WAF modes will affect how Nginx Ingress Controllers obtain the real client IP addresses.

• Load balancer access in WAF cloud CNAME access mode

When using the cloud CNAME access mode, requests go through WAF and are checked for protection before being sent to the load balancer. This means that even if the load balancer has transparent transmission of source IP addresses enabled, the client will receive the back-to-source IP address of WAF. Consequently, the Nginx Ingress Controller is unable to obtain the real client IP address by default. In this case, you can edit the NGINX Ingress Controller add-on and add the following configuration to the add-on parameters:

"enable-real-ip": *"true"*, "use-forwarded-headers": *"true",* "proxy-real-ip-cidr": *<Back-to-source IP address you obtained from WAF>*

• Load balancer access in the cloud WAF mode

This mode is transparent access (non-inline deployment) and supports only dedicated load balancers. In this mode, Nginx Ingress Controllers can obtain the real client IP address by default.

LoadBalancer

For a LoadBalancer Service, different types of clusters obtain source IP addresses in different scenarios. In some scenarios, source IP addresses cannot be obtained currently. For details, see **Scenarios in Which Source IP Address Can Be Obtained**.

- CCE Clusters (using VPC or Tunnel network): Source IP addresses can be obtained when either a shared or dedicated load balancer is used.
- CCE Turbo Clusters (using the Cloud Native Network 2.0): Source IP addresses can be obtained for dedicated load balancers, and for shared load balancers with hostNetwork enabled.

VPC and Container Tunnel Network Models

To enable the function of obtaining the source IP address on the console, take the following steps:

Step 1 When creating a LoadBalancer Service on the CCE console, set **Service Affinity** to **Node-level** instead of **Cluster-level**.

Create Service	Create from YAML						
Service Name	Enter a Service name.						
Service Type	ClusterIP Expose services through the internal IP of the cluster, which can only be accessed within the cluster		LoadBalancer Provide external services through ELB load balancing, high availability, ultra-high performance, stability and security	DNAT Expose cluster node access type services through NAT gateway, support multiple nodes to share and use elastic IP			
	1 It is recommended to select	the load balancing access type for o	ut-of-cluster access				
Service Affinity	rvice Affinity Cluster-level Node-level There is access control on the Service with node-level service affinity enabled. Description that the Service cannot be accessed in the cluster						
Namespace	default						

- **Step 2** Go to the ELB console and enable the function of obtaining the client IP address of the listener corresponding to the load balancer. **Transparent transmission of source IP addresses is enabled for dedicated load balancers by default. You do not need to manually enable this function.**
 - 1. Click in the upper left corner of the management console and select a region and a project.
 - 2. Choose Service List > Networking > Elastic Load Balance.
 - 3. On the **Elastic Load Balance** page, click the name of the target load balancer.
 - 4. Click the **Listeners** tab, locate the row containing the target listener, and click **Edit**. If modification protection exists, disable the protection on the basic information page of the listener and try again.
 - 5. Enable Transfer Client IP Address.

Figure 10-33 Enabling the function

Edit	
* Name	k8s_TCP_88
Frontend Protocol	ТСР
Frontend Port	88
* Access Control	All IP addresses
* Transfer Client IP Address	O If you enable this option, servers cannot be backend servers and clients at the same time.

----End

Cloud Native 2.0 Network Model (CCE Turbo Clusters)

When a LoadBalancer Service associated with a shared load balancer is created:

- For workloads with hostNetwork enabled, you can set **Service Affinity** to **Node-level** to obtain the source IP addresses.
- For other workloads, you cannot set Service Affinity to Node-level, so the source IP addresses cannot be obtained.

Dedicated load balancers are recommended. External access can be directly sent to containers. By default, transparent transmission of source IP addresses is enabled for dedicated load balancers. You do not need to manually enable

Transfer Client IP Address on the ELB console. Instead, you only need to select a dedicated load balancer when creating a LoadBalancer Service on the CCE console.



NodePort

Set the service affinity of a NodePort Service to **Node-level** instead of **Cluster-level**. That is, set **spec.externalTrafficPolicy** of the Service to **Local**.

NOTE

In clusters using Cloud Native 2.0 networks, if NodePort Services are used, only workloads with hostNetwork enabled support node-level service affinity. Therefore, only such workloads can obtain source IP addresses.

Figure 10-34 Selecting the node-level affinity

Create Service	Create from YAML			
Service Name	Enter a Service name.			
Service Type	ClusterIP Expose services through the internal IP of the cluster, which can only be accessed within the cluster	Expose services via IP and static port (NodePort) on each node	LoadBalancer Provide external services through ELB load balancing, high availability, ultra-high performance, stability and security	DNAT Expose cluster node access type services through NAT gateway, support multiple nodes to share and use elastic IP
	 It is recommended to select 	the load balancing access type for o	out-of-cluster access	
Service Affinity	Cluster-level Node	-level		
	There is access control on the Serv	ice with node-level service affinity er	nabled. Description that the Service	e cannot be accessed in the cluster

10.8 Increasing the Listening Queue Length by Configuring Container Kernel Parameters

Application Scenarios

By default, the listening queue (backlog) length of **net.core.somaxconn** is set to **128**. If the number of connection requests surpasses this limit during busy services, new requests will be declined. To avoid this issue, you can adjust the kernel parameter **net.core.somaxconn** to increase the length of the listening queue.

Procedure

Step 1 Modify kubelet configurations.

Method 1: Modifying the kubelet configuration of a node pool (Not supported by the default node pool)

×

- 1. Log in to the CCE console and click the cluster name to access the cluster console.
- 2. Locate the row containing the target node pool and choose **More** > **Manage**.

Figure 10-35 Managing node pool configurations

lode Pools Nodes							Quick Links	Create Node Pool
Export View Events Ope	ration Records	Quota (remaining/total): Cluster nodes	(47/50) CPU(cores) (8/400)					
Q. Search or filter by keyword.								Q
A Node Type : Elastic Cloud Server	Normal (VM) Enterprise Proj	act : default Total number of no	ies (actual/expected): 0 / 0 CPU Usa	ge/Request: / Memory (View Node) Update Scaling) (Auto Scal	ng More A
Specifications	AZ	Status	Actual/Desired Nodes	Number of yearly/mon	On-Demand Nodes	Auto Scaling	0	fanage
c7n.large.2 2 vCPUs 4 GiB	AZ3	 Normal 	0/0	0	0	Close	vi /	scept Node
Default/Deal on the state						~		Copy Synchronize
Total number of nodes (actual/exp	pected): 3/3					(View I	1000	Delete

3. Modify kubelet configuration parameters and add [net.core.somaxconn] to Allowed unsafe sysctls.

Figure 10-36 Modifying kubelet parameters

Manage Configurations (Node Pool				
Event creation QPS limit (event-qps)				
────────────────────────────────────				
Upper Limit for Burst Events (event-burst) ?				
─ 10 +				
Allowed unsafe sysctls (allowed-unsafe-sysctls)				
[net.core.somaxcon				

(Not Recommended) Method 2: Modifying kubelet parameters of a node

- 1. Log in to the target node.
- 2. Edit the **/opt/cloud/cce/kubernetes/kubelet/kubelet** file. In versions earlier than 1.15, the file is **/var/paas/kubernetes/kubelet/kubelet**.

Enable net.core.somaxconn. --allowed-unsafe-sysctls=net.core.somaxconn IrootRetest-565556-30106 ~]1 at t_var_pasz/kubernetes/kubelet/kubelet DREMD_ARGS" --bootstrap-kubeconfig=/var/pasz/kubernetes/kubelet/boot.comf --cert-dir=/var/pasz/kubernetes/kubelet/pki --rotate --certificates=true --hetwork-plugin=cni --cni-conf-dir=/var/pasz/kubernetes/kubelet/kubelet/ --bftDH_ARGS" --bootstrap-kubeconfig=/var/pasz/kubernetes/kubelet/kubeconfig --corfig=/var/pasz/kubernetes/kubelet/kubelet/ --bftDH_ARGS" --bootstrap-kubeconfig=/var/pasz/kubernetes/kubelet/kubeconfig --corfig=/var/pasz/kubernetes/kubelet/kubelet.config.pagz --networkide=192,168,116,149 --allow-privileged=True --v=2 --node-labels="so:.ame=EulerDS_2.0, SF5.os.version=3.10.0+062,14,0.1.h 147.culterov2r7.x86,d.os.architecture=amd6f,failure-domain.beta.kubernetes.io/zon#in=ta_ifailure-domain.beta.kubernetes.io/zon#in=ta_ifailure-domain.beta.kubernetes.io/is-baremetal='false'" --machine_id _file=var/pasa/conf/server.conf --cadvisor-port=4194 --allowed-unsafe-sysctls=net.core.somaxconn"

3. Restart kubelet. systemctl restart kubelet

Check the kubelet status.

systemctl status kubelet

After modifying the kubelet parameters of a node, the configuration will be restored after a cluster upgrade or node reset. Exercise caution during this process and consider modifying the kubelet configuration of the node pool.

Step 2 (Required only for clusters earlier than v1.25) Create a pod security policy.

kube-apiserver enables pod security policies for CCE clusters of versions earlier than v1.25. The configurations take effect only after **net.core.somaxconn** is added to **allowedUnsafeSysctls** in the pod security policy. For details about CCE security policies, see **Configuring a Pod Security Policy**.

The following is an example: apiVersion: policy/v1beta1 kind: PodSecurityPolicy metadata: annotations: seccomp.security.alpha.kubernetes.io/allowedProfileNames: '*' name: sysctl-psp spec: allowedUnsafeSysctls: - net.core.somaxconn allowPrivilegeEscalation: true allowedCapabilities: _ '*' fsGroup: rule: RunAsAny hostIPC: true hostNetwork: true hostPID: true hostPorts: - max: 65535 min: 0 privileged: true runAsGroup: rule: RunAsAny runAsUser: rule: RunAsAny seLinux: rule: RunAsAny supplementalGroups: rule: RunAsAny volumes:

After creating the pod security policy **sysctl-psp**, configure RBAC permission control for it.

The following is an example:

```
kind: ClusterRole
apiVersion: rbac.authorization.k8s.io/v1
metadata:
name: sysctl-psp
rules:
- apiGroups:
- "*"
resources:
- podsecuritypolicies
resourceNames:
- sysctl-psp
verbs:
- use
```

apiVersion: rbac.authorization.k8s.io/v1

kind: ClusterRoleBinding metadata: name: sysctl-psp roleRef: kind: ClusterRole name: sysctl-psp apiGroup: rbac.authorization.k8s.io subjects: - kind: Group name: system:authenticated apiGroup: rbac.authorization.k8s.io

Step 3 Create a workload, configure the kernel parameters, and ensure that the workload is affinity with the node with **net.core.somaxconn** enabled in **Step 1**.

apiVersion: apps/v1 kind: Deployment metadata: annotations: description: " labels: appgroup: " name: test1 namespace: default spec: replicas: 1 selector: matchLabels: app: test1 template: metadata: annotations: metrics.alpha.kubernetes.io/custom-endpoints: '[{"api":"","path":"","port":"","names":""}]' labels: app: test1 spec: containers: - image: 'nginx:1.14-alpine-perl' name: container-0 resources: requests: cpu: 250m memory: 512Mi limits: cpu: 250m memory: 512Mi imagePullSecrets: - name: default-secret securityContext: sysctls: name: net.core.somaxconn value: '3000' affinity: nodeAffinity: requiredDuringSchedulingIgnoredDuringExecution: nodeSelectorTerms: - matchExpressions: - key: kubernetes.io/hostname operator: In values: - 192.168.x.x # Node name.

Step 4 Go to the container and check whether the parameter settings take effect. kubectl exec -it cod name> -- /bin/sh

Run the following command in the container to check whether the configuration takes effect:

sysctl -a |grep somax

	-	-	
user@uw8i8he1ka75nyk-machi	<pre>ine:~\$ kubectl ge</pre>	t pod	
NAME R	READY STATUS	RESTARTS	AGE
test1-7794f95b55-fmsll 1	L/1 Running	0	17s
user@uw8i8he1ka75nyk-machi	ine:~\$ kubectl exe	ec -it test	1-7794f95b55-fmsll /bin/sh
/ # sysctl -a grep somax			
net.core.somaxconn = 3000			
sysctl: error reading key	'net.ipv6.conf.a	ll.stable_s	ecret': I/O error
sysctl: error reading key	'net.ipv6.conf.d	efault.stab	le_secret': I/O error
sysctl: error reading key	'net.ipv6.conf.e	th0.stable_	secret': I/O error
sysctl: error reading key	'net.ipv6.conf.lo	o.stable_se	cret': I/O error

Figure 10-37 Viewing the parameter configuration

----End

10.9 Configuring Passthrough Networking for a LoadBalancer Service

Application Scenarios

kube-proxy, which is responsible for forwarding intra-cluster traffic, adds the IP addresses of load balancers associated with the LoadBalancer Services to nodes' local forwarding rules by default. When a client from within a cluster accesses the IP address of a load balancer, the traffic is directly forwarded to the destination instead of being forwarded by the load balancer.

If node-level affinity is configured for a Service (with **externalTrafficPolicy** set to **Local**), the Service will forward traffic only to pods on the node that run these pods. When a node or pod accesses another pod in the same cluster, if the node where the client runs does not have the corresponding backend pod, the access may fail.

Solution

CCE supports passthrough networking. You can configure the **kubernetes.io**/ **elb.pass-through** annotation for the LoadBalancer Service so that the load balancer forwards the intra-cluster access to the IP address of the load balancer associated with the Service to backend pods.



Figure 10-38 Passthrough networking illustration

• CCE clusters

When a LoadBalancer Service is accessed within the cluster, the access is forwarded to the backend pods using iptables/IPVS by default.

When a LoadBalancer Service (configured with elb.pass-through) is accessed within the cluster, the access is first forwarded to the load balancer, then the nodes, and finally to the backend pods using iptables/IPVS.

• CCE Turbo clusters

When a client accesses a LoadBalancer Service from within the cluster, passthrough is used by default. In this case, the client directly accesses the load balancer private network IP address and then access a container through the load balancer.

Notes and Constraints

- In a CCE standard cluster, after passthrough networking is configured using a dedicated load balancer, the private IP address of the load balancer cannot be accessed from the node where the workload pod resides or other pods on the same node as the workload.
- Passthrough networking is not supported for clusters of v1.15 or earlier.
- In IPVS network mode, the passthrough settings of Services connected to the same load balancer must be the same.
- If node-level (local) service affinity is used, **kubernetes.io/elb.pass-through** is automatically set to **onlyLocal** to enable pass-through.

Procedure

This section describes how to create a Deployment using an Nginx image and create a Service with passthrough networking enabled.

Step 1 Use the kubectl command line tool to connect to the cluster. For details, see Connecting to a Cluster Using kubectl.

Step 2 Use the Nginx image to create a Deployment.

Create an nginx-deployment.yaml file. The file content is as follows:

apiVersion: apps/v1 kind: Deployment metadata: name: nginx spec: replicas: 2 selector: matchLabels: app: nginx template: metadata: labels: app: nginx spec: containers: - image: nginx:latest name: container-0 resources: limits: cpu: 100m memory: 200Mi requests: cpu: 100m memory: 200Mi imagePullSecrets: - name: default-secret

Run the following command to deploy the workload: kubectl create -f nginx-deployment.yaml

Step 3 Create a LoadBalancer Service and set **kubernetes.io/elb.pass-through** to **true**. For details about how to create LoadBalancer Service, see **LoadBalancer**.

The content of the nginx-elb-svc.yaml file is as follows: (In this example, a shared load balancer named **james** is automatically created.) apiVersion: v1 kind: Service metadata: annotations: kubernetes.io/elb.pass-through: "true" kubernetes.io/elb.class: union kubernetes.io/elb.autocreate: '{"type":"public","bandwidth_name":"ccebandwidth","bandwidth_chargemode":"bandwidth","bandwidth_size":5,"bandwidth_sharetype":"PER","eip_ty pe":"5_bgp","name":"james"}' labels: app: nginx name: nginx spec: externalTrafficPolicy: Local ports: - name: service0 port: 80 protocol: TCP targetPort: 80 selector: app: nginx type: LoadBalancer

Step 4 Run the following command to create the Service:

kubectl create -f nginx-elb-svc.yaml

----End

Verification

- **Step 1** Log in to the ELB console and check the load balancer (named **james** in this example) associated with the Service.
- **Step 2** Click the load balancer name and click the **Monitoring** tab.

There is 0 connections to the load balancer.

Summary L	isteners Monitoring Access Logs	Associated Serv	rices Tags						
Dimension Time Range	Lood balancer Listener Select Metric Period Rain data ~ 10 3h 12h 1d	7d	30d 🔠 Auto Refeati					View deta	•• 2
	Concurrent Connections ③	Max Min 0 0	Active Connections Court 12 12 09 06 0 1 1810 1820 1830 1840 1850 190	dax Min 0 0	Inactive Connections ③ Count 1.2 0.9 0.6 0.3 0 1810 1820 1820 1830	Max Min 0 0	New Connections © L2	Max M 0 0	lin O
	Incoming Packets O 12 0 0.6 0 0 0 12.0 1830 1840	Max Min 0 0	Outpoing Packets () Outpoing Packets () Outpoint 12	dax Min 0 0	Inbound Rate © Byes > 1.2	Max Min 0 0	Outboard Rate O Bytes - 1,2 - 0,9 - 0,6 - 0,3 - 0 18:30 18:40	Max M 0 0	in o

Step 3 Log in to an Nginx container in the cluster using kubectl and access the IP address of the load balancer.

- Obtain the Nginx containers in the cluster. kubectl get pod
 Information similar to the following is displayed: NAME READY STATUS RESTARTS AGE nginx-7c4c5cc6b5-vpncx 1/1 Running 0 9m47s nginx-7c4c5cc6b5-xj5wl 1/1 Running 0 9m47s
- 2. Log in to an Nginx container. kubectl exec -it *nginx-7c4c5cc6b5-vpncx* -- /bin/sh
- 3. Access the load balancer IP address.
- **Step 4** Wait for a while and check the monitoring data on the ELB console.

If a new access connection is displayed, the access is forwarded by the load balancer as expected.

----End

10.10 Accessing an External Network from a Pod

10.10.1 Accessing the Internet from a Pod

How to Implement

The method of accessing a public network address from a pod varies depending on the network model of the cluster. For details, see **Table 10-10**.

Implementat ion	Tunnel Network	VPC Network	Cloud Native 2.0 Network
Binding an EIP to the node where the pod resides	Supported	Supported	Not supported
Binding an EIP to the pod	Not supported	Not supported	Supported NOTE Bind an EIP to the pod. The EIP can be a static one. For details, see Configuring a Static EIP for a Pod.
Accessing the Internet through a NAT gateway	Supported	Supported	Supported

Table 10-10 Methods of accessing a public network address from a pod

The following uses a CCE Turbo cluster as an example to describe how to use a NAT gateway to access the Internet. The NAT gateway allows pods in a VPC to access the Internet through SNAT. SNAT translates private IP addresses to public IP addresses using the assigned EIP. This allows the pods in the VPC to share this EIP to access the Internet. For details, see **Figure 10-39**. SNAT allows pods in a VPC to access the Internet directly, even without an EIP. However, they cannot receive traffic from the Internet. The NAT gateway provides efficient support for high-concurrency connections and enables Internet access. It is well-suited for scenarios with a large volume of requests and connections.



Figure 10-39 How an SNAT rule works

Prerequisites

- A CCE cluster is available. For details, see **Buying a CCE Standard/Turbo Cluster**.
- A pod is available in the cluster. For details, see **Creating a Deployment**.

Procedure

To enable a pod to access the Internet, perform the following operations:

- **Step 1** Check the pod CIDR block.
 - 1. Log in to the management console.
 - 2. Click in the upper left corner and choose **Containers** > **Cloud Container Engine**.
 - 3. Click the cluster name to access the cluster console. In the navigation pane, choose **Overview**. In the **Networking Configuration** area, check the pod subnet.

Figure 10-40 Pod subnet

Networking Configuration

Network Model	Cloud Native Network 2.0
VPC	vpc-test 🖸
Subnet	subnet-test (Available/All IPs: 237/251)
Default Pod Subnet	subnet-test (Available/All IPs: 237/251) Add
Default CNI Security Group	cce-test-cce-eni-dg0f0 [
IPv4 Service CIDR Block	10.247.0.0/16
Forwarding	iptables
Default Node Security Group	cce-test-cce-node-dg0f0 📿 Edit

Step 2 Check the access from the pod to the Internet. **Log in to the target pod** and run the following command on the CloudShell page:

curl -I console-intl.huaweicloud.com

If information similar to the following is displayed, the pod cannot access the Internet:

curl: (7) Failed to connect to console-intl.huaweicloud.com port 80: Connection timed out

- Step 3 Assign an EIP. For details, see Assigning an EIP.
 - 1. Click \bigcirc in the upper left corner of the console and select a region.

D NOTE

Set **Region** to the one where the target pod is located.

- 2. Click in the upper left corner and choose **Networking** > **Elastic IP** in the expanded list.
- 3. On the **EIPs** page, click **Buy EIP**.
- 4. Configure parameters following instructions. For details, see Figure 10-41.

Figure 10-41 Buying an EIP

Billing Mode Yearly Monthly Pay per use	
Posice All Market and All All Annuality and	
region v r r r r r r r r r r r r r r r r r r	
Regions are geographic areas isolated from each other. Resources are region-specific and cannot be used across regions through internal network connections. For low network latency and quick resource acc	ss, select the nearest region.
EIP Type Dynamic BGP Static BGP EIP Pool (?)	
 Greater than or equal to 99 95% service availability rate 	
Billed By Bandwidth 👌 For heavy/stable traffic For light/sharply fluctuating traffic For light/sharply fluctuating traffic	
Biled based on total outbound traffic insepactive of usage duration. You can configure maximum bandwidh size, which is used only for limiting data transfer rate. If a pay-per-use EIP is unbound from an instance, the traffic will not be billed but the EIP will be billed to keep it allocated to your account unless it is released. Pficing details	
Bandwidth (Mbits) 5 10 20 50 100 300 Custom The value ranges from 1 to 300 Mbits. 📀	
Bandwidth Name bandwidth: 141d	
DDoS Protection	
DDxS Protection ⓒ Cloud Native Anti-DDxS Basic Provides up to 5 Gbits of DDxS mitigation capacity for free. If the attack to an EIP exceeds 5 Gbits, the EIP will be blocked.	
Eir Delails	
Enterprise Project 🗸 🧿 🔾 Create Enterprise Project [2]	
IPv6 EIP Enable IPv6 Internet access.	
IPv6 EIP is free during the Open Beta Test. After IPv6 EIP is enabled, you need to configure security group rules to allow traffic to and from 198, 19.0.016.	
Tag TINC's predefined tags are recommended for adding the same tag to different cloud resources. Create predefined tags (? Q.	
T rus ieg You can add 20 more tags.	

Step 4 Buy a NAT gateway. For details, see **Buy a Public NAT Gateway**.

- 1. Click \equiv in the upper left corner and choose **Networking** > **NAT Gateway**.
- 2. On the displayed page, click **Buy Public NAT Gateway**.
- 3. Configure parameters as prompted. Set **VPC** to the one used by the cluster and **Subnet** to the pod subnet. For details, see **Figure 10-42**. When using a CCE Turbo cluster, set **Subnet** to the one where the pod is located. When using a CCE standard cluster, set **Subnet** to the one where the node is located.

Basic Configuration	
Region	0 againting v
	Regions are geographic areas lookaled from each other. Resources are region-specific and cannot be used across regions through internal network connections. For low network latency and quick resource access, select the nearest region.
Biling Mode	Yearly Monthly Pay-ser-use
	Billed by the day. Each billing period starts from 08:00:00 and there is a one-day minimum.Learn more [2]
Specifications	Small Medium Large Eritra-targe
	Supports up to 10,000 connections. Learn more 🕐
Name	nd-800
VPC	vpc-test vpc 2 Vecv VPCs 2
Subnet	submet-test (192,168.0.0/ V Q. Create Submet 2 Vew Submets 2
	Available private IP addresses: 237 The selected subnet is for the NAT gateway only. To enable communications over the internet, add rules after the NAT gateway is created.
Enterprise Project	default V O Q. Create Enterprise Project [2]
 Advanced Settings (Optional) 	
SNAT Connection TCD Timoout (c)	1000 SMAT Presenting IDD Tensori (c) 200 SMAT Presenting (CNUT Tensori (c) 40 TCD TIME WAT (c) 5 Description: Tax

Figure 10-42 Buying a NAT gateway

- **Step 5** Configure an SNAT rule and bind the EIP to the subnet. For details, see Add an SNAT Rule.
 - 1. On the page displayed, click the name of the NAT gateway for which you want to add the SNAT rule.
 - 2. On the SNAT Rules tab page, click Add SNAT Rule.
 - 3. Configure parameters following instructions. For details, see Figure 10-43.

NOTE

SNAT rules take effect by CIDR block. When configuring CIDR blocks for different container network models, consider their communication modes. Follow these rules to ensure proper configuration:

- For a CCE standard cluster using a tunnel or VPC network, select the CIDR block where the node is located, which is the block selected during node creation.
- For a CCE Turbo cluster using Cloud Native Network 2.0, select the container CIDR block used during cluster creation.

If there are multiple CIDR blocks, you can create multiple SNAT rules or customize a CIDR block. Make sure that the selected CIDR block includes either the container subnet (for Cloud Native Network 2.0) or the node subnet (for tunnel network and VPC network).

Figure 10-43 Adding an SNAT rule

Add SNAT Rule		×
 If both an EIP and a NA It is not recommended th An SNAT rule cannot sh 	T gateway are configured for a server, data will be forwarded through the EIP. View restrictions hat an SNAT rule and a DNAT rule share the same EIP because there may be service conflicts. are an EIP with a DNAT rule with Port Type set to All ports.	
Public NAT Gateway Name	nat-f8bb	
* Scenario	VPC Direct Connect/Cloud Connect	
* CIDR Block	Existing Custom 3	
	subnet-test(192.168.0.0/24) V Q ③	
* Public IP Address Type	EIP Global EIP	
	You can select 19 more EIPs. ③ View EIP Q Specify filter criteria.	Q
	EIP EIP Type Bandwidth Na Bandwidth(Mb Billing Mode Enterprise Pro	
	60.204.146.52 Dynamic bandwidth-141d 5 Pay-per-use default test1	
	Selected EIPs(1): 60.204.146.52 ×	
	If multiple EIPs are selected for an SNAT rule, the system picks one at random to provide services accessible from the Internet.	
Monitoring	Create alarm rules in Cloud Eye to monitor your SNAT connections.	
Description		
	0/255 1/	
	Cancel	к

Step 6 Check whether the pod can access the Internet. **Log in to the target pod** and run the following command on the CloudShell page:

curl -I console-intl.huaweicloud.com

If information similar to the following is displayed, the pod can access the Internet:

```
HTTP/1.1 301 Moved Permanently
Server: CloudWAF
Date: Mon, 19 Aug 2024 12:43:20 GMT
Content-Type: text/html
Connection: keep-alive
Set-Cookie: HWWAFSESID=b4119798a9b29b3c77; path=/
Set-Cookie: HWWAFSESTIME=1724071396374; path=/
```

----End

10.10.2 Accessing Cloud Services from a Pod in the Same VPC

The method of accessing cloud services from a pod in the same VPC varies depending on the cluster's network model. For details, see **Table 1 Accessing cloud services from a pod in the same VPC**.

Network Model	Description
Tunnel network	Data packets are encapsulated through tunnels in the node network. If the node access permission is limited, you will not be able to access resources from a pod in the same VPC. If you encounter an access failure, verify that the security group of the service you are trying to access allows access from the node where the pod is located.
VPC network	Pod traffic is forwarded through VPC routing. If the container CIDR block is different from the VPC CIDR block of the node, the container cannot directly communicate with other IP addresses in the VPC. To access a service outside the cluster from a pod in the same VPC, you must configure the security group of the accessed service.
Cloud Native Network 2.0	Containers receive IP addresses from VPC CIDR blocks. Container CIDR blocks are part of the VPC subnet where the node is located. Therefore, the containers can directly communicate with other IP addresses in the VPC. If the access failed, check if the security group of the accessed service permits access from the container CIDR block.

Table 10-11 Accessing cloud services from a pod in the sa	me VPC
---	--------

Cloud services that communicate with CCE include ECS, ELB, RDS, DCS, Kafka, RabbitMQ, ModelArts, and DDS. To ensure successful communication, make sure your network is configured correctly and verify that the cloud service you want to access allows external access. For example, accessing DCS Redis requires being trustlisted. If you cannot configure the trustlist on the service console, create a service ticket in the target service. The following describes the operations and precautions for accessing an ECS from a pod and an RDS MySQL DB instance from a pod.

Prerequisites

- A CCE cluster is available. For details, see **Buying a CCE Standard/Turbo Cluster**.
- A pod is available in the cluster. For details, see Creating a Deployment.

Accessing an ECS from a Pod

The following uses a VPC network cluster as an example to describe how to configure security group rules for seamless access to an ECS in the same VPC. For example, the node CIDR block is 192.168.0.0/24, and the container CIDR block is 172.16.0.0/16.

Step 1 Buy an ECS. For details, see Purchasing and Using a Linux ECS (New Edition). The ECS and the cluster are in the same region and VPC, and the IP address of the ECS is 192.168.0.28. **Step 2** Log in to the pod. For details, see **Logging In to a Container**. Try to access the ECS from the pod.

ping 192.168.0.28

- If the ping command is available, execute it. If the following information is displayed, the access from the pod to the ECS failed:
 PING 192.168.0.28 (192.168.0.28): 56 data bytes
 --- 192.168.0.28 ping statistics -- 104 packets transmitted, 0 packets received, 100% packet loss
- If no ping command is available, add it. ping: command not found

The following uses the **Nginx:latest** container as an example to describe how to add a ping command. If the ping command is already available, skip this step.

- a. Ensure that the pod can access the Internet. For details, see Accessing the Internet from a Pod.
- b. Update the local software package index.

apt-get update

c. Install the **iputils-ping** software package, which provides the ping command.

apt-get install iputils-ping

d. Access the ECS again.

ping 192.168.0.28

If the following information is displayed, the ping command has been added:

PING 192.168.0.28 (192.168.0.28): 56 data bytes --- 192.168.0.28 ping statistics ---104 packets transmitted, 0 packets received, 100% packet loss

- **Step 3** Add the container CIDR block of the cluster to the inbound rules of the ECS security group so that all pods in the cluster can access the ECS. If you only want to grant access to the ECS from a specific pod in the cluster, you can add the IP address of that pod to the inbound rules of the ECS security group.
 - 1. On the console homepage, click in the upper left corner. In the expanded list, choose **Compute** > **Elastic Cloud Server** and click the target ECS name.
 - 2. Click the Security Groups tab. In the left pane of the page, click Manage Rule. On the Inbound Rules tab page, you can find that the source addresses include the CIDR block 192.168.0.0/18. Within this block, the node CIDR block 192.168.0.0/24 is included, but the container CIDR block 172.16.0.0/16 is not. If a cluster uses a VPC network model, you need to allow both the node and container CIDR blocks in the security group rules of the ECS outside the cluster to access the ECS from a pod. However, for other network models, you only need to allow the node CIDR block in the security group rules of the ECS.

Figure 10-44 Original inbound rules

Some security group rules will not take	effect for ECSs with certain specifications	a Learn more					×
Add Rule Fast Add Rule	Delete Allow Common Port	B Internet Rules: 7 View Security Group Conf	ganation Examples 🕑				
C, Belect a preperty or enter a keywend.							Q 0
Priority	Action	Type	Protocol & Port	Source	Description	Last Modified	Operation
01	Alter	194	TCP: 30040	192.168.0.0/18		Aug 12, 2024 17:27:35 GMT+00.00	Modify Replicate Delete
01	Allow	IPv4	IOMP : All	192,168,0.0/10		Aug 12, 2024 17:27:24 GMT-06:00	Modify Replicate Delete
1	Allav	Pvi	TCP : 3306	192.155.0.0/10		Aug 12, 2024 17:27:16 GMT-00:00	Modify Replicate Delete
0.1	Allery	PH	TCP : 22	Heat der Vale	Permit default Linux SSH port.	Aug 05, 2024 09:22:19 GMT+00:00	Modify Replicate Delete
01	Allav	194	TCP : 3309	and prove	Permit default Windows remote desilitop	Jul 13, 2022 09:10:05 ONT+08:00	Modily Replicate Delete
0 100	Allow	P4	AI	cetaut (t)		Jul 09, 2022 15:29:48 0MT+08:00	Modify Replicate Delete
. 100	Allav	Pvi	AI	detaut (*)		Jul 09, 2022 15:29:48 GMT+08:00	Modify Replicate Delete
Total Records: 7							10 × (8) >

3. Click Add Rule and enter 172.16.0.0/16 in the Source text box. For details, see Figure 10-45.

Figure 10-45 Adding a rule

Priority ⑦ Action ⑦ Type	Protocol & Port (?)	Source 💿	Description	Operation
1 Allow Y IPv4 Y	Protocols / All 🗸	IP address v	Replicate Delete	Replicate Delete
	1-65535	172.16.0.0/16 ×		

Step 4 Check whether the pod can access the ECS outside of the cluster in the same VPC. On the CloudShell page of the pod, run the following command again:

ping 192.168.0.28

If information similar to the following is displayed, the pod can access the ECS:

PING 192.168.0.28 (192.168.0.28): 56 data bytes			
64 bytes from 192.168.0.28: seq=0 ttl=64 time=1.412 ms			
64 bytes from 192.168.0.28: seq=1 ttl=64 time=1.400 ms			
64 bytes from 192.168.0.28: seq=2 ttl=64 time=1.299 ms			
64 bytes from 192.168.0.28: seq=3 ttl=64 time=1.283 ms			
192.168.0.28 ping statistics			
4 packets transmitted, 4 packets received, 0% packet loss			

```
----End
```

Accessing a Cloud Service (RDS for MySQL as an Example) from a Pod

The following uses a VPC network cluster as an example to describe how to configure security group rules for seamless access to an RDS for MySQL instance in the same VPC. For example, the node CIDR block is 192.168.0.0/24, and the container CIDR block is 172.16.0.0/16.

- Step 1 Buy an RDS for MySQL instance. For details, see Buying a DB Instance and Connecting to It Using a MySQL Client. The DB instance and the cluster are in the same region and VPC, and the IP address of the DB instance is 192.168.10.10.
- **Step 2** Log in to the pod. For details, see **Logging In to a Container**. Try to access the RDS for MySQL instance from the pod.

ping 192.168.10.10

• If the ping command is available, execute it. If the following information is displayed, the access from the pod to the RDS for MySQL instance failed:

PING 192.168.10.10 (192.168.10.10): 56 data bytes --- 192.168.10.10 ping statistics ---104 packets transmitted, 0 packets received, 100% packet loss

 If no ping command is available, add it. ping: command not found

The following uses the **Nginx:latest** container as an example to describe how to add a ping command. If the ping command is already available, skip this step.

- a. Ensure that the pod can access the Internet. For details, see Accessing the Internet from a Pod.
- b. Update the local software package index.

apt-get update

c. Install the **iputils-ping** software package, which provides the ping command.

apt-get install iputils-ping

d. Access the RDS for MySQL DB instance again.

ping 192.168.10.10

If the following information is displayed, the ping command has been added:

PING 192.168.10.10 (192.168.10.10): 56 data bytes --- 192.168.10.10 ping statistics ---104 packets transmitted, 0 packets received, 100% packet loss

- Step 3 Add the container CIDR block of the cluster to the inbound rules of the DB instance security group so that all pods in the cluster can access the RDS for MySQL instance. If you only want to grant access to the RDS for MySQL instance from a specific pod in the cluster, you can add the IP address of that pod to the inbound rules of the DB instance security group.
 - 1. On the console homepage, click in the upper left corner. In the expanded list, choose **Databases** > **Relational Database Service**. On the **Instances** page, click the DB instance name.
 - 2. In the navigation pane, choose **Connectivity & Security**. In the **Security Group Rules** area, click the target security group rule. On the **Inbound Rules** tab page, you can find that the source addresses include the CIDR block 192.168.0.0/18. Within this block, the node CIDR block 192.168.0.0/24 is included, but the container CIDR block 172.16.0.0/16 is not. If a cluster uses a VPC network model, you need to allow both the node and container CIDR blocks in the security group rules of the RDS for MySQL instance outside the cluster to access the DB instance from a pod. However, for other network models, you only need to allow the node CIDR block in the security group rules of the DB instance.

Figure 10-46 Original inbound rules

Some security group rules will not tak	e effect for ECSs with certain specification	a Learn more					×
AND RUE FROMAD RUE	Dolore Allow Common Por	B Inbound Rules: 7 View Security Group Conf	Igeration Examples 🕑				
Q. Select a property or enter a keyword.							08
Priority	Action	Type	Protecol & Pert	Source	Description	Last Modified	Operation
0.1	Alter	IPvi	TCP : 30040	192,168,0,010		Aug 12, 2024 17:27:35 GMT-08:08	Modify Replicate Delete
01	Alav	PH	IOMP : AI	192,108,0.0/10		Aug 12, 2024 17 27:24 GMT+00:00	Modify Replicate Delete
01	Allow	194	TCP: 3306	192, 168, 0, 0/18		Aug 12, 2024 17 27 16 OMT-08 00	Modity Replicate Delete
01	Allow	1914	TCP : 22	Heat der Vale	Permit default Linex SSH port.	Aug 06, 2024 08 22 19 (3MT-08:00	Mostly Replicate Delete
•	Alav	(Pvi	TCP : 3389	and being the second	Permit default Windows remote desistao	Jui 13, 2022 09:10:05 0MT+08:00	Modify Replicate Delete
100	Alav	P4	Al	detaut (*)		Jul 09, 2022 15:29:48 GMT+00:00	Modity Replicate Delete
. 100	Alav	Pvi	A	detaut 💿		Jul 09, 2022 15:29:48 GMT+00:00	Modity Replicate Delete
Total Records: 7							10 v < 8 >

 Click Add Rule and enter 172.16.0.0/16 in the Source text box. For details, see Figure 10-47.

Figure 10-47 Adding a rule

Priority ⑦ Action ⑦	Туре	Protocol & Port 🕜	Source ③	Description	Operation
	Protocols / All	IP address v	Poplicate Dolo	Replicate Delete	
1 Allow V IPv4 V		1-65535	172.16.0.0/16 ×		replicate Delete

Step 4 Check whether the pod can access the RDS for MySQL instance. On the CloudShell page of the pod, run the following command again:

ping 192.168.10.10

If information similar to the following is displayed, the pod can access the DB instance:

PING 192.168.10.10 (192.168.10.10): 56 data bytes 64 bytes from 192.168.10.10: seq=0 ttl=64 time=1.412 ms 64 bytes from 192.168.10.10: seq=1 ttl=64 time=1.400 ms 64 bytes from 192.168.10.10: seq=2 ttl=64 time=1.299 ms 64 bytes from 192.168.10.10: seq=3 ttl=64 time=1.283 ms --- 192.168.10.10 ping statistics ---4 packets transmitted, 4 packets received, 0% packet loss

----End

Troubleshooting a Pod Access Failure

If a pod cannot access the network, rectify the fault by referring to **Table 10-12**. If the fault persists, **submit a service ticket** to contact Huawei Cloud customer service.

Check Item	Possible Fault	Solution
Security group rules of the accessed service	 One of the following issues may be the cause of the failure: The security group's inbound rules prevent access to the node CIDR block or container CIDR block. The security group's inbound rules permit access to the node CIDR block and container CIDR block, but the protocol is incorrectly configured. NOTICE Run the ping command and use ICMP to test network connectivity. Before doing so, enable the ICMP port in the security group rule. 	 For possible cause add a security group rule. For details, see Adding a Security Group Rule. For possible cause change the protocol port in the security group rules. For details, see Modifying a Security Group Rule.
Trustlist	The trustlist for the accessed service does not have the node CIDR block and container CIDR block configured.	Add the container and node CIDR blocks to the trustlist. Find more information in the help document for the relevant service.
Domain name resolution	When accessing an external domain name, the pod uses its cluster's domain name resolution to resolve the destination address and accesses the address based on the pod's network policy. However, sometimes the domain name cannot be resolved, resulting in errors. The most common errors are listed below:	Locate the cause of the DNS exception. For details, see DNS Overview for troubleshooting.
	 Name or service not known Temporary failure in name resolution 	
	Unable to resolve hostname	
	DNS resolution failed	
	 Could not resolve MYHOST (nodename nor servname known), where MYHOST indicates the domain name that cannot be resolved 	

Check Item	Possible Fault	Solution
Network policy (applicable only to tunnel networks)	If you have configured a network policy for both your tunnel network cluster and the namespace where the pod is located, the network policy may prevent the pod from accessing the destination address.	If so, modify the network policy. For details, see Configuring Network Policies to Restrict Pod Access .

10.10.3 Accessing Cloud Services from a Pod in a Different VPC

Pods cannot communicate with each other across VPCs. To resolve this issue, you can use VPC peering to connect two VPCs so that pods in one VPC can access services in the other VPC. The method of setting up cross-VPC connectivity varies depending on the clusters' network types. For details, see **Table 10-13**. In this section, the VPC where the cluster is in is referred to as the cluster VPC, the VPC where the cloud service to be accessed is in is referred to as the destination VPC, and the subnet where the cloud service to be accessed is in is referred to as the destination VPC, and the subnet.

Network Model	Description	Difference
Tunnel network	Data packets are encapsulated through tunnels in the node network. To access cloud services from a pod in a different VPC, ensure that the node subnet can communicate with the destination subnet.	After creating a VPC peering connection between the cluster VPC and the destination VPC, you only need to create a route for the node subnet and the destination subnet.
VPC network	Pod traffic is forwarded through VPC routing. The VPC CIDR block of the cluster cannot overlap with the container CIDR block. When accessing services from a pod in a different VPC, ensure that the node subnet can communicate with the destination subnet and that the container CIDR block can also communicate with the destination subnet.	After creating a VPC peering connection between the cluster VPC and the destination VPC, you need to create a route for the destination subnet, cluster node subnet, and container CIDR block.

Table 10-13 Cross-VPC access for clusters of different network types

Network Model	Description	Difference
Cloud Native Network 2.0	In the Cloud Native 2.0 network model, container IP addresses are directly assigned from the VPC CIDR block. To access cloud services from a pod in a different VPC, ensure that the container subnet can communicate with the destination subnet.	After creating a VPC peering connection between the cluster VPC and the destination VPC, you only need to create a route for the container subnet and the destination subnet.

Cloud services that communicate with CCE include ECS, ELB, RDS, DCS, Kafka, RabbitMQ, ModelArts, and DDS. To ensure successful communication, make sure your network is configured correctly and verify that the cloud service you want to access allows external access. For example, accessing DCS Redis requires being trustlisted. If you cannot configure the trustlist on the service console, create a service ticket in the target service.

This section uses ECS and RDS for MySQL as examples to describe how to implement cross-VPC communication between pods in clusters that use different network models. **Table 10-14** shows the details about the cluster, ECS, and RDS for MySQL network information in the example.

Cloud Service	Container Network Model	Network Information
CCE	Container tunnel	Cluster VPC
	network (CCE standard cluster)	 Name: vpc-demo1
		 CIDR block: 192.168.0.0/18 (primary) and 172.1.0.0/24 (secondary)
		NOTE After a VPC is created, if the primary CIDR block is not enough, you can add secondary CIDR blocks to the VPC. For details, see Adding a Secondary IPv4 CIDR Block to a VPC. After a secondary CIDR block is added to the VPC, you can create a subnet based on the secondary CIDR block. The subnet can be used in CCE.
		 Subnet: 192.168.0.0/24, 192.168.60.0/28, and 172.1.0.0/26
		• Node subnet: 192.168.0.0/24
		• Container CIDR block: 172.18.1.0/24
	VPC network (CCE	Cluster VPC
	standard cluster)	 Name: vpc-demo1
		 CIDR block: 192.168.0.0/18 (primary) and 172.1.0.0/24 (secondary)
		 Subnet: 192.168.0.0/24, 192.168.60.0/28, and 172.1.0.0/26
		• Node subnet: 192.168.0.0/24
		• Container CIDR block: 172.18.1.0/24
	Cloud Native 2.0	Cluster VPC
	network (CCE Turbo	 Name: vpc-demo1
	cluster)	 CIDR block: 192.168.0.0/18 (primary) and 172.1.0.0/24 (secondary)
		 Subnet: 192.168.0.0/24, 192.168.60.0/28, and 172.1.0.0/26
		• Node subnet: 192.168.0.0/24
		• Container subnet: 192.168.60.0/28

Table 10-14 Network information
Cloud Service	Container Network Model	Network Information
ECS	N/A	 Destination VPC Name: vpc-demo2 CIDR block: 10.1.0.0/16 Subnet: 10.1.1.0/24 Subnet where the ECS resides (destination subnet): 10.1.1.0/24 ECS IP address: 10.1.1.24
RDS for MySQL	N/A	 Destination VPC Name: vpc-373896-1 CIDR block: 172.16.0.0/16 Subnet: 172.16.0.0/24 Subnet where RDS for MySQL resides (destination subnet): 172.16.0.0/24 IP address of RDS for MySQL: 172.16.0.167

Prerequisites

- A CCE cluster is available. For details, see **Buying a CCE Standard/Turbo Cluster**.
- A pod is available in the cluster. For details, see **Creating a Deployment**.
- You have an ECS or RDS for MySQL DB instance, which is in the same region as the cluster but in a different VPC. For details about how to purchase an ECS or a DB instance, see Purchasing and Using a Linux ECS (New Edition) and Buying a DB Instance and Connecting to It Using a MySQL Client.

Accessing Cloud Services from a Pod in a Different VPC

This part describes how to access an ECS or an RDS for MySQL DB instance from a pod in a different VPC.

Accessing an ECS from a Pod

The following describes how to access an ECS from pods that run in clusters that use the tunnel network model, VPC network model, and Cloud Native 2.0 network model, respectively. You can select a method based on your cluster types.

Accessing an ECS from a pod in a CCE standard cluster that uses the tunnel network model

1. Create a VPC peering connection between the cluster VPC and the destination VPC.

- a. Switch to the console, click = in the upper left corner, and choose **Networking** > **Virtual Private Cloud** in the expanded list.
- b. In the navigation pane, choose **VPC Peering Connections**. In the upper right corner of the displayed page, click **Create VPC Peering Connection**.
- c. Configure parameters following instructions. For details about the parameters, see **Table 10-15**.

Figure 10-48 Creating a VPC peering connection

Parameter	Description	Example Value
VPC Peering Connection Name	Mandatory. Enter a name for the VPC peering connection. The name can contain a maximum of 64 characters, including letters, digits, hyphens (-), and underscores (_).	peering- demo
Local VPC	Mandatory. Local-end VPC of the peering connection. You can choose one from the drop-down list.	vpc-demo1
Local VPC CIDR Block	CIDR block of the selected local-end VPC.	192.168.0.0/ 18 and 172.1.0.0/24
Account	 Mandatory. My account: The local and peer VPCs are from the same account. Another account: The local and peer VPCs are from different accounts. 	My account
Peer Project	The system fills in the corresponding project by default because Account is set to My account . For example, vpc-demo1 and vpc-demo2 are both under account A in region A. Then, the system fills in the project of account A in region A by default.	None
Peer VPC	This parameter is mandatory if Account is set to My account . VPC at the other end of the peering connection. You can choose one from the drop-down list.	vpc-demo2
Peer VPC CIDR Block	CIDR block of the selected peer VPC. NOTICE If the local and peer VPCs have overlapping CIDR blocks, the VPC peering connection may not take effect.	10.1.0.0/16

 Table 10-15 Parameters for creating a VPC peering connection

- d. After configuring the parameters, click **Create Now**.
- 2. In the displayed VPC Peering Connection Created dialog box, click Add Now and add a route for the node subnet and destination subnet. In the Add Route dialog box, configure the parameters following instructions. For details about the parameters, see Table 10-16.

Add Route				×
* VPC	vpc-demo1	~		
* Route Table	rtb-vpc-demo1(Default)	~	Q View Route Table	
* Destination	10.1.1.0/24	×		
★ Next Hop	peering-demo(6a96ca89-4915-4f76-ac	26-24a66		
Description				
		0/255 1/		

Figure 10-49 Adding a route for the node subnet and destination subnet

Add a route for the other VPC

To enable communications between VPCs connected by a VPC peering connection, you need to add forward and return routes to the route tables of the VPCs.Learn more 📿

* VPC	vpc-demo2	~
* Route Table	rtb-vpc-demo2(Default)	✓ Q View Route Table
* Destination	192.168.0.0/24	×
* Next Hop	peering-demo(6a96ca89-4915-4f76-ac26-24	4a6€
Description		
	0/2	55 %
		Cancel OK

Table 10-16 Parameters for adding a route for the node subnet and destination subnet

Parameter	Description	Example Value
VPC	Select a VPC that is connected by the VPC peering connection.	vpc-demo1

Parameter	Description	Example Value
Route Table	Route Table Select the route table of the VPC. The route will be added to this route table.	
	Each VPC comes with a default route table that manages the flow of outgoing traffic from the subnets in the VPC. In addition to the default route table, you can create a custom route table and associate it with the subnets in the VPC. Then, the custom route table controls outbound traffic of the subnets.	(default route table)
	 If there is only the default route table in the drop-down list, select the default route table. 	
	 If there are both default and custom route tables in drop-down list, select the route table associated with the subnet connected by the VPC peering connection. 	
Destination	IP address in the VPC at the other end of the VPC peering connection. The value can be a VPC CIDR block, subnet CIDR block, or ECS IP address.	10.1.1.0/24
Add a route for the other VPC	If you select this option, you can also add a route for the other VPC connected by the VPC peering connection.	Selected
	To allow VPCs connected through VPC peering to communicate, you must include forward and return routes in the VPCs' route tables.	
VPC	By default, the system selects the other VPC connected by the VPC peering connection. You do not need to specify this parameter.	vpc-demo2

Parameter	Description	Example Value	
Route Table	Select the route table of the VPC. The route will be added to this route table.	rtb-vpc- demo2	
	Each VPC comes with a default route table that manages the flow of outgoing traffic from the subnets in the VPC. In addition to the default route table, you can create a custom route table and associate it with the subnets in the VPC. Then, the custom route table controls outbound traffic of the subnets.	(default route table)	
	• If there is only the default route table in the drop-down list, select the default route table.		
	• If there are both default and custom route tables in drop-down list, select the route table associated with the subnet connected by the VPC peering connection.		
Destination	IP address in the VPC at the other end of the VPC peering connection. The value can be a VPC CIDR block, subnet CIDR block, or ECS IP address.	192.168.0.0/2 4	

3. Log in to the pod and enter the following code on the CloudShell page of the pod again, where *10.1.1.24* indicates the IP address of the ECS to be accessed: (For details about how to log in to a container, see Logging In to a Container.)

, ping 10.1.1.24

NOTE

If the access fails, check whether the traffic from the cluster node subnet is allowed in the inbound rules of the ECS security group. If it is not allowed, you need to add a security group rule and allow the corresponding traffic. For details, see Adding a Security Group Rule.

If a ping command is available and information similar to the following is displayed, cross-VPC access from the pod is successful:
 PING 10.1.1.24 (10.1.1.24): 56 data bytes
 64 bytes from 10.1.1.24: seq=0 ttl=64 time=1.412 ms
 64 bytes from 10.1.1.24: seq=1 ttl=64 time=1.400 ms
 64 bytes from 10.1.1.24: seq=2 ttl=64 time=1.299 ms
 64 bytes from 10.1.1.24: seq=3 ttl=64 time=1.283 ms
 --- 10.1.1.24 ping statistics -- 4 packets transmitted, 4 packets received, 0% packet loss

If no ping command is available, add it.
 ping: command not found

The following uses the **Nginx:latest** container as an example to describe how to add a ping command. If the ping command is already available, skip this step.

- i. Ensure that the pod can access the Internet. For details, see **Accessing the Internet from a Pod**.
- ii. Update the local software package index and install the **iputils-ping** software package that provides the ping command. apt-get update apt-get install iputils-ping
- iii. Access the ECS again. ping 10.1.1.24

If information similar to the following is displayed, the ping command has been added and the cross-VPC access from the pod is successful:

PING 10.1.1.24 (10.1.1.24): 56 data bytes 64 bytes from 10.1.1.24: seq=0 ttl=64 time=1.412 ms 64 bytes from 10.1.1.24: seq=1 ttl=64 time=1.400 ms 64 bytes from 10.1.1.24: seq=2 ttl=64 time=1.299 ms 64 bytes from 10.1.1.24: seq=3 ttl=64 time=1.283 ms --- 10.1.1.24 ping statistics ---4 packets transmitted, 4 packets received, 0% packet loss

Accessing an ECS from a pod in a CCE standard cluster that uses the VPC network model

- 1. Create a VPC peering connection between the cluster VPC and the destination VPC.
 - a. Switch to the console, click = in the upper left corner, and choose **Networking** > **Virtual Private Cloud** in the expanded list.
 - b. In the navigation pane, choose **VPC Peering Connections**. In the upper right corner of the displayed page, click **Create VPC Peering Connection**.
 - c. Configure parameters following instructions. For details about the parameters, see **Table 10-17**.

Figure 10-50 Creating a VPC peering connection

<	Create	VPC	Peering	Connection
---	--------	-----	---------	------------

 A VPC peering connection can connect VPCs from the same account or from different accounts as long as they are in the same region.Learn more ² Creating a VPC Peering Connection with Another VPC in Your Account ² Creating a VPC Peering Connection with a VPC in Another Account ² If you want to connect VPCs in different regions, use Cloud Connect ². 	
Basic Configuration	
Region • CN East-Shanghai1 •	
VPC Peering Connection Name peering-demo	
Description (Optional)	
Local VPC Settings	
Local VPC vpc-demo1 V Q	
Local VPC CIDR Block 192.168.0.0/18, 172.1.0.0/24	
Peer VPC Settings	
Account My account Another account ③	
Peer Project Cn-east-3 V If you select My account, the project is filled in by default.	
Peer VPC vpc-demo2 v	
Peer VPC CIDR Block 10.1.0.0/16	

Table 10-17 Parameters for creating a VPC peering connection

Parameter	Description	Example Value
VPC Peering Connection Name	Mandatory. Enter a name for the VPC peering connection. The name can contain a maximum of 64 characters, including letters, digits, hyphens (-), and underscores (-)	peering- demo
Local VPC	Mandatory. Local-end VPC of the peering connection. You can choose one from the drop-down list.	vpc-demo1

Parameter	Description	Example Value
Local VPC CIDR Block	CIDR block of the selected local-end VPC.	192.168.0.0/ 18 and 172.1.0.0/24
Account	Mandatory.	My account
	• My account : The local and peer VPCs are from the same account.	
	• Another account: The local and peer VPCs are from different accounts.	
Peer Project	The system fills in the corresponding project by default because Account is set to My account .	None
	For example, vpc-demo1 and vpc-demo2 are both under account A in region A. Then, the system fills in the project of account A in region A by default.	
Peer VPC	This parameter is mandatory if Account is set to My account .	vpc-demo2
	VPC at the other end of the peering connection. You can choose one from the drop-down list.	
Peer VPC CIDR Block	CIDR block of the selected peer VPC. NOTICE If the local and peer VPCs have overlapping CIDR blocks, the VPC peering connection may not take effect.	10.1.0.0/16

- d. After configuring the parameters, click **Create Now**.
- 2. In the displayed VPC Peering Connection Created dialog box, click Add Now and add a route for the node subnet and destination subnet. In the Add Route dialog box, configure the parameters following instructions. For details about the parameters, see Table 10-18.

Add Route				\times
* VPC	vpc-demo1	~		
* Route Table	rtb-vpc-demo1(Default)	~	Q View Route Table	
* Destination	10.1.1.0/24	×		
★ Next Hop	peering-demo(6a96ca89-4915-4f76-ac26-24a	a66		
Description				
	0/25	5 //		

Figure 10-51 Adding a route for the node subnet and destination subnet

Add a route for the other VPC

To enable communications between VPCs connected by a VPC peering connection, you need to add forward and return routes to the route tables of the VPCs.Learn more 📿

* VPC	vpc-demo2	~
* Route Table	rtb-vpc-demo2(Default)	✓ Q View Route Table
* Destination	192.168.0.0/24	×
* Next Hop	peering-demo(6a96ca89-4915-4f76-ac26-24	la66
Description		
	0/2	55 //
		Cancel OK

Table 10-18 Parameters for adding a route for the node subnet and destination subnet

Parameter	Description	Example Value
VPC	Select a VPC that is connected by the VPC peering connection.	vpc-demo1

Parameter	Description	Example Value
Route Table	Select the route table of the VPC. The route will be added to this route table.	rtb-vpc- demo1
	Each VPC comes with a default route table that manages the flow of outgoing traffic from the subnets in the VPC. In addition to the default route table, you can create a custom route table and associate it with the subnets in the VPC. Then, the custom route table controls outbound traffic of the subnets.	(default route table)
	 If there is only the default route table in the drop-down list, select the default route table. 	
	 If there are both default and custom route tables in drop-down list, select the route table associated with the subnet connected by the VPC peering connection. 	
Destination	IP address in the VPC at the other end of the VPC peering connection. The value can be a VPC CIDR block, subnet CIDR block, or ECS IP address.	10.1.1.0/24
Add a route for the other VPC	If you select this option, you can also add a route for the other VPC connected by the VPC peering connection.	Selected
	To allow VPCs connected through VPC peering to communicate, you must include forward and return routes in the VPCs' route tables.	
VPC	By default, the system selects the other VPC connected by the VPC peering connection. You do not need to specify this parameter.	vpc-demo2

Parameter	Description	Example Value
Route Table	Select the route table of the VPC. The route will be added to this route table.	rtb-vpc- demo2
	Each VPC comes with a default route table that manages the flow of outgoing traffic from the subnets in the VPC. In addition to the default route table, you can create a custom route table and associate it with the subnets in the VPC. Then, the custom route table controls outbound traffic of the subnets.	(default route table)
	 If there is only the default route table in the drop-down list, select the default route table. 	
	 If there are both default and custom route tables in drop-down list, select the route table associated with the subnet connected by the VPC peering connection. 	
Destination	IP address in the VPC at the other end of the VPC peering connection. The value can be a VPC CIDR block, subnet CIDR block, or ECS IP address.	192.168.0.0/2 4

3. On the current page, click **Add Route** and add a route for the destination VPC and the container CIDR block of the cluster. In the **Add Route** dialog box, set **VPC** to **vpc-demo2** and **Destination** to **172.18.1.0/24**. For details, see Figure 10-52.

Ś

k VPC	vpc-demo2	~	
Route Table	rtb-vpc-demo2(Default)	~	Q View Route Table
Destination	172.18.1.0/24	×	
Next Hop	peering-demo(6a96ca89-4915-4f	76-ac26-24a6t	
Description			
		0/255 1/	
Add a route for th	e other VPC		
To enable communica forward and return rou	tions between VPCs connected by a ites to the route tables of the VPCs.L	VPC peering cor earn more 🖸	nnection, you need to ad

Log in to the pod and enter the following code on the CloudShell page of the pod again, where 10.1.1.24 indicates the IP address of the ECS to be accessed: (For details about how to log in to a container, see Logging In to a Container.)

ping 10.1.1.24

NOTE

If the access fails, check whether the traffic from the cluster node subnet and container CIDR block is allowed in the inbound rules of the ECS security group. If it is not allowed, you need to add a security group rule and allow the corresponding traffic. For details, see Adding a Security Group Rule.

 If a ping command is available and information similar to the following is displayed, cross-VPC access from the pod is successful:

PING 10.1.1.24 (10.1.1.24): 56 data bytes 64 bytes from 10.1.1.24: seq=0 ttl=64 time=1.412 ms 64 bytes from 10.1.1.24: seq=1 ttl=64 time=1.400 ms 64 bytes from 10.1.1.24: seq=2 ttl=64 time=1.299 ms

- 64 bytes from 10.1.1.24: seq=3 ttl=64 time=1.283 ms
- --- 10.1.1.24 ping statistics --

4 packets transmitted, 4 packets received, 0% packet loss

If no ping command is available, add it.
 ping: command not found

The following uses the **Nginx:latest** container as an example to describe how to add a ping command. If the ping command is already available, skip this step.

i. Ensure that the pod can access the Internet. For details, see **Accessing the Internet from a Pod**.

- ii. Update the local software package index and install the **iputils-ping** software package that provides the ping command. apt-get update apt-get install iputils-ping
- iii. Access the ECS again. ping 10.1.1.24

If information similar to the following is displayed, the ping command has been added and the cross-VPC access from the pod is successful:

PING 10.1.1.24 (10.1.1.24): 56 data bytes 64 bytes from 10.1.1.24: seq=0 ttl=64 time=1.412 ms 64 bytes from 10.1.1.24: seq=1 ttl=64 time=1.400 ms 64 bytes from 10.1.1.24: seq=2 ttl=64 time=1.299 ms 64 bytes from 10.1.1.24: seq=3 ttl=64 time=1.283 ms --- 10.1.1.24 ping statistics ---4 packets transmitted, 4 packets received, 0% packet loss

Accessing an ECS from a pod in a CCE Turbo cluster that uses the Cloud Native 2.0 network model

- 1. Create a VPC peering connection between the cluster VPC and the destination VPC.
 - a. Switch to the console, click in the upper left corner, and choose **Networking** > **Virtual Private Cloud** in the expanded list.
 - b. In the navigation pane, choose **VPC Peering Connections**. In the upper right corner of the displayed page, click **Create VPC Peering Connection**.
 - c. Configure parameters following instructions. For details about the parameters, see **Table 10-19**.

Figure 10-53 Creating a VPC peering connection

<	Create	VPC	Peering	Connection
---	--------	-----	---------	------------

 A VPC peering connection can d Creating a VPC Peering Con Creating a VPC Peering Con If you want to connect VPCs in d 	connect VPCs from the same account or from different accounts as long as they are in the same region.Learn more 🕜 inection with Another VPC in Your Account 🕝 inection with a VPC in Another Account 🕜 different regions, use Cloud Connect 🕜.
Basic Configuration	
Region	♥ CN East-Shanghai1 ✓
VPC Peering Connection Name	peering-demo
Description (Optional)	0/255 %
Local VPC Settings	
Local VPC vp	c-demo1 V Q
Local VPC CIDR Block 192.	168.0.0/18, 172.1.0.0/24
Peer VPC Settings	
Account	My account Optimized Another account
Peer Project Cn	east-3 v u select My account, the project is filled in by default.
Peer VPC vp	c-demo2 V
Peer VPC CIDR Block 10.1.	0.0/16

Table 10-19 Parameters for creating a VPC peering connection

Parameter	Description	Example Value
VPC Peering Connection Name	Mandatory. Enter a name for the VPC peering connection. The name can contain a maximum of 64 characters, including letters, digits, hyphens (-), and underscores (-)	peering- demo
Local VPC	Mandatory. Local-end VPC of the peering connection. You can choose one from the drop-down list.	vpc-demo1

Parameter	Description	Example Value
Local VPC CIDR Block	CIDR block of the selected local-end VPC.	192.168.0.0/ 18 and 172.1.0.0/24
Account	Mandatory.	My account
	• My account : The local and peer VPCs are from the same account.	
	• Another account: The local and peer VPCs are from different accounts.	
Peer Project	The system fills in the corresponding project by default because Account is set to My account .	None
	For example, vpc-demo1 and vpc-demo2 are both under account A in region A. Then, the system fills in the project of account A in region A by default.	
Peer VPC	This parameter is mandatory if Account is set to My account .	vpc-demo2
	VPC at the other end of the peering connection. You can choose one from the drop-down list.	
Peer VPC CIDR Block	CIDR block of the selected peer VPC. NOTICE If the local and peer VPCs have overlapping CIDR blocks, the VPC peering connection may not take effect.	10.1.0.0/16

2. In the displayed **VPC Peering Connection Created** dialog box, click **Add Now** and add a route for the container subnet and destination subnet. In the **Add Route** dialog box, configure the parameters following instructions. For details about the parameters, see **Table 10-20**.

 \times



* VPC	vpc-demo1	~	
★ Route Table	rtb-vpc-demo1(Default)	~	Q View Route Table
* Destination	10.1.1.0/24	×	
★ Next Hop	peering-demo(6a96ca89-4915-4f76	-ac26-24a66	
Description			
		0/255 4	

* VPC	vpc-demo2	~
★ Route Table	rtb-vpc-demo2(Default)	✓ Q View Route Table
* Destination	192.168.60.0/28	×
★ Next Hop	peering-demo(6a96ca89-4915-4f76-ac26-2	4a66
Description		
	0/2	255 1/
		Cancel OK

Table 10-20 Parameters for adding a route for the container subnet and destination subnet

Parameter	Description	Example Value
VPC	Select a VPC that is connected by the VPC peering connection.	vpc-demo1

Parameter	Description	Example Value
Route Table	Select the route table of the VPC. The route will be added to this route table.	rtb-vpc- demo1
	Each VPC comes with a default route table that manages the flow of outgoing traffic from the subnets in the VPC. In addition to the default route table, you can create a custom route table and associate it with the subnets in the VPC. Then, the custom route table controls outbound traffic of the subnets.	(default route table)
	 If there is only the default route table in the drop-down list, select the default route table. 	
	 If there are both default and custom route tables in drop-down list, select the route table associated with the subnet connected by the VPC peering connection. 	
Destination	IP address in the VPC at the other end of the VPC peering connection. The value can be a VPC CIDR block, subnet CIDR block, or ECS IP address.	10.1.1.0/24
Add a route for the other VPC	If you select this option, you can also add a route for the other VPC connected by the VPC peering connection.	Selected
	To allow VPCs connected through VPC peering to communicate, you must include forward and return routes in the VPCs' route tables.	
VPC	By default, the system selects the other VPC connected by the VPC peering connection. You do not need to specify this parameter.	vpc-demo2

Parameter	Description	Example Value
Route Table	Select the route table of the VPC. The route will be added to this route table.	rtb-vpc- demo2
	Each VPC comes with a default route table that manages the flow of outgoing traffic from the subnets in the VPC. In addition to the default route table, you can create a custom route table and associate it with the subnets in the VPC. Then, the custom route table controls outbound traffic of the subnets.	(default route table)
	• If there is only the default route table in the drop-down list, select the default route table.	
	• If there are both default and custom route tables in drop-down list, select the route table associated with the subnet connected by the VPC peering connection.	
Destination	IP address in the VPC at the other end of the VPC peering connection. The value can be a VPC CIDR block, subnet CIDR block, or ECS IP address.	192.168.60.0/ 28

3. Log in to the pod and enter the following code on the CloudShell page of the pod again, where *10.1.1.24* indicates the IP address of the ECS to be accessed: (For details about how to log in to a container, see Logging In to a Container.)

, ping 10.1.1.24

NOTE

If the access fails, check whether the traffic from the cluster container subnet is allowed in the inbound rules of the ECS security group. If it is not allowed, you need to add a security group rule and allow the corresponding traffic. For details, see Adding a Security Group Rule.

If a ping command is available and information similar to the following is displayed, cross-VPC access from the pod is successful:
 PING 10.1.1.24 (10.1.1.24): 56 data bytes
 64 bytes from 10.1.1.24: seq=0 ttl=64 time=1.412 ms
 64 bytes from 10.1.1.24: seq=1 ttl=64 time=1.400 ms
 64 bytes from 10.1.1.24: seq=2 ttl=64 time=1.299 ms
 64 bytes from 10.1.1.24: seq=3 ttl=64 time=1.283 ms
 --- 10.1.1.24 ping statistics -- 4 packets transmitted, 4 packets received, 0% packet loss

 If no ping command is available, add it. ping: command not found

The following uses the **Nginx:latest** container as an example to describe how to add a ping command. If the ping command is already available, skip this step.

- i. Ensure that the pod can access the Internet. For details, see **Accessing the Internet from a Pod**.
- ii. Update the local software package index and install the **iputils-ping** software package that provides the ping command. apt-get update apt-get install iputils-ping
- iii. Access the ECS again. ping 10.1.1.24

If information similar to the following is displayed, the ping command has been added and the cross-VPC access from the pod is successful:

PING 10.1.1.24 (10.1.1.24): 56 data bytes 64 bytes from 10.1.1.24: seq=0 ttl=64 time=1.412 ms 64 bytes from 10.1.1.24: seq=1 ttl=64 time=1.400 ms 64 bytes from 10.1.1.24: seq=2 ttl=64 time=1.299 ms 64 bytes from 10.1.1.24: seq=3 ttl=64 time=1.283 ms --- 10.1.1.24 ping statistics ---4 packets transmitted, 4 packets received, 0% packet loss

Accessing a Cloud Service (RDS for MySQL as an Example) from a Pod

The following describes how to access an RDS for MySQL DB instance from pods that run in clusters that use the tunnel network model, VPC network model, and Cloud Native 2.0 network model, respectively. You can select a method based on your cluster types.

Accessing an RDS for MySQL DB instance from a pod in a CCE standard cluster that uses the tunnel network model

- 1. Create a VPC peering connection between the cluster VPC and the destination VPC.
 - a. Switch to the console, click in the upper left corner, and choose **Networking** > **Virtual Private Cloud** in the expanded list.
 - b. In the navigation pane, choose **VPC Peering Connections**. In the upper right corner of the displayed page, click **Create VPC Peering Connection**.
 - c. Configure parameters following instructions. For details about some of the parameters, see **Table 10-21**.

Figure 10-55 Creating a VPC peering connection

< | Create VPC Peering Connection

 A VPC peering connection can connect VPCs from the same account or from different accounts as long as they are in the same region.Learn more ⁽²⁾ Creating a VPC Peering Connection with Another VPC in Your Account ⁽²⁾ Creating a VPC Peering Connection with a VPC in Another Account ⁽²⁾ If you want to connect VPCs in different regions, use Cloud Connect ⁽²⁾. 			
Basic Configuration			
Region	♥ CN East-Shanghai1 ∨		
VPC Peering Connection Na	me peering-b34b		
Description (Optional)	0/255 //		
Local VPC Settings			
Local VPC	vpc-373896-1 V Q		
Local VPC CIDR Block	172.16.0.0/12		
Peer VPC Settings			
Account	My account		
Peer Project	cn-east-3 V		
	If you select My account, the project is filled in by default.		
Peer VPC	vpc-demo1 V		
Peer VPC CIDR Block	192.168.0.0/18, 172.1.0.0/24		

Table 10-21 Parameters for creating a VPC peering connection

Parameter	Description	Example Value
VPC Peering Connection Name	Mandatory. Enter a name for the VPC peering connection.	peering- b34b
	The name can contain a maximum of 64 characters, including letters, digits, hyphens (-), and underscores (_).	
Local VPC	Mandatory. Local-end VPC of the peering connection. You can choose one from the drop-down list.	vpc-373896- 1

Parameter	Description	Example Value
Local VPC CIDR Block	CIDR block of the selected local-end VPC.	172.16.0.0/1 2
Account	Mandatory.	My account
	• My account : The local and peer VPCs are from the same account.	
	• Another account: The local and peer VPCs are from different accounts.	
Peer Project	The system fills in the corresponding project by default because Account is set to My account .	None
	For example, vpc-demo1 and vpc-demo2 are both under account A in region A. Then, the system fills in the project of account A in region A by default.	
Peer VPC	This parameter is mandatory if Account is set to My account .	vpc-demo1
	VPC at the other end of the peering connection. You can choose one from the drop-down list.	
Peer VPC CIDR Block	CIDR block of the selected peer VPC. NOTICE If the local and peer VPCs have overlapping CIDR blocks, the VPC peering connection may not take effect.	192.168.0.0/ 18 and 172.1.0.0/24

- d. After configuring the parameters, click **Create Now**.
- 2. In the displayed **VPC Peering Connection Created** dialog box, click **Add Now** and add a route for the node subnet and destination subnet. In the **Add Route** dialog box, configure the parameters following instructions. For details about the parameters, see **Table 10-22**.

×

Add Route			
* VPC	vpc-373896-1	~	
★ Route Table	rtb-d43b	~	Q View Route Table
★ Destination	192.168.0.0/24	×	
★ Next Hop	peering-d34b(5ae9f0d3-43bb-4950-968f-fec	c9b4	
Description			
	0/2	255 4	

Figure 10-56 Adding a route for the node subnet and destination subnet

Add a route for the other VPC

To enable communications between VPCs connected by a VPC peering connection, you need to add forward and return routes to the route tables of the VPCs.Learn more 🕐

* VPC	vpc-demo1 ~	
★ Route Table	rtb-vpc-demo1(Default)	Q View Route Table
* Destination	172.16.0.0/24 ×	
★ Next Hop	peering-d34b(5ae9f0d3-43bb-4950-968f-fec9b-	4
Description		
	0/255	11
		Cancel OK

Table 10-22 Parameters for adding a route for the node subnet and destination subnet

Parameter	Description	Example Value
VPC	Select a VPC that is connected by the VPC peering connection.	vpc-373896- 1

Parameter	Description	Example Value
Route Table	 Select the route table of the VPC. The route will be added to this route table. Each VPC comes with a default route table that manages the flow of outgoing traffic from the subnets in the VPC. In addition to the default route table, you can create a custom route table and associate it with the subnets in the VPC. Then, the custom route table controls outbound traffic of the subnets. If there is only the default route table in the drop-down list, select the default route table. If there are both default and custom route tables in drop-down list, select the route table associated with the subnet 	rtb-d43b (custom route table) NOTICE The custom route table must be associated with the subnet connected by the VPC peering connection.
	connected by the VPC peering connection.	
Destination	IP address in the VPC at the other end of the VPC peering connection. The value can be a VPC CIDR block, subnet CIDR block, or ECS IP address.	192.168.0.0/2 4
Add a route for the other VPC	If you select this option, you can also add a route for the other VPC connected by the VPC peering connection. To allow VPCs connected through VPC peering to communicate, you must include forward and return routes in the VPCs' route tables.	Selected
VPC	By default, the system selects the other VPC connected by the VPC peering connection. You do not need to specify this parameter.	vpc-demo1

Parameter	Description	Example Value
Route Table	Select the route table of the VPC. The route will be added to this route table.	rtb-vpc- demo1 (default route table)
	Each VPC comes with a default route table that manages the flow of outgoing traffic from the subnets in the VPC. In addition to the default route table, you can create a custom route table and associate it with the subnets in the VPC. Then, the custom route table controls outbound traffic of the subnets.	
	• If there is only the default route table in the drop-down list, select the default route table.	
	• If there are both default and custom route tables in drop-down list, select the route table associated with the subnet connected by the VPC peering connection.	
Destination	IP address in the VPC at the other end of the VPC peering connection. The value can be a VPC CIDR block, subnet CIDR block, or ECS IP address.	172.16.0.0/24

3. Log in to the pod and enter the following code on the CloudShell page of the pod again, where *172.16.0.167* indicates the IP address of the RDS for MySQL DB instance to be accessed: (For details about how to log in to a container, see Logging In to a Container.) ping *172.16.0.167*

NOTE

If the access fails, check whether the traffic from the cluster node subnet is allowed in the inbound rules of the DB instance security group. If it is not allowed, you need to add a security group rule and allow the corresponding traffic. For details, see Adding a Security Group Rule.

If a ping command is available and information similar to the following is displayed, cross-VPC access from the pod is successful:
 PING 172.16.0.167 (172.16.0.167) 56(84) bytes of data.
 64 bytes from 172.16.0.167: icmp_seq=1 ttl=63 time=0.516 ms
 64 bytes from 172.16.0.167: icmp_seq=2 ttl=63 time=0.418 ms
 64 bytes from 172.16.0.167: icmp_seq=3 ttl=63 time=0.376 ms
 --- 172.16.0.167 ping statistics -- 3 packets transmitted, 3 received, 0% packet loss, time 1001ms

If no ping command is available, add it.
 ping: command not found

The following uses the **Nginx:latest** container as an example to describe how to add a ping command. If the ping command is already available, skip this step.

i. Ensure that the pod can access the Internet. For details, see **Accessing the Internet from a Pod**.

- ii. Update the local software package index and install the **iputils-ping** software package that provides the ping command. apt-get update apt-get install iputils-ping
- iii. Access the RDS for MySQL DB instance again. ping 172.16.0.167

If information similar to the following is displayed, the ping command has been added and the cross-VPC access from the pod is successful:

PING 172.16.0.167 (172.16.0.167) 56(84) bytes of data. 64 bytes from 172.16.0.167: icmp_seq=1 ttl=63 time=0.516 ms 64 bytes from 172.16.0.167: icmp_seq=2 ttl=63 time=0.418 ms 64 bytes from 172.16.0.167: icmp_seq=3 ttl=63 time=0.376 ms --- 172.16.0.167 ping statistics ---3 packets transmitted, 3 received, 0% packet loss, time 1001ms

Accessing an RDS for MySQL DB instance from a pod in a CCE standard cluster that uses the VPC network model

- 1. Create a VPC peering connection between the cluster VPC and the destination VPC.
 - a. Switch to the console, click in the upper left corner, and choose **Networking** > **Virtual Private Cloud** in the expanded list.
 - b. In the navigation pane, choose **VPC Peering Connections**. In the upper right corner of the displayed page, click **Create VPC Peering Connection**.
 - c. Configure parameters following instructions. For details about some of the parameters, see **Table 10-23**.

Figure 10-57 Creating a VPC peering connection

< | Create VPC Peering Connection

 A VPC peering connection can connect VPCs from the same account or from different accounts as long as they are in the same region.Learn more ⁽²⁾ Creating a VPC Peering Connection with Another VPC in Your Account ⁽²⁾ Creating a VPC Peering Connection with a VPC in Another Account ⁽²⁾ If you want to connect VPCs in different regions, use Cloud Connect ⁽²⁾. 			
Basic Configuration			
Region	♥ CN East-Shanghai1 ✓		
VPC Peering Connection N	ame peering-b34b		
Description (Optional)	0/255 1/2		
Local VPC Settings			
Local VPC	vpc-373896-1 V Q		
Local VPC CIDR Block	172.16.0.0/12		
Peer VPC Settings			
Account	My account Another account ①		
Peer Project	cn-east-3 V		
	If you select My account, the project is filled in by default.		
Peer VPC	vpc-demo1 V		
Peer VPC CIDR Block	192.168.0.0/18, 172.1.0.0/24		

Table 10-23 Parameters for creating a VPC peering connection

Parameter	Description	Example Value
VPC Peering Connection Name	Mandatory. Enter a name for the VPC peering connection.	peering- b34b
	The name can contain a maximum of 64 characters, including letters, digits, hyphens (-), and underscores (_).	
Local VPC	Mandatory. Local-end VPC of the peering connection. You can choose one from the drop-down list.	vpc-373896- 1

Parameter	Description	Example Value
Local VPC CIDR Block	CIDR block of the selected local-end VPC.	172.16.0.0/1 2
Account	Mandatory.	My account
	• My account : The local and peer VPCs are from the same account.	
	• Another account : The local and peer VPCs are from different accounts.	
Peer Project	The system fills in the corresponding project by default because Account is set to My account .	None
	For example, vpc-demo1 and vpc-demo2 are both under account A in region A. Then, the system fills in the project of account A in region A by default.	
Peer VPC	This parameter is mandatory if Account is set to My account .	vpc-demo1
	VPC at the other end of the peering connection. You can choose one from the drop-down list.	
Peer VPC CIDR Block	CIDR block of the selected peer VPC. NOTICE If the local and peer VPCs have overlapping CIDR blocks, the VPC peering connection may not take effect.	192.168.0.0/ 18 and 172.1.0.0/24

- d. After configuring the parameters, click **Create Now**.
- 2. In the displayed **VPC Peering Connection Created** dialog box, click **Add Now** and add a route for the node subnet and destination subnet. In the **Add Route** dialog box, configure the parameters following instructions. For details about the parameters, see **Table 10-24**.

×

Add Route			
* VPC	vpc-373896-1	~	
★ Route Table	rtb-d43b	~	Q View Route Table
★ Destination	192.168.0.0/24	×	
★ Next Hop	peering-d34b(5ae9f0d3-43bb-4950-968f-fec	c9b4	
Description			
	0/2	255 4	

Figure 10-58 Adding a route for the node subnet and destination subnet

Add a route for the other VPC

To enable communications between VPCs connected by a VPC peering connection, you need to add forward and return routes to the route tables of the VPCs.Learn more 🕐

* VPC	vpc-demo1	~
★ Route Table	rtb-vpc-demo1(Default)	View Route Table
* Destination	172.16.0.0/24	×
★ Next Hop	peering-d34b(5ae9f0d3-43bb-4950-968f-fec9	b4
Description		
	0/25	5 //
	(Cancel OK

Table 10-24 Parameters for adding a route for the node subnet and destination subnet

Parameter	Description	Example Value
VPC	Select a VPC that is connected by the VPC peering connection.	vpc-373896- 1

Parameter	Description	Example Value
Route Table	 Select the route table of the VPC. The route will be added to this route table. Each VPC comes with a default route table that manages the flow of outgoing traffic from the subnets in the VPC. In addition to the default route table, you can create a custom route table and associate it with the subnets in the VPC. Then, the custom route table controls outbound traffic of the subnets. If there is only the default route table in the drop-down list, select the default route table. If there are both default and custom route table associated with the subnet connected by the VPC peering 	rtb-d43b (custom route table) NOTICE The custom route table must be associated with the subnet connected by the VPC peering connection.
Destination	IP address in the VPC at the other end of the VPC peering connection. The value can be a VPC CIDR block, subnet CIDR block, or ECS IP address.	192.168.0.0/2 4
Add a route for the other VPC	If you select this option, you can also add a route for the other VPC connected by the VPC peering connection. To allow VPCs connected through VPC peering to communicate, you must include forward and return routes in the VPCs' route tables.	Selected
VPC	By default, the system selects the other VPC connected by the VPC peering connection. You do not need to specify this parameter.	vpc-demo1

Parameter	Description	Example Value	
Route Table	Select the route table of the VPC. The route will be added to this route table.	rtb-vpc- demo1 (default route table)	
	Each VPC comes with a default route table that manages the flow of outgoing traffic from the subnets in the VPC. In addition to the default route table, you can create a custom route table and associate it with the subnets in the VPC. Then, the custom route table controls outbound traffic of the subnets.		
	 If there is only the default route table in the drop-down list, select the default route table. 		
	 If there are both default and custom route tables in drop-down list, select the route table associated with the subnet connected by the VPC peering connection. 		
Destination	IP address in the VPC at the other end of the VPC peering connection. The value can be a VPC CIDR block, subnet CIDR block, or ECS IP address.	172.16.0.0/24	

3. On the current page, click **Add Route**. In the displayed dialog box, set **VPC** to **vpc-373896-1** and **Destination** to **172.18.1.0/24** and add a route for the destination VPC and the container CIDR block of the cluster. For details, see **Figure 10-59**.

Figure 10-	59 Adding	a route	to the	container	CIDR block
------------	-----------	---------	--------	-----------	------------

* VPC	vpc-373896-1	~	
★ Route Table	rtb-d43b	~	View Route Table
* Destination	172.18.1.0/24	×	
★ Next Hop	peering-d34b(5ae9f0d3-43b	b-4950-968f-fec9b4	
Description			
		0/255 4	
Add a route fo	r the other VPC		
To enable commun	ications between VPCs connected	I by a VPC peering cor	nnection, you need to add

4. Log in to the pod and enter the following code on the CloudShell page of the pod again, where *172.16.0.167* indicates the IP address of the RDS for MySQL DB instance to be accessed: (For details about how to log in to a container, see Logging In to a Container.) ping *172.16.0.167*

NOTE

If the access fails, check whether the traffic from the cluster node subnet and container CIDR block is allowed in the inbound rules of the DB instance security group. If it is not allowed, you need to add a security group rule and allow the corresponding traffic. For details, see Adding a Security Group Rule.

 If a ping command is available and information similar to the following is displayed, cross-VPC access from the pod is successful:
 PING 172.16.0.167 (172.16.0.167) 56(84) bytes of data.

64 bytes from 172.16.0.167: icmp_seq=1 ttl=63 time=0.516 ms 64 bytes from 172.16.0.167: icmp_seq=2 ttl=63 time=0.418 ms

64 bytes from 172.16.0.167: icmp_seq=2 ttl=63 time=0.476 ms

--- 172.16.0.167 ping statistics ---

3 packets transmitted, 3 received, 0% packet loss, time 1001ms

 If no ping command is available, add it. ping: command not found

The following uses the **Nginx:latest** container as an example to describe how to add a ping command. If the ping command is already available, skip this step.

- i. Ensure that the pod can access the Internet. For details, see **Accessing the Internet from a Pod**.
- ii. Update the local software package index and install the **iputils-ping** software package that provides the ping command.

apt-get update apt-get install iputils-ping

iii. Access the RDS for MySQL DB instance again. ping 172.16.0.167

If information similar to the following is displayed, the ping command has been added and the cross-VPC access from the pod is successful:

PING 172.16.0.167 (172.16.0.167) 56(84) bytes of data. 64 bytes from 172.16.0.167: icmp_seq=1 ttl=63 time=0.516 ms 64 bytes from 172.16.0.167: icmp_seq=2 ttl=63 time=0.418 ms 64 bytes from 172.16.0.167: icmp_seq=3 ttl=63 time=0.376 ms --- 172.16.0.167 ping statistics ---3 packets transmitted, 3 received, 0% packet loss, time 1001ms

Accessing an RDS for MySQL DB instance from a pod in a CCE Turbo cluster that uses the Cloud Native 2.0 network model

- 1. Create a VPC peering connection between the cluster VPC and the destination VPC.
 - a. Switch to the console, click in the upper left corner, and choose **Networking** > **Virtual Private Cloud** in the expanded list.
 - b. In the navigation pane, choose **VPC Peering Connections**. In the upper right corner of the displayed page, click **Create VPC Peering Connection**.
 - c. Configure parameters following instructions. For details about some of the parameters, see **Table 10-25**.

Figure 10-60 Creating a VPC peering connection

< | Create VPC Peering Connection

 A VPC peering connection can connect VPCs from the same account or from different accounts as long as they are in the same region.Learn more ⁽²⁾ Creating a VPC Peering Connection with Another VPC in Your Account ⁽²⁾ Creating a VPC Peering Connection with a VPC in Another Account ⁽²⁾ If you want to connect VPCs in different regions, use Cloud Connect ⁽²⁾. 			
Basic Configuration			
Region	♥ CN East-Shanghai1 ∨		
VPC Peering Connection Na	ame peering-b34b		
Description (Optional)	0/255 1/2		
Local VPC Settings			
Local VPC	vpc-373896-1 V Q		
Local VPC CIDR Block	172.16.0.0/12		
Peer VPC Settings			
Account	My account ⑦		
Peer Project	cn-east-3 v		
	If you select My account, the project is filled in by default.		
Peer VPC	vpc-demo1 V		
Peer VPC CIDR Block	192.168.0.0/18, 172.1.0.0/24		

Table 10-25 Parameters for creating a VPC peering connection

Parameter	Description	Example Value
VPC Peering Connection Name	Mandatory. Enter a name for the VPC peering connection. The name can contain a maximum of 64	peering- b34b
	characters, including letters, digits, hyphens (-), and underscores (_).	
Local VPC	Mandatory. Local-end VPC of the peering connection. You can choose one from the drop-down list.	vpc-373896- 1

Parameter	Description	Example Value
Local VPC CIDR Block	CIDR block of the selected local-end VPC.	172.16.0.0/1 2
Account	Mandatory.	My account
	• My account : The local and peer VPCs are from the same account.	
	• Another account: The local and peer VPCs are from different accounts.	
Peer Project	The system fills in the corresponding project by default because Account is set to My account .	None
	For example, vpc-demo1 and vpc-demo2 are both under account A in region A. Then, the system fills in the project of account A in region A by default.	
Peer VPC	This parameter is mandatory if Account is set to My account .	vpc-demo1
	VPC at the other end of the peering connection. You can choose one from the drop-down list.	
Peer VPC CIDR Block	CIDR block of the selected peer VPC. NOTICE If the local and peer VPCs have overlapping CIDR blocks, the VPC peering connection may not take effect.	192.168.0.0/ 18 and 172.1.0.0/24

- d. After configuring the parameters, click **Create Now**.
- 2. In the displayed VPC Peering Connection Created dialog box, click Add Now and add a route for the container subnet and destination subnet. In the Add Route dialog box, configure the parameters following instructions. For details about the parameters, see Table 10-26.

Add Route			
* VPC	vpc-373896-1	~	
* Route Table	rtb-d43b	~	Q View Route Table
* Destination	192.168.60.0/28	×	
* Next Hop	peering-d34b(5ae9f0d3-43bb-4950-	-968f-fec9b4	
Description			
		0/255 🥢	
🗸 Add a route fo	or the other VPC		
To enable commur forward and return	nications between VPCs connected by a V routes to the route tables of the VPCs.Lea	PC peering cor arn more 🖸	nection, you need to add
* VPC	vpc-demo1	~	
* Route Table	rtb-vpc-demo1(Default)	~	Q View Route Table
Destination	172.16.0.0/24	~	

* Route Table	rtb-vpc-demo1(Default)	~	Q View Route Table
★ Destination	172.16.0.0/24	×)
★ Next Hop	peering-d34b(5ae9f0d3-43bb-495	0-968f-fec9b4	
Description			
		0/255 4	J
		\square	Cancel OK

Table 10-26 Parameters for adding a route for the container subnet and destination subnet

Parameter	Description	Example Value
VPC	Select a VPC that is connected by the VPC peering connection.	vpc-373896- 1
Parameter	Description	Example Value
-------------------------------------	---	---
Route Table	 Select the route table of the VPC. The route will be added to this route table. Each VPC comes with a default route table that manages the flow of outgoing traffic from the subnets in the VPC. In addition to the default route table, you can create a custom route table and associate it with the subnets in the VPC. Then, the custom route table controls outbound traffic of the subnets. If there is only the default route table in the drop-down list, select the default route table. If there are both default and custom route tables in drop-down list, select the route table associated with the subnet 	rtb-d43b (custom route table) NOTICE The custom route table must be associated with the subnet connected by the VPC peering connection.
	connected by the VPC peering connection.	
Destination	IP address in the VPC at the other end of the VPC peering connection. The value can be a VPC CIDR block, subnet CIDR block, or ECS IP address.	192.168.60.0/ 28
Add a route for the other VPC	If you select this option, you can also add a route for the other VPC connected by the VPC peering connection. To allow VPCs connected through VPC peering to communicate, you must include forward and return routes in the VPCs' route tables.	Selected
VPC	By default, the system selects the other VPC connected by the VPC peering connection. You do not need to specify this parameter.	vpc-demo1

Parameter	Description	Example Value
Route Table	Select the route table of the VPC. The route will be added to this route table.	rtb-vpc- demo1
	Each VPC comes with a default route table that manages the flow of outgoing traffic from the subnets in the VPC. In addition to the default route table, you can create a custom route table and associate it with the subnets in the VPC. Then, the custom route table controls outbound traffic of the subnets.	(default route table)
	• If there is only the default route table in the drop-down list, select the default route table.	
	• If there are both default and custom route tables in drop-down list, select the route table associated with the subnet connected by the VPC peering connection.	
Destination	IP address in the VPC at the other end of the VPC peering connection. The value can be a VPC CIDR block, subnet CIDR block, or ECS IP address.	172.16.0.0/24

3. Log in to the pod and enter the following code on the CloudShell page of the pod again, where *172.16.0.167* indicates the IP address of the RDS for MySQL DB instance to be accessed: (For details about how to log in to a container, see Logging In to a Container.) ping *172.16.0.167*

NOTE

If the access fails, check whether the traffic from the cluster container subnet is allowed in the inbound rules of the DB instance security group. If it is not allowed, you need to add a security group rule and allow the corresponding traffic. For details, see Adding a Security Group Rule.

If a ping command is available and information similar to the following is displayed, cross-VPC access from the pod is successful:
 PING 172.16.0.167 (172.16.0.167) 56(84) bytes of data.
 64 bytes from 172.16.0.167: icmp_seq=1 ttl=63 time=0.516 ms
 64 bytes from 172.16.0.167: icmp_seq=2 ttl=63 time=0.418 ms
 64 bytes from 172.16.0.167: icmp_seq=3 ttl=63 time=0.376 ms
 --- 172.16.0.167 ping statistics -- 3 packets transmitted, 3 received, 0% packet loss, time 1001ms

If no ping command is available, add it.
 ping: command not found

The following uses the **Nginx:latest** container as an example to describe how to add a ping command. If the ping command is already available, skip this step.

i. Ensure that the pod can access the Internet. For details, see **Accessing the Internet from a Pod**.

- ii. Update the local software package index and install the **iputils-ping** software package that provides the ping command. apt-get update apt-get install iputils-ping
- iii. Access the RDS for MySQL DB instance again. ping 172.16.0.167

If information similar to the following is displayed, the ping command has been added and the cross-VPC access from the pod is successful:

PING 172.16.0.167 (172.16.0.167) 56(84) bytes of data. 64 bytes from 172.16.0.167: icmp_seq=1 ttl=63 time=0.516 ms 64 bytes from 172.16.0.167: icmp_seq=2 ttl=63 time=0.418 ms 64 bytes from 172.16.0.167: icmp_seq=3 ttl=63 time=0.376 ms --- 172.16.0.167 ping statistics ---3 packets transmitted, 3 received, 0% packet loss, time 1001ms

Troubleshooting a Pod Access Failure

If a pod cannot access the network, rectify the fault by referring to **Table 10-27**. If the fault persists, **submit a service ticket** to contact Huawei Cloud customer service.

NOTE

The CIDR blocks used for pod access depend on the cluster network model. For details, see **Table 10-13**. The container CIDR blocks mentioned in the following section are specific to a cluster that uses the VPC network model.

Table 10-27	Troubleshooting	methods
-------------	-----------------	---------

Check Item	Possible Fault	Solution
Security group rules of the accessed service	 One of the following issues may be the cause of the failure: The security group's inbound rules prevent access to the node CIDR block or container CIDR block. The security group's inbound rules permit access to the node CIDR block and container CIDR block, but the protocol is incorrectly configured. NOTICE Run the ping command and use ICMP to test network connectivity. Before doing so, enable the ICMP port in the security group rule.	 For possible cause add a security group rule. For details, see Adding a Security Group Rule. For possible cause change the protocol port in the security group rules. For details, see Modifying a Security Group Rule.

Check Item	Possible Fault	Solution
VPC peering connection	The CIDR blocks of the local and peer VPCs are overlapping.	There are two possible causes for this issue. For details, see Overlapping CIDR Blocks of Local and Peer VPCs .
		• If the subnet where the node CIDR block resides does not overlap with the subnet where the accessed service resides, you can create a VPC peering connection that points to the subnet.
		 If the subnet where the node CIDR block resides overlaps with the subnet where the accessed service resides, you need to replan the network.
Route	 One of the following issues may be the cause of the failure: The custom route table used to add routes for the two VPCs is not associated with the destination subnet. (The cluster VPC route table must be associated with the node subnet, and the route table of the accessed service must be associated with the subnet where the accessed service is located.) There is no route between the accessed service subnet and the container CIDR block. 	 For possible cause 1, locate the target route table, click Associate Subnet, and select the correct subnet. For problem 2, add a route for the destination service subnet and the container CIDR block.
Trustlist	The trustlist for the accessed service does not have the node CIDR block and container CIDR block configured.	Add the container and node CIDR blocks to the trustlist. Find more information in the help document for the relevant service.

Check Item	Possible Fault	Solution
Domain name resolution	When accessing an external domain name, the pod uses its cluster's domain name resolution to resolve the destination address and accesses the address based on the pod's network policy. However, sometimes the domain name cannot be resolved, resulting in errors. The most common errors are listed below:	Locate the cause of the DNS exception. For details, see DNS Overview for troubleshooting.
	 Name or service not known 	
	 Temporary failure in name resolution 	
	Unable to resolve hostname	
	 DNS resolution failed 	
	 Could not resolve MYHOST (nodename nor servname known), where MYHOST indicates the domain name that cannot be resolved 	
Network policy (applicable only to tunnel networks)	If you have configured a network policy for both your tunnel network cluster and the namespace where the pod is located, the network policy may prevent the pod from accessing the destination address.	If so, modify the network policy. For details, see Configuring Network Policies to Restrict Pod Access .

10.11 Deploying Nginx Ingress Controllers Using a Chart

10.11.1 Deploying NGINX Ingress Controller in Custom Mode

Background

Nginx Ingress Controller is a popular open source ingress controller in the industry and is widely used. Large-scale clusters require multiple ingress controllers to distinguish different traffic. For example, if some services in a cluster need to be accessed through a public network ingress, but some internal services cannot be accessed through a public network and can only be accessed by other services in the same VPC, you can deploy two independent Nginx Ingress Controllers and bind two different load balancers.



Figure 10-62 Application scenario of multiple Nginx ingresses

Solution

You can use either of the following solutions to deploy multiple NGINX ingress controllers in the same cluster.

• (Recommended) Install the NGINX Ingress Controller add-on and deploy multiple instances in the same cluster with just a few clicks. For details, see Nginx Ingress Controller.

For clusters 1.23, the add-on of 2.2.52 or later must be installed. For clusters 1.23 or later, the add-on of 2.5.4 or later must be installed.

• Install the open source Helm package. The parameters to be configured in this solution are complex. You need to configure the **ingress-class** parameter (default value: **nginx**) to declare the listening ranges of different NGINX ingress controllers. In this way, when creating an ingress, you can select different NGINX ingress controllers to distinguish traffic.

Prerequisites

• Public images may need to be pulled during the installation. Therefore, bind an EIP to the node.

Notes and Constraints

- If multiple Nginx Ingress Controllers are deployed, each Controller needs to interconnect with a load balancer. Ensure that the load balancer has at least two listeners and ports 80 and 443 are not occupied by listeners. If dedicated load balancers are used, specify the network type.
- When the nginx-ingress template and image provided by the community are used, CCE does not provide additional maintenance for service loss caused by community software defects. **Exercise caution when serving commercial purposes.**

Deploying an Nginx Ingress Controller

You can perform the following operations to deploy another NGINX Ingress Controller in the cluster.

Step 1 Obtain a chart.

Go to the **chart page**, select a proper version, and download the Helm chart in **.tgz** format. This section uses the chart of version **4.4.2** as an example. This chart applies to CCE clusters of v1.21 or later. The configuration items in the chart may vary according to the version. The configuration in this section takes effect only for the chart of **4.4.2** version.

Step 2 Upload the chart.

- 1. Log in to the CCE console and click the cluster name to access the cluster console. In the navigation pane, choose **App Templates** and click **Upload Chart** in the upper right corner.
- 2. Click Select File, select the chart to be uploaded, and click Upload.

< cce-test *	Cluster: cce-test E CCE cluster / My Charts		P Quick Links Upload Chart
Cluster Information	My Charts Releases		2
BB Resources	A Helm, on which CCE charts depend, has evolved from v2 to v	Upload Chart ×	idions, but they may be incompatible with the ones in Helm v2. You need to check and adapt the
Nodes	tunctions Differences Between Helm v2 and Helm v3 and Ada If you have difficulties in switching to Helm v3 for now, you can	- 3 Salart and unlead a chart file	0, 2022. Deploying an Application Through the Helm v2 Client
Workloads			
Networking		Note 1. If you upload a chart package with the same name and version, the existing package	Enter a chart name. Q C
Storage	newer-nginx-ingress	will be overridden. 2. The TGZ package to be uploaded must be named in the format of name-version.tgz.	
ConfigMaps and Secrets	Ingress controller for Kubernetes using NGINX	The value of name cannot exceed 64 characters. 3. OBS bucket	
Custom Resources	Latest Version 4.4.2	created, which is used only to store charts. 4. A chart with multiple versions consumes the same amount of portion of chart quota.	
Namespaces	Versions 1	Available chart quota: 198	
& 08M	Updated 68 days ago	Interd Canal	
Node Scaling	Install Update Download Delete	Cynodd Coenson	
Workload Scaling			
Add-ons			
Charts 1			
Cluster Upgrade			
Logging OBT			

Step 3 Customize the value.yaml file.

You can create a **value.yaml** configuration file on the local PC to configure workload installation parameters. During workload installation, you only need to import this configuration file for customized installation. Other unspecified parameters will use the default settings.

```
The configuration content is as follows:
controller:
 image:
  repository: registry.k8s.io/ingress-nginx/controller
  registry: "
image: ""
  tag: "v1.5.1" # Controller version
  digest: ""
 ingressClassResource:
  name: ccedemo
                        # The name of each Ingress Controller in the same cluster must be unique and
cannot be nginx or cce.
  controller Value: "k8s.io/ingress-nginx-demo" # The listening identifier of each Ingress Controller in the
same cluster must be unique and cannot be set to k8s.io/ingress-nginx.
ingressClass: ccedemo # The name of each Ingress Controller in the same cluster must be unique and
cannot be nginx or cce.
 service:
  annotations:
    kubernetes.io/elb.id: 5083f225-9bf8-48fa-9c8b-67bd9693c4c0 #ELB ID
    kubernetes.io/elb.class: performance # This annotation is required only for dedicated load balancers.
 config:
  keep-alive-requests: 100
 extraVolumeMounts: # Mount the /etc/localtime file on the node to synchronize the time zone.
   - name: localtime
    mountPath: /etc/localtime
    readOnly: true
 extraVolumes:
```

```
- name: localtime
   type: Hostpath
    hostPath:
     path: /etc/localtime
 admissionWebhooks: # Disable webhook authentication.
  enabled: false
  patch:
   enabled: false
 resources: # Set the controller's resource limit, which can be customized.
  requests:
   cpu: 200m
   memory: 200Mi
defaultBackend: # Set defaultBackend.
 enabled: true
 image:
  repository: registry.k8s.io/defaultbackend-amd64
  registry: ""
  image: ""
  tag: "1.5"
  digest: ""
```

For details about the preceding parameters, see Table 10-28.

Step 4 Create a release.

- 1. Log in to the CCE console and click the cluster name to access the cluster console. In the navigation pane, choose **App Templates**.
- 2. In the list of uploaded charts, click Install.
- 3. Set Release Name, Namespace, and Version.
- 4. Click **Select File** next to **Configuration File**, select the YAML file created locally, and click **Install**.
- 5. On the **Releases** tab page, you can view the installation status of the release.

Uninstall							Release Name	▼ Search keyword	QC
Release Name ↓Ξ	Status ↓Ξ	Namespace	Chart Name ↓Ξ	Chart Version	Release Versi	Updated ↓	Latest Event	Operation	
ingress-demo	 Installing 	default	new-ingress-n	4.2.5	1	1 minutes ago	Initial install underway	Upgrade Roll Bac	K More 💌

----End

Performing Verification

Deploy a workload and configure the newly deployed Nginx Ingress Controller to provide network access for the workload.

- **Step 1** Create an Nginx workload.
 - 1. Log in to the CCE console, click the created cluster, choose **Workloads** in the navigation pane, and click **Create Workload** in the upper right corner.
 - 2. Enter the following information and click OK. apiVersion: apps/v1 kind: Deployment metadata: name: nginx spec: replicas: 1 selector: matchLabels: app: nginx strategy: type: RollingUpdate template: metadata:

```
labels:
    app: nginx
  spec:
    containers:
    - image: nginx # If an image from an open source image registry is used, enter the image
name. If you use an image in My Images, obtain the image path from SWR.
     imagePullPolicy: Always
     name: nginx
   imagePullSecrets:
    - name: default-secret
apiVersion: v1
kind: Service
metadata:
 labels:
  app: nginx
 name: nginx
spec:
 ports:
 - name: service0
  port: 80
                      # Port for accessing a Service.
  protocol: TCP
                       # Protocol used for accessing a Service. The value can be TCP or UDP.
  targetPort: 80
                       # Port used by the service to access the target container. In this example, the
Nginx image uses port 80 by default.
selector:
                      # Label selector. A Service selects a pod based on the label and forwards the
requests for accessing the Service to the pod.
  app: nginx
 type: ClusterIP
                       # Type of a Service. ClusterIP indicates that a Service is only reachable from
within the cluster.
```

- **Step 2** Create an ingress and use the newly deployed Nginx Ingress Controller to provide network access.
 - 1. In the navigation pane, choose **Services & Ingresses**. Click the **Ingresses** tab and click **Create from YAML** in the upper right corner.

NOTE

When interconnecting with Nginx Ingress Controller that is not deployed using an addon, you can create an ingress only through YAML.

2. Enter the following information and click **OK**.

```
For clusters of v1.23 or later:
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
name: ingress-test
namespace: default
spec:
 ingressClassName: ccedemo # Enter the ingressClass of the newly created Nginx Ingress Controller.
rules:
 - host: foo.bar.com
  http:
   paths:
    - path: /
     pathType: ImplementationSpecific # The matching depends on IngressClass.
     backend:
      service:
       name: nginx # Replace it with the name of your target Service.
       port:
        number: 80 # Replace it with the port of your target Service.
     property:
      ingress.beta.kubernetes.io/url-match-mode: STARTS_WITH
```

For clusters earlier than v1.23:

```
apiVersion: networking.k8s.io/v1beta1
kind: Ingress
metadata:
```

```
name: tomcat-t1
 namespace: test
 annotations:
  kubernetes.io/ingress.class: ccedemo # Enter the ingressClass of the newly created Nginx Ingress
Controller.
spec:
 rules:
  - host: foo.bar.com
   http:
     paths:
       - path: /
        pathType: ImplementationSpecific
        backend:
         serviceName: nginx # Replace it with the name of your target Service.
         servicePort: 80
                          # Replace it with the port of your target Service.
       property:
         ingress.beta.kubernetes.io/url-match-mode: STARTS_WITH
```

- **Step 3** Log in to the cluster node and access the application through the Controller in the nginx-ingress add-on of the cluster and the newly deployed Nginx Ingress Controller service, respectively.
 - Use the new Nginx Ingress Controller service to access the application (the Nginx page is expected to be displayed). **192.168.114.60** is the ELB address of the new Nginx Ingress Controller service. curl -H "Host: foo.bar.com" http://192.168.114.60

```
[root@192-168-9-72 paas]# curl -H "Host: foo.bar.com" http://192.168.114.60
<!DOCTYPE html>
<html>
<head>
<title>Welcome to nginx!</title>
<style>
    body {
       width: 35em;
        margin: 0 auto;
        font-family: Tahoma, Verdana, Arial, sans-serif;
    }
</style>
</head>
<body>
<h1>Welcome to nginx!</h1>
If you see this page, the nginx web server is successfully installed and
working. Further configuration is required.
For online documentation and support please refer to
<a href="http://nginx.org/">nginx.org</a>.<br/>
Commercial support is available at
<a href="http://nginx.com/">nginx.com</a>.
<em>Thank you for using nginx.</em>
</body>
</html>
```

• Use the Controller service in the nginx-ingress add-on (404 is expected to be returned). **192.168.9.226** is the ELB address of the nginx-ingress add-on. curl -H "Host: foo.bar.com" http://192.168.9.226

```
[root@192-168-9-72 paas]# curl -H "Host: foo.bar.com" http://192.168.9.226
default backend - 404[root@192-168-9-72 paas]#
```

----End

Parameters

Table	10-28	nginx-ingress	parameters
-------	-------	---------------	------------

Parameter	Description
controller.image.reposito ry	ingress-nginx image address. It is recommended that this parameter be set to the same as the nginx- ingress add-on image provided by CCE. You can also customize the parameter.
	 nginx-ingress add-on image: You can view its image path in the YAML file of the installed add- on.
	 Custom: The custom path must ensure that the image can be pulled.
controller.image.registry	Domain name of the image repository. This parameter must be set together with controller.image.image .
	If controller.image.repository has been set, you do not need to set this parameter. You are advised to leave controller.image.registry and controller.image.image empty.
controller.image.image	Image name. This parameter must be set together with controller.image.registry .
	If controller.image.repository has been set, you do not need to set this parameter. You are advised to leave controller.image.registry and controller.image.image empty.
controller.image.tag	ingress-nginx image version. It is recommended that this parameter be set to the same as the nginx- ingress add-on image provided by CCE. You can also customize the image.
	The image version of the nginx-ingress add-on can be viewed in the YAML file of the installed add-on and needs to be replaced based on the add-on version.
controller.ingressClass	Specifies the name of the IngressClass of the Ingress Controller.
	NOTE The name of each Ingress Controller in the same cluster must be unique and cannot be set to nginx or cce . nginx is the default listening identifier of Nginx Ingress Controller in the cluster, and cce is the configuration of ELB Ingress Controller. Example: ccedemo
controller.image.digest	You are advised to leave this parameter empty. If this parameter is specified, pulling the nginx- ingress add-on image provided by CCE may fail.

Parameter	Description
controller.ingressClassRe source.name	The parameter value must be the same as that of ingressClass. Example: ccedemo
controller.ingressClassRe source.controllerValue	The listening identifier of each Ingress Controller in the same cluster must be unique and cannot be set to k8s.io/ingress-nginx , which is the default listening identifier of Nginx Ingress Controller. Example: k8s.io/ingress-nginx-demo
controller.config	Nginx configuration parameter. For details, see Community Documents . Parameter settings out of the range do not take effect. You are advised to add the following configurations: "keep-alive-requests": "100"
controller.extraInitContai ners	 init container, which is executed before the main container is started and can be used to initialize pod parameters. For details about parameter configuration examples, see Parameter Optimization in High-Concurrency Scenarios.
controller.admissionWeb hooks.enabled	Specifies whether to enable admissionWebhooks to verify the validity of ingress objects. This prevents ingress-controller from continuously reloading resources due to incorrect configurations, which may cause service interruption. Set this parameter to false , indicating that the function is disabled. To enable this function, see the example in admissionWebhook Configuration .
controller.admissionWeb hooks.patch.enabled	Specifies whether to enable admissionWebhooks. Set this parameter to false .
controller.service.annotat ions	A key-value pair. The ELB ID needs to be added, as shown in the following: kubernetes.io/elb.id: 5083f225-9bf8-48fa-9c8b-67bd9693c4c0 For dedicated load balancers, add elb.class as follows: kubernetes.io/elb.class: performance
controller.resources.requ ests.cpu	Specifies the quantity of CPU resources requested by the Nginx controller. This parameter can be customized.
controller.resources.requ ests.memory	Specifies the quantity of memory resources requested by the Nginx controller. This parameter can be customized.

Parameter	Description
defaultBackend.image.re pository	default-backend image path. It is recommended that this parameter be set to the same as the nginx- ingress add-on image provided by CCE. You can also customize the parameter.
	 nginx-ingress add-on image: You can view its image path in the YAML file of the installed add-on. Custom: The custom path must ensure that the image can be pulled.
	inage can be pulled.
defaultBackend.image.ta g	default-backend image version. It is recommended that this parameter be set to the same as the nginx- ingress add-on image provided by CCE. You can also customize the image.

For details about more parameters, see ingress-nginx.

10.11.2 Advanced Configuration of Nginx Ingress Controller

Parameter Optimization in High-Concurrency Scenarios

In high-concurrency scenarios, you can configure parameters for optimization in either of the following ways:

- 1. Use ConfigMap to optimize the overall parameters of Nginx Ingress Controller.
- 2. Use InitContainers to optimize the kernel parameters of Nginx Ingress Controller.

The optimized value.yaml configuration file is as follows:

```
controller:
 image:
  repository: registry.k8s.io/ingress-nginx/controller
  registry: "
image: ""
  tag: "v1.5.1" # Controller version
  digest: ""
 ingressClassResource:
  name: ccedemo
                        # The name of each Ingress Controller in the same cluster must be unique and
cannot be nginx or cce.
  controller Value: "k8s.io/ingress-nginx-demo" # The listening identifier of each Ingress Controller in the
same cluster must be unique and cannot be set to k8s.io/ingress-nginx.
ingressClass: ccedemo # The name of each Ingress Controller in the same cluster must be unique and
cannot be nginx or cce.
 service:
  annotations:
    kubernetes.io/elb.id: 5083f225-9bf8-48fa-9c8b-67bd9693c4c0 #ELB ID
  kubernetes.io/elb.class: performance # This annotation is required only for dedicated load balancers.
# Nginx parameter optimization
 config:
  keep-alive-requests: 10000
  upstream-keepalive-connections: 200
  max-worker-connections: 65536
# Kernel parameter optimization
 extraInitContainers:
```

- name: init-myservice
image: busybox
securityContext:
privileged: true
command: ['sh', '-c', 'sysctl -w net.core.somaxconn=65535;sysctl -w
net.ipv4.ip_local_port_range="1024 65535""]
extraVolumeMounts: # Mount the /etc/localtime file on the node to synchronize the time zone.
- name: localtime
mountPath: /etc/localtime
readOnly: true
extraVolumes:
- name: localtime
type: Hostpath
hostPath:
path: /etc/localtime
admissionWebhooks: # Disable webhook authentication.
enabled: false
patch:
enabled: false
resources: # Set the controller's resource limit, which can be customized.
requests:
cpu: 200m
memory: 200Mi
defaultBackend: # Set defaultBackend.
enabled: true
image:
repository: registry.k8s.io/defaultbackend-amd64
registry: ""
image: ""
tag: "1.5"
digest: ""

admissionWebhook Configuration

Nginx Ingress Controller supports admissionWebhook configuration. You can configure the **controller.admissionWebhook** parameter to verify the validity of ingress objects. This prevents ingress-controller from continuously reloading resources due to incorrect configuration, which may cause service interruption.

NOTE

• When the admissionWebhook feature is used, webhook-related configurations must be enabled on the API server, including MutatingAdmissionWebhook and ValidatingAdmissionWebhook.

The feature switch is --admission-

control = Mutating Admission We bhook, Validating Admission We bhook.

If it is not enabled, submit a service ticket to enable it.

• After admissionWebhook is enabled, if you need to uninstall and reinstall Nginx Ingress Controller, residual secrets exist and need to be manually cleared.

The **value.yaml** configuration file for enabling admissionWebhook is as follows:

```
controller:

image:

repository: registry.k8s.io/ingress-nginx/controller

registry: ""

image: ""

tag: "v1.5.1" # Controller version

digest: ""

ingressClassResource:

name: ccedemo # The name of each Ingress Controller in the same cluster must be unique and

cannot be nginx or cce.

controllerValue: "k8s.io/ingress-nginx-demo" # The listening identifier of each Ingress Controller in the

same cluster must be unique and cannot be set to k8s.io/ingress-nginx.

ingressClass: ccedemo # The name of each Ingress Controller in the same cluster must be unique and
```

cannot be nginx or cce. service: annotations: kubernetes.io/elb.id: 5083f225-9bf8-48fa-9c8b-67bd9693c4c0 #FLB ID kubernetes.io/elb.class: performance # This annotation is required only for dedicated load balancers. config: keep-alive-requests: 100 extraVolumeMounts: # Mount the /etc/localtime file on the node to synchronize the time zone. - name: localtime mountPath: /etc/localtime readOnly: true extraVolumes: - name: localtime type: Hostpath hostPath: path: /etc/localtime admissionWebhooks: annotations: {} enabled: true extraEnvs: [] failurePolicy: Fail port: 8443 . certificate: "/usr/local/certificates/cert" key: "/usr/local/certificates/key" namespaceSelector: {} objectSelector: {} labels: {} existingPsp: "" networkPolicyEnabled: false service: annotations: {} externalIPs: [] loadBalancerSourceRanges: [] servicePort: 443 type: ClusterIP createSecretJob: resources: #Annotation{} limits: cpu: 20m memory: 40Mi requests: cpu: 10m memory: 20Mi patchWebhookJob: resources: {} patch: enabled: true image: registry: registry.k8s.io #registry.k8s.io is the image repository of the webhook official website. Replace it with the address of the repository where the image is located. # webhook image image: ingress-nginx/kube-webhook-certgen tag: v1.1.1 digest: " pullPolicy: IfNotPresent priorityClassName: " podAnnotations: {} nodeSelector: kubernetes.io/os: linux tolerations: [] labels: {} securityContext: runAsNonRoot: true runAsUser: 2000 fsGroup: 2000 resources: # Set the controller's resource limit, which can be customized. requests: cpu: 200m memory: 200Mi defaultBackend: # Set defaultBackend.

```
enabled: true
image:
repository: registry.k8s.io/defaultbackend-amd64
registry: ""
image: ""
tag: "1.5"
digest: ""
```

Check whether admissionWebhook is verified when incorrect annotations are configured for the ingress.

For example, configure the following incorrect annotations for the ingress:

```
annotations:
nginx.ingress.kubernetes.io/auth-tls-pass-certificate-to-upstream: "false"
nginx.ingress.kubernetes.io/auth-tls-verify-client: optional
nginx.ingress.kubernetes.io/auth-tls-verify-depth: "1"
```

When the ingress service is created, the following interception information is displayed:

You can run `kubectl replace -f /tmp/kubectl-edit-v7a52.yaml` to try this update again. [rootdclient-server ~]# kubectl edit ingress ingress-demo -n hzj-test error: ingresses.networking.k8s.io "ingress-demo" could not be patched: admission webhook "validate.nginx.ingress.ku) netes.io" denied the <mark>request</mark> :
Error: exit status 1
2023/04/03 17:51:53 [warn] 630##30: the "http2_max_field_size" directive is obsolete, use the "large_client_header_bu
ers offective instead in /tmp/nginx/nginx-crg1129/50669:143 nginx: [warn] the "http2_max_field_size" directive is obsolete, use the "large_client_header_buffers" directive inste in /tmp/nginx/nginx-crg1129750869:143
2023/04/03 17:51:53 [warn] 630#630: the "http2_max_header_size" directive is obsolete, use the "large_client_header_b fers" directive instead in /tmp/nginx/nginx-cfg1129750869:144
nginx: [warn] the "http2_max_header_size" directive is obsolete, use the "large_client_header_buffers" directive inst d in /tmp/nginx/nginx-cfg1129750869:144
2023/04/03 17:51:53 [warn] 630#630: the "http2_max_requests" directive is obsolete, use the "keepalive_requests" dire ive instead in /tmp/nginx/nginx-cfg1129750869:145
nginx: [warn] the "http2_max_requests" directive is obsolete, use the "keepalive_requests" directive instead in /tmp/ jax/nainx-cfa1129750869:145
2023/04/03 17:51:53 [emerg] 630#630: "proxy_buffering" directive is duplicate in /tmp/nginx/nginx-cfg1129750869:407 nginx: [emerg] "proxy_buffering" directive is duplicate in /tmp/nginx/nginx-cfg1129750869:407 nginx: configuration file /tmp/nginx/nginx-cfg1129750869 test failed

10.12 CoreDNS Configuration Optimization

10.12.1 CoreDNS Optimization Overview

Application Scenarios

DNS is one of the important basic services in Kubernetes. When the container DNS policy is not properly configured and the cluster scale is large, DNS resolution may time out or fail. In extreme cases, a large number of services in the cluster may fail to be resolved. This section describes the best practices of CoreDNS configuration optimization in Kubernetes clusters to help you avoid such problems.

Solution

CoreDNS configuration optimization includes clients and servers.

On the client, you can optimize domain name resolution requests to reduce resolution latency, and use proper container images and NodeLocal DNSCache to reduce resolution exceptions.

- Optimizing Domain Name Resolution Requests
- Selecting a Proper Image
- Avoiding Occasional DNS Resolution Timeout Caused by IPVS Defects
- Using NodeLocal DNSCache
- Upgrading the CoreDNS in the Cluster Timely
- Adjusting the DNS Configuration of the VPC and VM

On the server, you can adjust the CoreDNS deployment status or CoreDNS configuration to improve the availability and throughput of CoreDNS in the cluster.

- Monitoring the coredns Add-on
- Adjusting the CoreDNS Deployment Status
- Configuring CoreDNS

For more information about CoreDNS configurations, see https://coredns.io/.

CoreDNS open-source community: https://github.com/coredns/coredns

Prerequisites

- You have created a CCE cluster. For details about how to create a cluster, see **Buying a CCE Standard/Turbo Cluster**.
- You can access the cluster using kubectl. For details, see **Connecting to a Cluster Using kubectl**.
- The CoreDNS add-on is installed in the cluster. The latest version of CoreDNS is recommended. For details, see **CoreDNS**.

10.12.2 Client

10.12.2.1 Optimizing Domain Name Resolution Requests

DNS resolution is frequently used in Kubernetes clusters. Based on the characteristics of DNS resolution in Kubernetes, you can optimize domain name resolution requests in the following ways.

Using a Connection Pool

When a containerized application needs to frequently request another service, you are advised to use a connection pool. The connection pool can cache the link information of the upstream service to avoid the overhead of DNS resolution and TCP link reestablishment for each access.

Optimizing the resolve.conf File in the Container

The **ndots** and **search** parameters in the **resolve.conf** file determine the domain name resolution efficiency. For details about the two parameters, see **DNS Configuration**.

Optimizing the Domain Name Configuration

When a container needs to access a domain name, configure the domain name based on the following rules to improve the domain name resolution efficiency.

- 1. When a pod accesses a Service in the same namespace, use *<service-name>*, which indicates the Service name.
- 2. When a pod accesses a Service across namespaces, use *<service-name>.<namespace-name>*. *<namespace-name>* indicates the namespace where the Service is located.
- 3. When a pod accesses a domain name outside the cluster, it uses the FQDN domain name. This type of domain name is specified by adding a period (.) at the end of a typical domain name to avoid multiple invalid search attempts caused by search domain combination. For example, to access **www.huaweicloud.com**, use the FQDN domain name **www.huaweicloud.com**.

Using Local Cache

If the cluster specifications are large and the number of DNS resolution requests is large, you can cache the DNS resolution result on the node. You are advised to use NodeLocal DNSCache. For details, see **Using NodeLocal DNSCache to Improve DNS Performance**.

10.12.2.2 Selecting a Proper Image

The musl libc library of the Alpine container image differs from the standard glibc library in the following aspects:

- Alpine 3.3 and earlier versions do not support the **search** parameter. As a result, search domains cannot be specified for discovering Services.
- Multiple DNS servers configured in **/etc/resolve.conf** are concurrently requested. As a result, NodeLocal DNSCache cannot improve the DNS performance.
- When the same Socket is used to request A and AAAA records concurrently, the Conntrack source port conflict is triggered in the kernel of an earlier version. As a result, packet loss occurs.
- If the domain name cannot be resolved when Alpine is used as the base container image, update the base container image for testing.

For details about the functional differences from glibc, see **Functional differences from glibc**.

10.12.2.3 Avoiding Occasional DNS Resolution Timeout Caused by IPVS Defects

Description

When kube-proxy uses IPVS load balancing, you may encounter DNS resolution timeout occasionally during CoreDNS scale-in or restart.

This problem is caused by a Linux kernel defect. For details, see https://github.com/torvalds/linux/commit/ 35dfb013149f74c2be1ff9c78f14e6a3cd1539d1.

Solution

You can use NodeLocal DNSCache to minimize the impact of IPVS defects. For details, see **Using NodeLocal DNSCache to Improve DNS Performance**.

10.12.2.4 Using NodeLocal DNSCache

Challenges

When the number of DNS requests in a cluster increases, the load of CoreDNS increases and the following issues may occur:

- Increased delay: CoreDNS needs to process more requests, which may slow down the DNS query and affect service performance.
- Increased resource usage: To ensure DNS performance, CoreDNS requires higher specifications.

Solution

NodeLocal DNSCache can improve the stability and performance of service discovery.

For details about NodeLocal DNSCache and how to deploy it in a cluster, see **Using NodeLocal DNSCache to Improve DNS Performance**.

10.12.2.5 Upgrading the CoreDNS in the Cluster Timely

CoreDNS provides simple functions and is compatible with different Kubernetes versions. CCE periodically synchronizes bugs from the community and upgrades the coredns add-on. You are advised to periodically upgrade the CoreDNS. The CCE add-on management center supports the CoreDNS installation and upgrade. You can define the CoreDNS version in the cluster. If the version can be upgraded, upgrade the CoreDNS component in the cluster as soon as possible.

You can upgrade CoreDNS in a cluster by performing the following procedure:

- **Step 1** Log in to the CCE console and click the name of the target cluster to access the cluster console. In the navigation pane, choose **Add-ons**.
- **Step 2** Locate the CoreDNS add-on and click **Upgrade**.

DNS	coredns Service discovery		\$
	Version	1.23.1	New version available.
	Status		Running
	Updated		53 days ago
U	lpgrade		Uninstall

Step 3 Set parameters as prompted. For details, see coredns (System Resource Add-on, Mandatory).

----End

10.12.2.6 Adjusting the DNS Configuration of the VPC and VM

When the coredns add-on is started, it obtains the DNS configuration in the **resolve.conf** file from the deployed instance by default and uses the configuration as the upstream resolution server address. Before the coredns add-on is restarted, the **resolve.conf** configuration on the node is not reloaded. Suggestions:

- Ensure that the **resolve.conf** configuration of each node in the cluster is the same. In this way, the coredns add-on can schedule requests to any node in the cluster.
- When modifying the **resolve.conf** file, if the node has a coredns add-on, restart the coredns add-on timely to ensure status consistency.

10.12.3 Server

10.12.3.1 Monitoring the coredns Add-on

- CoreDNS exposes health metrics such as resolution results through the standard Prometheus API to detect exceptions on the CoreDNS server or even upstream DNS server.
- Port for obtaining coredns metrics. The default zone listening IP address is
 {\$*POD_IP*}:9153. Retain the default value. Otherwise, Prometheus cannot
 collect the metrics.
- If you use on-premises Prometheus to monitor Kubernetes clusters, you can observe related metrics on Prometheus and configure alarms for the key metrics. For details, see **prometheus**.

10.12.3.2 Adjusting the CoreDNS Deployment Status

In CCE clusters, the CoreDNS add-on is installed by default, and it can run on the same cluster nodes as your service containers. You need to pay attention to the following points when deploying CoreDNS:

- Properly Changing the Number of CoreDNS Replicas
- Properly Deploying the CoreDNS Pods
- Deploying CoreDNS Separately Using Custom Parameters
- Automatically Expanding the CoreDNS Capacity Based on the HPA

Properly Changing the Number of CoreDNS Replicas

You are advised to set the number of CoreDNS replicas to at least 2 in any case and keep the number of replicas within a proper range to support the resolution of the entire cluster. The default number of pods for installing the add-on in a CCE cluster is **2**.

- Modifying the number of CoreDNS replicas, CPUs, and memory size will change CoreDNS' parsing capability. Therefore, evaluate the impact before the operation.
- By default, podAntiAffinity (pod anti-affinity) is configured for the add-on. If a node already has a CoreDNS pod, no new pod can be added. That is, only one CoreDNS pod can run on a node. If there are more configured CoreDNS replicas than cluster nodes, the excess pods cannot be scheduled. Therefore, keep the number of replicas less than or equal to the number of nodes.

Properly Deploying the CoreDNS Pods

- By default, podAntiAffinity (pod anti-affinity) is configured for CoreDNS, so CoreDNS pods are forcibly deployed on different nodes in a cluster. It is recommended that you deploy CoreDNS pods on nodes in different AZs to prevent the add-on from being interrupted by faults in a single AZ.
- The CPU and memory of the cluster node where the coredns add-on runs must not be used up. Otherwise, the QPS and response of domain name resolution will be affected. It is recommended that you use the custom parameters to **deploy CoreDNS separately**.

Deploying CoreDNS Separately Using Custom Parameters

It is recommended that CoreDNS be deployed separately from resource-intensive workloads to prevent CoreDNS performance deterioration or unavailability due to service fluctuation. You can customize parameters to deploy CoreDNS on a dedicated node.

NOTE

There should be more nodes than CoreDNS pods. You need to avoid deploying multiple CoreDNS pods on a single node.

Step 1 Log in to the CCE console and click the cluster name to access the cluster console. In the navigation pane, choose **Nodes**.

Step 2 Click the **Nodes** tab, select the node dedicated for CoreDNS, and click **Labels and Taints** above the node list.

Add the following labels:

- Key: node-role.kubernetes.io/coredns
- Value: true

Add the following taints:

- Key: node-role.kubernetes.io/coredns
- Value: true
- Effect: NoSchedule

Figure 10-63 Adding a label and a taint

Labels and Taints				×
A Changing the Kubernetes labels or taints of a node will only affect	the scheduling of new	oods on that node, not the existing	ones.	
Batch Operation You can add, update, or delete labels or taints of specified nodes in batc'	hes.			
Add/Up V Kubernetes Label V node-role kubernetes	io/c true	Delete		
Add/Up V Inode-role.kubernetes	io/c true	NoSchedule	✓ Delete	
+ Add Operation				

- **Step 3** In the navigation pane, choose **Add-ons**, locate **CoreDNS**, and click **Edit**.
- Step 4 Select Custom Policies for Node Affinity and add the preceding node label.Add tolerations for the preceding taint.

Figure 10-64 Adding a toleration

Node Affinity	Not configured	Node Affinity	Specified Node Po	ol Scheduling C	ustom Policies	
	Users can fill in the node lat be performed according to t	bel of the desired plug-in d he cluster's default schedu	teployment to achieve a uling strategy	more flexible scheduling st	rategy. If not filled in, random so	heduling will
	Label Key		Operator 🥎	Label Value		Operation
	node-role.kubernetes.io	/coredns	In	✓ true		Delete
	⊕ Add Policy					
Toleration (?)	 By default, two tolers seconds. This impro 	ance policies are added to ves the plug-in availability	o the plug-in. When a noc . You are advised not to r	de is abnormal, the plug-in modify the policy.	deployment instance is evicted	after 60
	Taint key 🕜	Operator ⑦ T	aint value	Taint Policy	Toleration Time Windo	Operation
	node.kubernetes.io	Exists ~		NoExecute ~	60	Delete
	node.kubernetes.io	Exists ~ -		NoExecute ~	60	Delete
	node-role.kubernete	Equal V	true	NoSchedule v		Delete

Step 5 Click OK.

----End

Automatically Expanding the CoreDNS Capacity Based on the HPA

HPA frequently scales down the number of the coredns add-on replicas. Therefore, you are advised not to use HPA. If HPA is required, you can configure HPA auto scaling policies using the CCE Advanced HPA add-on. The process is as follows:

- Step 1 Log in to the CCE console and click the name of the target cluster to access the cluster console. In the navigation pane, choose Add-ons, locate the CCE Advanced HPA add-on on the right, and click Install.
- **Step 2** Configure the add-on parameters and click **Install**. For details about the add-on, see **CCE Advanced HPA**.
- Step 3 In the navigation pane, choose Workloads, select the kube-system namespace, locate the row containing the CoreDNS pod, and click Auto Scaling in the Operation column.

In the **HPA Policies** area, you can customize HPA policies based on metrics such as CPU usage and memory usage to automatically scale out the CoreDNS pods.

Auto ocaning	
Policy Type	HPA+CronHPA CustomedHPA Policies
	Configure either HPA+CronHPA or CustomedHPA policies. You are advised to configure HPA+CronHPA policies.
HPA Policies Workload pods are	scaled when system metrics (CPU usage and memory usage) and custom Prometheus metrics meet specified requirements. HPA Policy
Policy	
Pod Range	1 - 10 The workload pods are scaled within this range when the policy is triggered.
Cooldown Period	Scale-in 5 minutes Scale-out 3 minutes The same scaling operation will not be triggered again within the specified intervals.
System Policy	Metric ⑦ Desired Value ⑦ Tolerance Range ⑦ Operation
	CPU usage 70 % 63 % - 77 % Delete
	+

Figure 10-65 Creating an auto scaling policy

Step 4 Click **Create**. If the latest status is **Started**, the policy has taken effect.

----End

Auto Cooling

10.12.3.3 Configuring CoreDNS

On the console, the CoreDNS add-on can only be configured with the preset specifications, which can satisfy most of the service requirements. In some scenarios where there are requirements on the CoreDNS resource usage, you may need to customize the add-on specifications.

CoreDNS official document: https://coredns.io/plugins/

Configuring CoreDNS Specifications

- **Step 1** Log in to the CCE console and click the cluster name to access the cluster console.
- Step 2 In the navigation pane, choose Add-ons. Then, click Edit under CoreDNS.
- Step 3 In the Specifications area, configure coredns specifications.
- **Step 4** Change the number of replicas, CPU quotas, and memory quotas as needed to adjust the domain name resolution QPS provided by CoreDNS.

Specifications					
Pods	- 2 +				
Containers	coredns	CPU Quota Request 500m	Limit 500m	Memory Quota Request 512Mi	Limit 512Mi
	 The request must b Ensure that node re If your service imposed specifications 	e less than or equal to the l sources are sufficient in the uses special requirements o	limit. Otherwise, the creatio e cluster. If not, the add-on n domain name resolution,	n fails. It is recommende instance cannot be sche adjust the parameter set	d that they are set to the same value. duled and you need to reinstall the add-on. tings as needed. Configure Add-on

Step 5 Click OK.

----End

Properly Configuring the Stub Domain for DNS

- **Step 1** Log in to the CCE console and click the cluster name to access the cluster console.
- **Step 2** In the navigation pane, choose **Add-ons**. Then, click **Edit** under **CoreDNS**.
- Step 3 Add a stub domain in the Parameters area. The format is a key-value pair. The key is a DNS suffix domain name, and the value is a DNS IP address or a group of DNS IP addresses, for example, consul.local -- 10.150.0.1.

Figure 10-66 Adding a stub domain

Stub Domain		
consul.local	10.150.0.1	Delete
+ Add		
A domain name server for a custom do "acme.local – 1.2.3.4,6.7.8.9" means th	main name in key-value pair. Th hat DNS requests with the ".acm	e key is a suffix of DNS domain name, and the value is one or more DNS IP addresses. For example, ie.local" suffix are forwarded to a DNS server listening at 1.2.3.4,6.7.8.9.
Corefile:		
<pre>::5353 { bind {\$POD_IP} cache 30 { servfail 5s } errors health {\$POD_IP}:8080 kubernetes cluster.loca pods insecure fallthrough in-addr.a } loadbalance round_rot promethous {\$POD_IP}</pre>) Il in-addr.arpa ip6.ar arpa ip6.arpa Din 1:0152	rpa {

```
forward . /etc/resolv.conf {
    policy random
  }
  reload
  ready {$POD_IP}:8081
}
consul.local:5353 {
    bind {$POD_IP}
    errors
    cache 30
    forward . 10.150.0.1
}
```

- Step 4 Click OK.
- **Step 5** Choose **ConfigMaps and Secrets** in the navigation pane, select the **kube-system** namespace, and view the ConfigMap data of **coredns** to check whether the update is successful.

----End

Properly Configuring the Host

To specify hosts for a specific domain name, you can use the hosts add-on. An example is as follows:

- **Step 1** Log in to the CCE console and click the cluster name to access the cluster console.
- **Step 2** In the navigation pane, choose **Add-ons**. Then, click **Edit** under **CoreDNS**.
- **Step 3** Edit extended parameters in **Parameters** and add the following content to the **plugins** field:

```
{

"configBlock": "192.168.1.1 www.example.com\nfallthrough",

"name": "hosts"
```

NOTICE

The **fallthrough** field must be configured. **fallthrough** indicates that when the domain name to be resolved cannot be found in the hosts file, the resolution task is transferred to the next add-on of CoreDNS. If **fallthrough** is not specified, the task ends and the domain name resolution stops. As a result, the domain name resolution in the cluster fails.

For details about how to configure the hosts file, visit **https://coredns.io/plugins/** hosts/.





Corefile:

```
.:5353 {
  bind {$POD_IP}
  hosts {
   192.168.1.1 www.example.com
   fallthrough
  }
  cache 30
  errors
  health {$POD_IP}:8080
  kubernetes cluster.local in-addr.arpa ip6.arpa {
    pods insecure
     fallthrough in-addr.arpa ip6.arpa
  }
  loadbalance round_robin
  prometheus {$POD_IP}:9153
  forward . /etc/resolv.conf {
    policy random
  }
  reload
  ready {$POD_IP}:8081
```

Step 4 Click OK.

Step 5 Choose **ConfigMaps and Secrets** in the navigation pane, select the **kube-system** namespace, and view the ConfigMap data of **coredns** to check whether the update is successful.

----End

Configuring the Default Protocol Between the forward Plug-in and the Upstream DNS Service

- Step 1 The NodeLocal DNSCache uses TCP to communicate with the CoreDNS. The CoreDNS communicates with the upstream DNS server based on the protocol used by the request source. By default, external domain name resolution requests from service containers pass through NodeLocal DNSCache and CoreDNS in sequence, and finally request the DNS server in the VPC using TCP.
- Step 2 However, the cloud server does not support TCP. To use NodeLocal DNSCache, modify the CoreDNS configuration so that UDP is preferentially used to communicate with the upstream DNS server, preventing resolution exceptions. You are advised to use the following method to modify the CoreDNS configuration file:

The forward plug-in is used to set the upstream Nameservers DNS server. The following parameters are included:

prefer_udp: Even if a request is received through TCP, UDP must be used first.

If you want CoreDNS to preferentially use UDP to communicate with upstream systems, set the protocol in the forward plug-in to **prefer_udp**. For details about the forward plug-in, see **https://coredns.io/plugins/forward/**.

- 1. Log in to the CCE console and click the cluster name to access the cluster console.
- 2. In the navigation pane, choose Add-ons. Then, click Edit under CoreDNS.
- 3. Edit the advanced configuration under **Parameters** and modify the following content in the **plugins** field:

```
"configBlock": "prefer_udp",
  "name": "forward",
  "parameters": ". /etc/resolv.conf"
}
Corefile:
Corefile: |-
 .:5353 {
   bind {$POD_IP}
   cache 30 {
      servfail 5s
   }
   errors
    health {$POD_IP}:8080
    kubernetes cluster.local in-addr.arpa ip6.arpa {
      pods insecure
       fallthrough in-addr.arpa ip6.arpa
   }
   loadbalance round_robin
    prometheus {$POD_IP}:9153
    forward . /etc/resolv.conf {
      prefer_udp
   }
   reload
    ready {$POD_IP}:8081
```

----End

Configuring IPv6 Resolution Properly

If the IPv6 kernel module is not disabled on the Kubernetes cluster host machine, the container initiates IPv4 and IPv6 resolution at the same time by default when requesting the coredns add-on. Generally, only IPv4 addresses are used. Therefore, if you only configure **DOMAIN in IPv4 address**, the coredns add-on forwards the request to the upstream DNS server for resolution because the local configuration cannot be found. As a result, the DNS resolution request of the container slows down.

CoreDNS provides the template plug-in. After being configured, CoreDNS can immediately return an empty response to all IPv6 requests to prevent the requests from being forwarded to the upstream DNS.

- **Step 1** Log in to the CCE console and click the cluster name to access the cluster console.
- **Step 2** In the navigation pane, choose **Add-ons**. Then, click **Edit** under **CoreDNS**.

- **Step 3** Edit extended parameters in **Parameters** and add the following content to the **plugins** field.
 - AAAA indicates an IPv6 resolution request. If **NXDOMAIN** is returned in the **rcode** control response, meaning that no resolution result is returned.

For details about the template plug-in, visit **https://github.com/coredns/coredns/** tree/master/plugin/template.

```
{
    "configBlock": "rcode NXDOMAIN",
    "name": "template",
    "parameters": "ANY AAAA"
}
```

- Step 4 Click OK.
- **Step 5** In the navigation pane, choose **ConfigMaps and Secrets**. In the **kube-system** namespace, view the coredns configuration data to check whether the update is successful.

< cce-test +	Cluster: cce-test E	uster / Namespace: kube-system -	/ ConfigMaps	Quick Links Create Config	Vlap	Create from YAML
Cluster Information	ConfigMaps Secrets					
BB Resources				×		
Nodes	Delete	View ConfigMap Data				Q C
Workloads	ConfigMap Name	Key	Value		tion	
Networking	coredns				AML Up	date Delete
Storage ConfigMaps and Secrets	everest-driver-th-c		.:5353 (bind (\$POD_IP) cache 30 errorm		/AML UP	date Delete
Custom Resources	Cluster-versions		health [\$POD_IF]:8080 kubernetes cluster.local in-addr.arpa ipö.arpa { pods insecure fallabasach izzaddz arma inő arma		raml Up	date Delete
Namespaces	scheduler-config	Corefile	} loadbalance round_robin prometheus (\$POD_TP):9153		rAML Up	date Delete
Node Scaling	kube-root-ca.crt		forward . /etc/resolv.conf { policy random }		YAML Up	date Delete
Add-ons	extension-apiserv		reload template ANY AAAA (rcode NXDOMAIN }		AML Up	date Delete
Charts			ready [\$POD_IP]:8081			
Cluster Upgrade						
Logging OBT			Close			

Corresponding Corefile content:

```
.:5353 {
  bind {$POD_IP}
  cache 30
  errors
  health {$POD_IP}:8080
  kubernetes cluster.local in-addr.arpa ip6.arpa {
    pods insecure
    fallthrough in-addr.arpa ip6.arpa
  }
  loadbalance round_robin
  prometheus {$POD_IP}:9153
  forward . /etc/resolv.conf {
    policy random
  }
  reload
  template ANY AAAA {
    rcode NXDOMAIN
  }
  ready {$POD_IP}:8081
```

----End

Properly Configuring Cache Policies

If you configure CoreDNS with an upstream DNS server, you can implement a cache policy that enables CoreDNS to use the expired local cache when it is unable to access the upstream DNS server.

- **Step 1** Log in to the CCE console and click the cluster name to access the cluster console.
- **Step 2** In the navigation pane, choose **Add-ons**. Then, click **Edit** under **CoreDNS**.
- Step 3 Edit extended parameters in Parameters and modify the cache content in the plugins field. For details about how to configure the cache, see https://coredns.io/plugins/cache/.



Step 4 Click OK.

Step 5 In the navigation pane, choose **ConfigMaps and Secrets**. Select the **kube-system** namespace, view the data of the ConfigMap named **coredns** to check whether the update is successful.

Corresponding Corefile content:

```
::5353 {
    bind {$POD_IP}
    cache 30 {
        servfail 5s
        serve_stale 60s immediate
    }
    errors
    health {$POD_IP}:8080
    kubernetes cluster.local in-addr.arpa ip6.arpa {
        pods insecure
        fallthrough in-addr.arpa ip6.arpa
    }
    loadbalance round_robin
    prometheus {$POD_IP}:9153
```

```
forward . /etc/resolv.conf {
policy random
}
reload
ready {$POD_IP}:8081
```

----End

10.13 Pre-Binding Container Elastic Network Interfaces for CCE Turbo Clusters

In the Cloud Native 2.0 network model, each pod is allocated an elastic network interface or a supplementary network interface. The speed of elastic network interface creation and binding is slower than that of pod scaling, severely affecting the container startup speed in large-scale batch creation. Therefore, the Cloud Native Network 2.0 model provides the dynamic pre-binding of container elastic network interfaces to accelerate pod startup while improving IP resource utilization.

Notes and Constraints

- CCE Turbo clusters of 1.19.16-r4, 1.21.7-r0, 1.23.5-r0, 1.25.1-r0 or later support pre-binding of elastic network interfaces, global configuration at the cluster level, and custom settings at the node pool level. Custom settings of nodes out of a node pool is not supported.
- CCE Turbo clusters of 1.19.16-r2, 1.21.5-r0, 1.23.3-r0 to 1.19.16-r4, 1.21.7-r0, 1.23.5-r0 only support two parameters, **nic-minimum-target** and **nic-warm-target**, and do not support custom settings at the node pool level.
- Modify the dynamic pre-binding parameters using the console or API instead of the node annotations in the background. Otherwise, the modified annotations will be overwritten by the original values after the cluster is upgraded.
- CCE Turbo clusters of 1.19.16-r4, 1.21.7-r0, 1.23.5-r0, 1.25.1-r0, or earlier support high and low threshold for elastic network interface buffers. If users have enabled this feature, the original high and low threshold for elastic network interface pre-binding parameters is automatically converted to the dynamic pre-binding parameters of the container elastic network interfaces. If you want to modify the dynamic pre-binding parameters of the original high and low threshold to **0:0** on the cluster configuration management console.
- If the node type of a CCE Turbo node pool is BMS and the cluster version is 1.19.16-r4, 1.21.7-r0, 1.23.5-r0, or any version earlier than 1.25.1-r0, the high and low thresholds for elastic network interface pre-binding (0.3:0.6 by default) are used by default. After the cluster is upgraded, the original high and low thresholds still take effect. You are advised to convert the high and low threshold parameters to the dynamic pre-binding parameters of container elastic network interfaces on the configuration management console of the node pool and delete the high and low threshold configuration to enable the latest dynamic pre-binding parameters.
- If the node type of a CCE Turbo node pool is not BMS and the cluster version is 1.19.16-r4, 1.21.7-r0, 1.23.5-r0, or any version earlier than 1.25.1-r0, the

high and low thresholds for elastic network interface pre-binding (0.3:0.6 by default) are used by default. After the cluster is upgraded, the original high and low threshold still takes effect. If you want to enable the cluster-level global configuration, delete the annotation (**node.yangtse.io/eni-warm-policy**) of the node in the background.

How It Works

CCE Turbo provides four dynamic pre-binding parameters for container elastic network interfaces. You can properly configure the parameters based on your service requirements. (The node pool-level dynamic elastic network interface prebinding parameters take priority over the cluster-level dynamic elastic network interface pre-binding parameters.)

 Table 10-29 Parameters of the dynamic elastic network interface pre-binding policy

Parameter	Defa ult Valu e	Description	Suggestion
Minimum Number of Container ENIs Bound to a Node	10	Minimum number of container elastic network interfaces bound to a node. The parameter value must be a positive integer. The value 10 indicates that there are at least 10 container elastic network interfaces bound to a node. If the number you entered exceeds the container elastic network interface quota of the node, the elastic network interface quota of the node will be used.	Configure these parameters based on the number of pods.

Parameter	Defa ult Valu e	Description	Suggestion
Upper Limit of Pre-bound Container ENIs	0	If the number of elastic network interfaces bound to a node exceeds the value of nic-maximum-target , the system does not proactively pre- bind elastic network interfaces. If the value of this parameter is greater than or equal to the value of nic-minimum-target , the check on the maximum number of the pre- bound elastic network interfaces is enabled. Otherwise, the check is disabled. The parameter value must be a positive integer. The value 0 indicates that the check on the upper limit of pre-bound container elastic network interfaces is disabled. If the number you entered exceeds the container elastic network interface quota of the node, the elastic network interface quota of the node will be used.	Configure these parameters based on the number of pods.
Container ENIs Dynamically Pre- bound to a Node	2	Minimum number of pre-bound elastic network interfaces on a node. The value must be a number. When the value of nic-warm-target + the number of bound elastic network interfaces is greater than the value of nic-maximum-target , the system will pre-bind elastic network interfaces based on the difference between the value of nic-maximum-target and the number of bound elastic network interfaces.	Set this parameter to the number of pods that can be scaled out instantaneo usly within 10 seconds.

Parameter	Defa ult Valu e	Description	Suggestion
Threshold for Unbinding Pre- bound Container ENIs	2	 Only when the number of idle elastic network interfaces on a node minus the value of nic-warm-target is greater than the threshold, the prebound elastic network interfaces will be unbound and reclaimed. The value can only be a number. Setting a larger value of this parameter slows down the recycling of idle elastic network interfaces and accelerates pod startup. However, the IP address usage decreases, especially when IP addresses are insufficient. Therefore, exercise caution when increasing the value of this parameter. Setting a smaller value of this parameter. Setting a smaller value of this parameter accelerates the recycling of idle elastic network interfaces and improves the IP address usage. However, when a large number of pods increase instantaneously, the startup of some pods slows down. 	Set this parameter based on the difference between the number of pods that are frequently scaled on most nodes within minutes and the number of pods that are instantly scaled out on most nodes within 10 seconds.

Configuration Example

Lev el	Service Scenario	Configuration Example
Clus ter	All nodes use the c7.4xlarge.2 model (supplementary network interface quota: 128). Most nodes run about 20 pods. Most nodes can run a maximum of 60 pods. Most nodes can scale out 10 pods within 10 seconds. Most nodes frequently scale in or out 15 pods within minutes.	Cluster-level global configuration: • nic-minimum-target: 20 • nic-maximum-target: 60 • nic-warm-target: 10 • nic-max-above-warm- target: 5

Lev el	Service Scenario	Configuration Example
Nod e pool	A node pool that uses the c7.8xlarge.2 high-specification model is created in the cluster. (supplementary network interface quota: 256) Most nodes run about 100 pods. Most nodes can run a maximum of 128 pods. Most nodes can scale out 10 pods within 10 seconds. Most nodes frequently scale in or out 12 pods within minutes.	Custom settings at the node pool level: • nic-minimum-target: 100 • nic-maximum-target: 120 • nic-warm-target: 10 • nic-max-above-warm- target: 2

NOTE

Pods using HostNetwork are excluded.

Cluster-level Global Configuration

Step 1 Log in to the CCE console. In the navigation pane, choose **Clusters**.

Step 2 Click ******* next to the target cluster and choose **Manage**.

Figure 10-68 Managing a cluster

Monitor	Kubectl Shell
Pay-per-use Created on Jai	Create Node Pool
	Create Node
	Workloads
	View Event
	Change Billing Mode
	Manage
	Hibernate
	Modify Specifications
	Delete Cluster

Step 3 In the window that slides out from the right, click **Networking Components**. For details about the parameter configurations, see **Configuration Example**.

Manage Component (Cluster			
Customize the settings of Kubernetes native components or CCE-developed components to satisfy your demands. Details about the parameters: Configuring Clusters Components			
✓ kube-apiserver			
 ✓ Scheduler 			
✓ kube-controller-manager			
Networking Components			
Best Practices for Configuring NIC Dynamic Preheating			
The minimum number of network cards bound to the container at the cluster level (nic-minimum-target) ③			
Quantity Percentage - 10 + pods			
${igodoldsymbol{\widehat{S}}}$ Set to the number of pods running on most nodes.			
Cluster-level node preheating container NIC upper limit check value (nic-maximum-target)			
Quantity Percentage - 0 + pods			
\bigotimes Set to the most number of pods that can run on most nodes.			
Number of NICs for dynamically warming up containers at the cluster level (nic-warm-target) 🧿			
Set to the number of pods that can be scaled out within 10 seconds on most nodes.			
Cluster-level node warm-up container NIC recycling threshold (nic-max-above-warm-target) 🧿			
$\left(\begin{array}{c c} - & 2 & + \end{array} \right)$			
\bigotimes Set to the number of pods that are frequently scaled within minutes on most nodes.			
prebound-subeni-percentage (prebound-subeni-percentage)			
0:0			

Step 4 After the configuration is complete, click **OK**. Wait for about 10 seconds for the configuration to take effect.

----End

Custom Settings at the Node Pool Level

- **Step 1** Log in to the CCE console.
- **Step 2** Click the cluster name to access the cluster console, choose **Nodes** in the navigation pane, and click the **Node Pools** tab.
- **Step 3** Locate the row containing the target node pool and click **Manage**.
- **Step 4** In the window that slides out from the right, click **Networking Components** and enable node pool container elastic network interface pre-binding. For details about the parameter configurations, see **Configuration Example**.



Step 5 After the configuration is complete, click **OK**. Wait for about 10 seconds for the configuration to take effect.

----End

10.14 Connecting a Cluster to the Peer VPC Through an Enterprise Router

Application Scenarios

An enterprise router connects virtual private clouds (VPCs) and on-premises networks to build a central hub network and implement communication between VPCs in the same region. It has high specifications, provides high bandwidth, and delivers high performance. With the enterprise routers, CCE clusters in different VPCs can access each other.

Clusters in different VPCs cannot communicate with VMs in the peer VPCs within a short period of time after containers are created in the clusters. To solve this problem, attach the peer VPCs to the enterprise routers. In a CCE Turbo cluster, configure parameters to delay the pod startup to solve this problem. For details, see **Configuring Parameters to Delay the Pod Startup in a CCE Turbo Cluster**.

Network Planning

Before attaching VPCs to the enterprise routers, determine the VPC CIDR blocks and the enterprise routers' route table. Ensure that the following requirements are met.
Resource	Description
VPC	 The VPC CIDR blocks cannot overlap. The CIDR blocks of the VPCs are propagated to the enterprise routers' route tables. These CIDR blocks cannot be modified. Overlapping CIDR blocks may cause route conflicts. Additionally, the container CIDR blocks cannot conflict with the node CIDR blocks in the peer VPCs. Otherwise, the network is unavailable. If your existing VPCs have overlapping CIDR blocks, manually add static routes to the route tables of the enterprise routers.
	The destination can be the VPC subnet CIDR blocks or smaller ones.
Enterprise router	After Default Route Table Association and Default Route Table Propagation are enabled and a VPC attachment is created, the system automatically:
	• Associates the VPC attachment with the default route table of the enterprise router.
	• Propagates the VPC attachment to the default route table of the enterprise router. The route table automatically learns the VPC CIDR blocks.

For details, see Step 1: Plan Networks and Resources.

Creating an Enterprise Router

- **Step 1** Log in to the management console.
- **Step 2** Click ^I in the upper left corner and select the desired region and project.
- **Step 3** Choose **Service List > Networking > Enterprise Router**.
- **Step 4** Enter the **Enterprise Router** page.
- **Step 5** Click **Create Enterprise Router** in the upper right corner.
- **Step 6** Enter the **Create Enterprise Router** page.
- **Step 7** Configure basic information following instructions. For details about the parameter settings, see **Table 10-30**.

Create Enterprise Router	0
* Region	•
	Regions are geographic areas isolated from each other. Resources are region-specific and cannot be used across regions through internal network connections. For low network latency and quick resource access, select the nearest region.
* AZ 🕐	AZ2 () AZ3 () ·
	ceest, mv Aza in tuningure duive-sturie tepunyment für high availability.
* Name	er-ab56
* ASN (?)	64512
Default Route Table Association	Enable 🕜
Default Route Table Propagation	Enable 🔞
Auto Accept Shared Attachments	Enable 🕐
* Enterprise Project	-Select- C C Create Enterprise Project
Tag	It is recommended that you use TMS's predefined tag function to add the same tag to different cloud resources. View predefined tags: C
	Tag key Tag value
	You can add 10 more tags.

Figure 10-69 Creating an enterprise router

Parameter	Setting	Example Value
Region	Select the region nearest to your target users. Once the enterprise router is created, the region cannot be changed.	CN-Hong Kong
AZ	Select two AZs to deploy your enterprise router. You can change them after the enterprise router is created.	AZ1 AZ2
Name	Specify the enterprise router name. You can change it after the enterprise router is created.	er-test-01
ASN	Enter an ASN based on your network plan. It cannot be changed after the enterprise router is created.	64800
Default Route Table Association	If you select this option, you do not need to create route tables or associations. You can change your option after the enterprise router is created.	Enable
Default Route Table Propagation	If you select this option, you do not need to create route tables, propagations, or routes. You can change your option after the enterprise router is created.	Enable

Parameter	Setting	Example Value
Auto Accept Shared Attachment s	If you do not select this option, you must accept the requests for creating attachments to this enterprise router from other users with whom this enterprise router is shared.	Disable
Enterprise Project	Select an enterprise project for the enterprise router. You can change it after the enterprise router is created.	default
Tag	Add tags to help you identify your enterprise router. You can change them after the enterprise router is created.	Tag key: test Tag value: 01
Description	Provide supplementary information about the enterprise router. You can change it after the enterprise router is created.	-

Step 8 Click Create Now.

- **Step 9** Check the information on the page displayed and click **Submit**. Back to the enterprise router list.
- **Step 10** View the enterprise router's status. If its status changes from **Creating** to **Normal**, the enterprise router is created.

Figure 10-70 Enterprise router created

. In 1963, 1984	e			Manage Attachment	M	inage Route Table	Manage Sharing	
Status	Normal	ID		Attachr	nents (Used	к а		
AZ 🕐	AZ1,AZ3	ASN	64512		VPG	Peering		
Default Route Table Association	Enabled	Default Route Table Propagation	Enabled		1	2		
Auto Accept Shared Attachments	Disabled	Enterprise Project	default					
Created	Apr 20, 2023 17:17:36 GMT+08:00							

----End

Creating a VPC Attachment to an Enterprise Router

Step 1 Log in to the management console.

- **Step 2** Click O in the upper left corner and select the desired region and project.
- **Step 3** Choose **Service List > Networking > Enterprise Router**.
- **Step 4** Search for the target enterprise router by name.
- **Step 5** Perform either of the following operations to go to the **Attachments** tab:
 - Locate the pane containing the target enterprise router and click **Manage Attachment**.
 - Click the name of the enterprise router and click the **Attachments** tab. For details, see **Adding VPC Attachments to an Enterprise Router**.

Step 6 Click **Create Attachment** to go to the corresponding page.

Step 7 Configure the parameters following instructions. For details, see **Table 10-31**.

< Create Attachment	
* Name	er-attach-01
* Attachment Type	VPC
* Attached Resource	vpc v Q Select a subnet for connecting a VPC to the enterprise router. subnet v Q The subnet needs enough IP addresses for both the system and the router, so select a mask of /28 or smaller.
* Auto Add Routes	Automatically add routes (with this enterprise router as the next hop and 10.0.0.0/8, 172.16.0.0/12, and 192.168.0.0/16 as the destinations) to all route tables of the selected VPC.
Advanced Settings \vee	Description Tag

Figure 10-71 Creating an attachment

Table 10-31 Parameter description

Parameter	Setting	Example Value
Name	Specify the name of the VPC attachment. You can change it after the attachment is created.	er-attach-01
Attachment Type	Select VPC . The type cannot be changed after the attachment is created.	VPC
Attached Resource	 Select the VPC to be attached to the enterprise router from the drop-down list. The VPC cannot be changed after the attachment is created. Select the subnet to be attached to the 	 VPC: vpc- demo-01 Subnet: subnet- demo-01
	enterprise router from the drop-down list. The subnet cannot be changed after the attachment is created.	

Parameter	Setting	Example Value
Auto Add Routes	• If you enable Auto Add Routes when creating a VPC attachment, you do not need to manually add static routes to the VPC route table. Instead, the system automatically adds routes (with this enterprise router as the next hop and 10.0.0.0/8, 172.16.0.0/12, and 192.168.0.0/16 as the destinations) to all route tables of the VPC.	Enable
	 If an existing route in the VPC route tables has a destination to 10.0.0/8, 172.16.0.0/12, or 192.168.0.0/16, the routes will fail to be added. In this case, do not enable Auto Add Routes. After the attachment is created, manually add routes. 	
	• Do not set the destination of a route (with an enterprise router as the next hop) to 0.0.0.0/0 in the VPC route table. If an ECS in the VPC has an EIP bound, the VPC route table will have a policy-based route with 0.0.0.0/0 as the destination, which has a higher priority than the route with the enterprise router as the next hop. In this case, traffic is forwarded to the EIP and cannot reach the enterprise router.	
Description	Provide supplementary description about the attachment. You can change it after the attachment is created.	-
Тад	Add tags to help you identify your attachment. You can change them after the attachment is created.	Tag key: test Tag value: 01

Step 8 Click **Create Now**. If the status changes from **Creating** to **Normal**, the attachment is created. Repeat steps **Step 6** to **Step 7** to attach other VPCs.

Figure 10-72 Attaching a VPC to the enterprise router

There are some restrictions when	ou attach VPCs to an enterprise rout	ter to build a network. To make sure network con	nnectivity is not interrupted, see Enterprise Ro	uter Networking Schemes 🗹.			
Dearn now 🕑 to crease an attache	wirz,						
reate Attachment							
Select a property or enter a keywork							
mellD Θ	Status Θ	Attachment Type 🖯	Attached Resource Θ	Account/Project ID 🖯	Association Route Table Θ	Description O	Operation

----End

Verifying the Network

- **Step 1** Log in to the CCE console and search for the CCE cluster that has been attached to the VPC.
- **Step 2** Click the name of the target cluster. In the navigation pane, choose **Nodes** to view the IP address of the node.

Figure 10-73 Viewing the IP address of the node on the CCE console



- **Step 3** Log in to the node. For details, see **Logging In to a Node**. In this example, use VNC provided on the management console to log in to the ECS.
- **Step 4** Run the following command on the ECS console:

ping {ECS address}

Take the cluster in **vpc-ER-demo2** as an example. Log in to the **er-demo2-04260** node and access the **er-demo1-61379** node in the cluster in **vpc-ER-demo1**. The IP address of the node is **192.168.0.131**.

ping 192.168.0.131

If the following information is displayed, the network is accessible.

Figure 10-74 Viewing the command output

```
[root@er-demo-91067 "]# ping 192.168.0.131
PING 192.168.0.131 (192.168.0.131) 56(84) bytes of data.
64 bytes from 192.168.0.131: icmp_seq=1 ttl=63 time=0.417 ms
64 bytes from 192.168.0.131: icmp_seq=2 ttl=63 time=0.399 ms
64 bytes from 192.168.0.131: icmp_seq=3 ttl=63 time=0.356 ms
64 bytes from 192.168.0.131: icmp_seq=4 ttl=63 time=0.340 ms
64 bytes from 192.168.0.131: icmp_seq=5 ttl=63 time=0.323 ms
64 bytes from 192.168.0.131: icmp_seq=5 ttl=63 time=0.323 ms
64 bytes from 192.168.0.131: icmp_seq=6 ttl=63 time=0.354 ms
```

Step 5 Repeat the preceding steps to verify the communication between nodes.

If a node using the peer VPC cannot be pinged, check:

- 1. Whether the security group rules of the node allow the ICMP protocol.
- 2. Whether a CIDR block conflict occurs in the VPC route table. Note that the container CIDR blocks cannot conflict with the default CIDR blocks of the enterprise routers. For details, see **Network Planning**.

Figure 10-75 Viewing the VPC route table

Routes							
Delete Add Route Re	Pplicate Route QLearn how to configure in	outes.					C
Q Specify filter criteria.							
Destination	IP Addresses	Next Hop Type	Next Hop	Туре	Description	Operation	
Local	4	Local	Local	System	Default route that enables insta	Modity Delete	
10.0.0.00	1	Enterprise Router	er-100-000	Custom		Modity Delete	
172.16.0.0/12	1	Enterprise Router	er (millingen af	Custom		Modify Delote	
192.168.0.0/16	1	Enterprise Router	or -	Custom		Modify Delete	

----End

10.15 Accessing an IP Address Outside of a Cluster That Uses a VPC Network by Using Source Pod IP Addresses Within the Cluster

In a CCE cluster that uses a VPC network, when pods try to communicate with external systems, CCE automatically translates the source IP addresses of the pods into the IP addresses of the nodes that are running them. This allows pods to communicate with external systems using the node IP addresses. This process is known as pod IP address masquerading or Source Network Address Translation (SNAT).

You are allowed to configure private CIDR blocks for your clusters using the **nonMasqueradeCIDRs** parameter. If a pod tries to access a private CIDR block, the source node will not perform NAT on the pod IP address. Instead, the VPC route table can directly send the pod data packet to the destination, which means, the pod IP address is directly used to communicate with the private CIDR block in the cluster. For details about how to access an external IP address from a pod, see **Accessing an External Network from a Pod**.



Figure 10-76 Pod IP address translation

Prerequisites

You have a cluster that uses the VPC network and whose version is v1.23.14-r0, v1.25.9-r0, v1.27.6-r0, v1.28.4-r0, or later.

Default Non-Masqueraded CIDR Block Settings in a CCE Cluster

By default, CCE uses the following well-known private CIDR blocks as nonmasqueraded CIDR blocks in each cluster:

- 10.0.0/8
- 172.16.0.0/12
- 192.168.0.0/16

Additionally, in a CCE cluster that uses a secondary VPC CIDR block, adding or resetting a node will automatically include the secondary CIDR block in the non-masqueraded CIDR blocks.

This means that when a pod communicates with external resources and accesses these CIDR blocks, the source IP address of the data packet remains unchanged and is not translated into the node IP address.

Scenarios Where the Default Non-Masqueraded CIDR Blocks Do Not Fit

The default non-masqueraded CIDR block settings in CCE clusters apply to typical scenarios, but in certain specific scenarios, these default settings may not be sufficient to meet user requirements. The following shows typical examples:

• Cross-node access to pods in a cluster

When a node in a Kubernetes cluster needs to access a pod on another node, the response data packet sent from the pod is automatically subject to SNAT. This changes the source IP address from the pod IP address to the IP address of the node that runs the pod. However, this automatic IP address translation can sometimes lead to communication issues, making cross-node access impossible.

To enable a node to access pods on other nodes, you can add the CIDR block of the subnet where the node is located to the **nonMasqueradeCIDRs** parameter. This will skip SNAT and allow the original IP addresses of pods on these nodes to be retained.

• Access from other resources in the same VPC as a cluster to pods in the cluster

In certain scenarios, it may be necessary to access the original IP addresses of pods on different nodes in a CCE cluster directly from other resources (such as ECSs) in the same VPC as the cluster. However, with SNAT enabled by default, the source IP addresses of the data packets are replaced with the IP addresses of the nodes that run these pods when the data packets pass through the nodes. This makes it difficult for these resources to access pods directly.

To enable direct access from resources in the same VPC as the cluster to pods, you can add the CIDR blocks of the subnets where these resources are located to the **nonMasqueradeCIDRs** parameter. This will skip SNAT and ensure that the source IP addresses of the data packets remain the same as the original IP addresses of pods.

Precautions

If a security group or ACL is configured for a cloud service and only the IP address of the node where the pod runs is allowed to access the service, SNAT is required to translate the pod IP address into the node IP address for successful access. As a result, the CIDR block of the subnet where the server is located cannot be added to the **nonMasqueradeCIDRs** configuration.

The default setting of pod IP address masquerading (SNAT) is usually sufficient. However, if you need to retain the original IP addresses of pods in specific scenarios, you can configure the **nonMasqueradeCIDRs** parameter.

Before doing so, make sure you have evaluated your application scenario and understood the potential risks of improper configuration, because it may block access within clusters. If you are unsure whether to configure this parameter, it is recommended that you keep the default settings and adjust the configuration later once the requirement is clarified.

Procedure

To reserve the source IP address of a pod when the pod accesses a CIDR block, you can configure **nonMasqueradeCIDRs** to specify the CIDR block that does not need to be masqueraded.

- **Step 1** Log in to the CCE console and click the cluster name to access the cluster console.
- **Step 2** In the navigation pane, choose **Settings** and click the **Network** tab.

- **Step 3** Modify the range of the CIDR blocks for non-masquerading access to preserve the source pod IP address when accessing a specified CIDR block. Make sure the parameter configuration complies with the following rules:
 - Each CIDR block must comply with the CIDR format and must be a valid IPv4 CIDR block.

Example of a correct CIDR block: **192.168.1.0/24**

Example of an incorrect CIDR block: **192.168.1.1/24** (incompliant with the CIDR format)

- The CIDR blocks you configured do not overlap with each other. Example of correct CIDR blocks: **192.168.1.0/24** and **192.168.2.0/24** Example of incorrect CIDR blocks: **192.168.1.0/24** and **192.168.1.128/25** (The two CIDR blocks overlap.)
- The **nonMasqueradeCIDRs** parameter must contain all destination CIDR blocks that you want them to use the original pod IP addresses for communications.
- **Step 4** After the modification, click **Confirm configuration**. The setting takes effect within 1 minute.

----End

10.16 Using Kmesh on Huawei Cloud CCE

Kmesh is a high-performance service mesh data plane software built on eBPF and programmable kernels. By offloading traffic management to the kernel, Kmesh eliminates the need for proxy software during service communication within the meshes. This approach significantly shortens the traffic forwarding path and enhances the forwarding performance of service access.



Kmesh's dual-engine mode leverages eBPF for traffic interception in the kernel space and uses waypoint proxies to manage complex Layer 7 traffic. This design separates Layer 4 and Layer 7 governance between the kernel space (eBPF) and the user space (Waypoint). The dual-engine mode reduces latency by 30% compared to Istio ambient mesh. Additionally, unlike the kernel-native mode, it



does not require kernel enhancements, making it more flexible and broadly applicable.

This section describes how to deploy the Kmesh dual-engine mode in a CCE cluster to meet your requirements for using Kmesh on CCE.

Prerequisites

• A CCE cluster of v1.28 or later is added, and there are nodes in the cluster. For details about how to buy a cluster and create nodes in it, see **Buying a CCE Standard/Turbo Cluster** and **Creating a Node**.

A CCE standard cluster using a VPC or tunnel network is used as an example. The kernel of a newly created node must be 5.10 or later, so Ubuntu 22.04 is used in this example.

 You have a VM that can access the Internet. It can access the cluster using kubectl and has Helm installed.

Step 1: Install the Service Mesh Control Plane

Kmesh serves as the data plane for service meshes and communicates with the control plane using the xDS protocol. Istiod is used as the Kmesh control plane, though any control plane supporting the xDS protocol can theoretically function as Kmesh's control plane. For details, see Kmesh Quick Start Guide.

- **Step 1** Log in to the VM where kubectl is installed and add an Istio repository. helm repo add istio https://istio-release.storage.googleapis.com/charts helm repo update
- Step 2 Install the Istio base chart.

kubectl create namespace istio-system helm install istio-base istio/base -n istio-system

Step 3 Install the Istiod chart.

helm install istiod istio/istiod --namespace istio-system --set pilot.env.PILOT_ENABLE_AMBIENT=true

The **pilot.env.PILOT_ENABLE_AMBIENT=true** parameter is required to enable Kmesh to establish a gRPC connection with Istiod.

Check the result. kubectl get svc -n istio-system | grep istiod

Information similar to the following is displayed:

istiod ClusterIP 10.247.51.34 <none> 15010/TCP,15012/TCP,443/TCP,15014/TCP 4h8m

Step 4 Install the Kubernetes Gateway API CRD.

kubectl get crd gateways.gateway.networking.k8s.io &> /dev/null || \
 { kubectl kustomize "github.com/kubernetes-sigs/gateway-api/config/crd/experimental?
ref=444631bfe06f3bcca5d0eadf1857eac1d369421d" | kubectl apply -f -; }

----End

Step 2: Install Kmesh

Step 1 Download the Kmesh Helm package and kmeshctl to a local PC.

download helm package curl -L -o kmesh-pakcage.tgz https://github.com/kmesh-net/kmesh/releases/download/v1.0.0/kmesh-helmv1.0.0.tgz tar -zxvf kmesh-pakcage.tgz

download kmeshctl curl -L -o kmeshctl https://github.com/kmesh-net/kmesh/releases/download/v1.0.0/kmeshctl-linux-amd64

- **Step 2** Install Kmesh using Helm. helm install kmesh ./kmesh-helm -n kmesh-system --create-namespace
- **Step 3** After the installation command is executed, check the Kmesh startup status. kubectl get pod -A | grep kmesh

Information similar to the following is displayed:

kmesh-system kmesh-dc5l8 1/1 Running 0 4h25m

Step 4 Check the Kmesh running status.

kubectl logs -n kmesh-system kmesh-dc5l8

Information similar to the following is displayed:

```
time="2025-04-15T13:17:12Z" level=info msg="FLAG: --bpf-fs-path=\"/sys/fs/bpf\"" subsys=manager
time="2025-04-15T13:17:12Z" level=info msg="FLAG: --cgroup2-path=\"/mnt/kmesh_cgroup2\"
subsys=manager
time="2025-04-15T13:17:12Z" level=info msg="FLAG: --cni-etc-path=\"/etc/cni/net.d\"" subsys=manager
time="2025-04-15T13:17:12Z" level=info msg="FLAG: --conflist-name=\"\"" subsys=manager
time="2025-04-15T13:17:12Z" level=info msg="FLAG: --enable-bypass=\"false\"" subsys=manager time="2025-04-15T13:17:12Z" level=info msg="FLAG: --enable-ipsec=\"false\"" subsys=manager
time="2025-04-15T13:17:12Z" level=info msg="FLAG: --enable-mda=\"false\"" subsys=manager
time="2025-04-15T13:17:12Z" level=info msg="FLAG: --enable-secret-manager=\"false\"" subsys=manager
time="2025-04-15T13:17:12Z" level=info msg="FLAG: --help=\"false\"" subsys=manager
time="2025-04-15T13:17:12Z" level=info msg="FLAG: --mode=\"dual-engine\"" subsys=manager
time="2025-04-15T13:17:12Z" level=info msg="FLAG: --monitoring=\"true\"" subsys=manager
time="2025-04-15T13:17:12Z" level=info msg="FLAG: --plugin-cni-chained=\"true\"" subsys=manager time="2025-04-15T13:17:12Z" level=info msg="FLAG: --profiling=\"false\"" subsys=manager
time="2025-04-15T13:17:12Z" level=info msg="kmesh start with Normal" subsys=bpf
time="2025-04-15T13:17:14Z" level=info msg="bpf loader start successfully" subsys=manager
time="2025-04-15T13:17:14Z" level=info msg="start kmesh manage controller successfully"
subsys=controller
time="2025-04-15T13:17:14Z" level=info msg="proxy ztunnel~192.168.1.174~kmesh-dc5l8.kmesh-
system~kmesh-system.svc.cluster.local connect to discovery address istiod.istio-system.svc:15012"
subsys=controller/config
time="2025-04-15T13:17:14Z" level=info msg="controller start successfully" subsys=manager
time="2025-04-15T13:17:14Z" level=info msg="start write CNI config" subsys="cni installer"
time="2025-04-15T13:17:14Z" level=info msg="kmesh cni use chained\n" subsys="cni installer"
time="2025-04-15T13:17:14Z" level=info msg="reload authz config from last epoch"
subsys=workload_controller
time="2025-04-15T13:17:14Z" level=info msg="Copied /usr/bin/kmesh-cni to /opt/cni/bin." subsys="cni
```

```
installer"
time="2025-04-15T13:17:14Z" level=info msg="wrote kubeconfig file /etc/cni/net.d/kmesh-cni-kubeconfig"
subsys="cni installer"
time="2025-04-15T13:17:14Z" level=info msg="cni config file: /etc/cni/net.d/10-kindnet.conflist" subsys="cni
installer"
time="2025-04-15T13:17:14Z" level=info msg="start cni successfully" subsys=manager
time="2025-04-15T13:17:14Z" level=info msg="start watching file /var/run/secrets/kubernetes.io/
serviceaccount/token" subsys="cni installer"
```

----End

Step 3: Use Kmesh

Step 1 To use Kmesh to manage namespaces, add the **istio.io/dataplane-mode=Kmesh** label to the **default** namespace.

kubectl label namespace default istio.io/dataplane-mode=Kmesh

Step 2 View the namespaces managed by Kmesh. kubectl get namespace -L istio.io/dataplane-mode

Information similar to the following is displayed:

NAME STATUS AGE DATAPLANE-MODE default Active 92d Kmesh

Step 3 Deploy the Bookinfo sample application.

kubectl apply -f https://raw.githubusercontent.com/istio/istio/release-1.21/samples/bookinfo/platform/kube/bookinfo.yaml

- **Step 4** Deploy the sleep application as the cURL client. kubectl apply -f https://raw.githubusercontent.com/istio/istio/release-1.21/samples/sleep/sleep.yaml
- Step 5 Install a waypoint. Waypoints can be applied to namespaces, Services, and pods, offering flexibility in their configuration. It is possible to install multiple waypoints at different granularity within a single namespace. To use a waypoint, you must add the istio.io/use-waypoint label and set its value to the name of the waypoint. Additionally, you can specify a custom waypoint image using the -- image option, with the default image being ghcr.io/kmesh-net/waypoint: {VERSION}. In this example, a waypoint is configured for a namespace. For details, see Install Waypoint.

kmeshctl waypoint apply -n default --enroll-namespace --image ghcr.io/kmesh-net/waypoint:latest

Information similar to the following is displayed:

waypoint default/waypoint applied namespace default labels with "istio.io/use-waypoint: waypoint"

Step 6 View the pods.

kubectl get pod

Information similar to the following is displayed:

READY STATUS RESTARTS AGE NAMF details-v1-86545f5dfb-p6kgw 1/1 Running 0 22m 1/1 productpage-v1-7c74cbdbcc-tnk7w Running 0 22m ratings-v1-57544668d4-vx9h2 1/1 Running 0 22m reviews-v1-5f58978c56-xqjtz Running 0 1/1 22m reviews-v2-7bd564ffc6-pdskr 22m 1/1 Running 0 reviews-v3-7dfb7c4b64-bzjm8 1/1 Running 0 22m 18m sleep-5fcd8fd6c8-9wj7l 1/1 Running 0 waypoint-75686498f6-ksrnh 1/1 Running 0 9m15s

Step 7 Verify that Bookinfo works as expected.

kubectl exec deploy/sleep -- curl -s http://productpage:9080/ | grep -o "<title>.*</title>"

Information similar to the following is displayed:

<title>Simple Bookstore App</title>

----End

Step 4: Configure Weight-based Routing

Step 1 Configure weight-based routing to direct 90% of requests to **reviews-v1** and 10% to **reviews-v2**.

kubectl apply -f -<<EOF apiVersion: networking.istio.io/v1alpha3 kind: VirtualService metadata: name: reviews spec: hosts - reviews http: - route: - destination: host: reviews subset: v1 weight: 90 - destination: host: reviews subset: v2 weight: 10 apiVersion: networking.istio.io/v1alpha3 kind: DestinationRule metadata: name: reviews spec: host: reviews trafficPolicy: loadBalancer: simple: RANDOM subsets: - name: v1 labels: version: v1 - name: v2 labels: version: v2 - name: v3 labels: version: v3 EOF

Step 2 Verify that approximately 90% of the traffic is routed to **reviews-v1**.

kubectl exec deploy/sleep -- sh -c "for i in \\$(seq 1 100); do curl -s http://productpage:9080/productpage | grep reviews-v.-; done"

Information similar to the following is displayed:

```
<u>reviews-v1-5f58978c56-xqjtz</u>
<u>reviews-v1-5f58978c56-xqjtz</u>
<u>reviews-v1-5f58978c56-xqjtz</u>
<u>reviews-v1-5f58978c56-xqjtz</u>
<u>reviews-v1-5f58978c56-xqjtz</u>
<u>reviews-v1-5f58978c56-xqjtz</u>
<u>reviews-v1-5f58978c56-xqjtz</u>
<u>reviews-v1-5f58978c56-xqjtz</u>
```

<u>reviews-v1-5f58978c56-xqjtz</u>
<u>reviews-v1-5f58978c56-xqjtz</u>
<u>reviews-v1-5f58978c56-xqjtz</u>
<u>reviews-v2-7bd564ffc6-pdskr</u>
<u>reviews-v1-5f58978c56-xqjtz</u>
<u>reviews-v1-5f58978c56-xqjtz</u>
<u>reviews-v1-5f58978c56-xqjtz</u>
<u>reviews-v1-5f58978c56-xqjtz</u>
<u>reviews-v2-7bd564ffc6-pdskr</u>
<u>reviews-v1-5f58978c56-xqjtz</u>
<u>reviews-v1-5f58978c56-xqjtz</u>
<u>reviews-v1-5f58978c56-xqjtz</u>

In this configuration, the **default** namespace is managed by Kmesh, with a waypoint proxy deployed for the namespace. All traffic sent to the reviews service is forwarded by Kmesh to the waypoint, which routes 90% of the requests to **reviews-v1** and 10% to **reviews-v2** based on the configured weights.

----End

11 Storage

11.1 Expanding the Storage Space

The storage classes that can be expanded for CCE nodes are as follows:

Туре	Name	Purpose	Capacity Expansion Method
Node disk	System disk	A disk attached to a node for installing the operating system	Expanding System Disk Capacity
	Data disk	The first data disk attached to a node for container engine and kubelet	 Expanding the Container Engine Capacity Expanding the kubelet Capacity Expanding Capacity of the Disk Shared by Container Engine and kubelet
Contain er storage	Pod containe r space	The base size of a container, which is, the upper limit of the disk space occupied by each pod (including the storage space occupied by container images)	Expanding the Capacity of a Data Disk Used by Pod (basesize)
	PVC	Storage resources mounted to the containers	Expanding a PVC

Table 11-1 Capacity expansion methods

Expanding System Disk Capacity

EulerOS 2.9 is used as the sample OS. There is only one partition (/dev/vda1) with a capacity of 50 GiB in the system disk /dev/vda, and then 50 GiB is added to the system disk. In this example, the additional 50 GiB is allocated to the existing /dev/vda1 partition.

Step 1 Expand the system disk capacity on the EVS console. For details, see Expanding EVS Disk Capacity.

Only the storage capacity of the EVS disk is expanded. You also need to perform the following operations to expand the partition and file system.

Step 2 Log in to the node and run the **growpart** command to check whether growpart has been installed.

If the tool operation guide is displayed, the growpart has been installed. Otherwise, run the following command to install growpart:

yum install cloud-utils-growpart

Step 3 Run the following command to view the total capacity of the system disk **/dev/vda**:

fdisk -l

Step 4

If the following information is displayed, the total capacity of /dev/vda is 100 GiB.

[root@test-48162 ~]# fdisk -l Disk /dev/vda: 100 GiB, 107374182400 bytes, 209715200 sectors Units: sectors of 1 * 512 = 512 bytes Sector size (logical/physical): 512 bytes / 512 bytes I/O size (minimum/optimal): 512 bytes / 512 bytes Disklabel type: dos Disk identifier: 0x78d88f0b
Device Boot Start End Sectors Size ld Type /dev/vda1 * 2048 104857566 104855519 50G 83 Linux
Disk /dev/vdb: 100 GiB, 107374182400 bytes, 209715200 sectors Units: sectors of 1 * 512 = 512 bytes Sector size (logical/physical): 512 bytes / 512 bytes I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk /dev/mapper/vgpaas-dockersys: 90 GiB, 96632569856 bytes, 188735488 sectors Units: sectors of 1 * 512 = 512 bytes Sector size (logical/physical): 512 bytes / 512 bytes I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk /dev/mapper/vgpaas-kubernetes: 10 GiB, 10733223936 bytes, 20963328 sectors Units: sectors of 1 * 512 = 512 bytes Sector size (logical/physical): 512 bytes / 512 bytes I/O size (minimum/optimal): 512 bytes / 512 bytes
Run the following command to check the capacity of the system disk partition /dev/vda1 :
Information similar to the following is displayed:
[root@test-48162 ~]# df -TH Filesystem Type Size Used Avail Use% Mounted on

] <i>"</i> ur rrr		
Filesystem	Туре	Size	Used Avail Use% Mounted on
devtmpfs	devtm	pfs 1.8	G 0 1.8G 0% /dev
tmpfs	tmpfs	1.8G	0 1.8G 0% /dev/shm
tmpfs	tmpfs	1.8G	13M 1.8G 1% /run
tmpfs	tmpfs	1.8G	0 1.8G 0% /sys/fs/cgroup

/dev/vda1	ext4	53G	3.3G	47G	7	% /	
tmpfs	tmpfs	1.8G	75M	1.8G	5%	6 /tmp	0
/dev/mapper/vgpaas-	dockersys	ext4	95	G 1.3	G	89G	2% /var/lib/docker
/dev/mapper/vgpaas-l	kubernete	s ext4	11	IG 39	9M	10G	1% /mnt/paas/kubernetes/kubelet

Step 5 Run the following command to extend the partition using growpart:

growpart System disk Partition number

The partition number is **1** because there is only one **/dev/vda1** partition in the system disk, as shown in the following command:

growpart /dev/vda 1

Information similar to the following is displayed:

CHANGED: partition=1 start=2048 old: size=104855519 end=104857567 new: size=209713119 end=209715167

Step 6 Run the following command to extend the file system:

resize2fs Disk partition

An example command is as follows:

resize2fs /dev/vda1

Information similar to the following is displayed:

resize2fs 1.45.6 (20-Mar-2020) Filesystem at /dev/vda1 is mounted on /; on-line resizing required old_desc_blocks = 7, new_desc_blocks = 13 The filesystem on /dev/vda1 is now 26214139 (4k) blocks long.

Step 7 Run the following command to view the new capacity of the **/dev/vda1** partition: df -TH

Information similar to the following is displayed:

[root@test-48162 ~]	# df -TH	
Filesystem	Туре	Size Used Avail Use% Mounted on
devtmpfs	devtm	pfs 1.8G 0 1.8G 0% /dev
tmpfs	tmpfs	1.8G 0 1.8G 0% /dev/shm
tmpfs	tmpfs	1.8G 13M 1.8G 1% /run
tmpfs	tmpfs	1.8G 0 1.8G 0% /sys/fs/cgroup
/dev/vda1	ext4	106G 3.3G 98G 4% /
tmpfs	tmpfs	1.8G 75M 1.8G 5% /tmp
/dev/mapper/vgpaas	-dockersys	s ext4 95G 1.3G 89G 2% /var/lib/docker
/dev/mapper/vgpaas	-kubernet	es ext4 11G 39M 10G 1% /mnt/paas/kubernetes/kubelet

Step 8 Log in to the CCE console and click the cluster name to access the cluster console. In the navigation pane, choose Nodes. In the right pane, click the Nodes tab, locate the row containing the target node, and choose More > Sync Server Data in the Operation column.

----End

Expanding Data Disk Capacity

The first data disk of a CCE node is composed of container engine and kubelet space by default. If either of them reaches full capacity, you can expand the disk space as needed.

In clusters of v1.21.10-r0, v1.23.8-r0, v1.25.3-r0, and later, CCE enables container engine (Docker/containerd) and kubelet to share the space of the first data disk. If the shared disk space is insufficient, you can expand it.

Expanding the Container Engine Capacity

The available container engine space affects image pulls and container startup and running. This section uses containerd as an example to describe how to expand the container engine capacity.

Step 1 Expand the capacity of a data disk on the EVS console. For details, see **Expanding EVS Disk Capacity**.

Only the storage capacity of EVS disks can be expanded. You need to perform the following operations to expand the capacity of logical volumes and file systems.

- **Step 2** Log in to the CCE console and click the cluster name to access the cluster console. In the navigation pane, choose Nodes. In the right pane, click the Nodes tab, locate the row containing the target node, and choose **More** > Sync Server Data in the **Operation** column.
- **Step 3** Log in to the target node.
- **Step 4** Run **lsblk** to view the block device information of the node.

A data disk is divided depending on the container storage **Rootfs**:

Overlayfs: No independent thin pool is allocated. Image data is stored in dockersys.

Check the disk and partition space of the device. 1.

# ISDIK			
NAME	MAJ:MIN RN	1 SIZE R	O TYPE MOUNTPOINT
sda	8:0 0 50G	0 disk	
└─sda1	8:1 0 50)G 0 par	t /
sdb	8:16 0 150	G 0 disk	# The data disk has been expanded to 150 GiB, but 50 GiB
space is free.			
-vgpaas-dock	ersys 253:0	0 90G	0 lvm /var/lib/containerd
└─vgpaas-kubei	netes 253:1	0 10G	0 lvm /mnt/paas/kubernetes/kubelet

Expand the disk capacity. 2.

> Add the new disk capacity to the **dockersys** logical volume used by the container engine.

> Expand the PV capacity so that LVM can identify the new EVS a. capacity. /dev/sdb specifies the physical volume where dockersys is located. pvresize /*dev/sdb*

Information similar to the following is displayed:

Physical volume "/dev/sdb" changed 1 physical volume(s) resized or updated / 0 physical volume(s) not resized

Expand 100% of the free capacity to the logical volume. *vgpaas*/ b. *dockersys* specifies the logical volume used by the container engine. lvextend -l+100%FREE -n vgpaas/dockersys

Information similar to the following is displayed:

Size of logical volume vgpaas/dockersys changed from <90.00 GiB (23039 extents) to 140.00 GiB (35840 extents).

Logical volume vgpaas/dockersys successfully resized.

 Adjust the size of the file system. /dev/vgpaas/dockersys specifies the file system path of the container engine. resize2fs /dev/vgpaas/dockersys

Information similar to the following is displayed:

Filesystem at /dev/vgpaas/dockersys is mounted on /var/lib/containerd; on-line resizing required old_desc_blocks = 12, new_desc_blocks = 18 The filesystem on /dev/vgpaas/dockersys is now 36700160 blocks long.

3. Check whether the capacity has been expanded.

# ISDIK					
NAME	MAJ:MIN RM	1 SIZE	RO TYPE	MOUNTPOINT	
sda	8:0 0 50G	0 disk			
└─sda1	8:1 0 50)G 0 p	art /		
sdb	8:16 0 1500	G 0 dis	k		
-vgpaas-doo	: kersys 253:0	0 14	0G 0 lvr	n /var/lib/contai	nerd
└─vgpaas-kub	ernetes 253:1	0 10	G 0 lvm	/mnt/paas/kube	ernetes/kubelet

Device Mapper: A thin pool is allocated to store image data.

1. Check the disk and partition space of the device.

7	# lsblk	
ľ	NAME	MAJ:MIN RM SIZE RO TYPE MOUNTPOINT
١	/da	8:0 0 50G 0 disk
	└─vda1	8:1 0 50G 0 part /
١	/db	8:16 0 200G 0 disk
	-vgpaas-dockersys	253:0 0 18G 0 lvm /var/lib/docker
	vgpaas-thinpool_tmeta	253:1 0 3G 0 lvm
	└─vgpaas-thinpool	253:3 0 67G 0 lvm # Space used by thin pool
	vgpaas-thinpool_tdata	253:2 0 67G 0 lvm
	└──vgpaas-thinpool	253:3 0 67G 0 lvm
	-vanaas-kubernetes	253:4 0 10G 0 lvm /mnt/naas/kubernetes/kubelet

2. Expand the disk capacity.

Option 1: Add the new disk capacity to the thin pool.

a. Expand the PV capacity so that LVM can identify the new EVS capacity. /*dev*/vdb specifies the physical volume where thin pool is located.

pvresize /dev/vdb

Information similar to the following is displayed:

Physical volume "/dev/vdb" changed

1 physical volume(s) resized or updated / 0 physical volume(s) not resized

b. Expand 100% of the free capacity to the logical volume. *vgpaas/thinpool* specifies the logical volume used by the container engine. Ivextend -l+100%FREE -n *vgpaas/thinpool*

Information similar to the following is displayed:

Size of logical volume vgpaas/thinpool changed from <67.00 GiB (23039 extents) to <167.00 GiB (48639 extents).

Logical volume vgpaas/thinpool successfully resized.

- c. Do not need to adjust the size of the file system, because the thin pool is not mounted to any devices.
- d. Run the **lsblk** command to check the disk and partition space of the device and check whether the capacity has been expanded. If the new disk capacity was added to the thin pool, the capacity has been expanded.

# lsblk	
NAME	MAJ:MIN RM SIZE RO TYPE MOUNTPOINT
vda	8:0 0 50G 0 disk
└─vda1	8:1 0 50G 0 part /
	-

vdb	8:16 0 200G 0 disk	
-vgpaas-dockersys	253:0 0 18G 0 lvm /var/li	b/docker
vgpaas-thinpool_tmeta	253:1 0 3G 0 lvm	
└─vgpaas-thinpool	253:3 0 167G 0 lvm	# Thin pool space after
capacity expansion		
vgpaas-thinpool_tdata	253:2 0 67G 0 lvm	
vgpaas-thinpool	253:3 0 67G 0 lvm	
└─vgpaas-kubernetes	253:4 0 10G 0 lvm /mnt	/paas/kubernetes/kubelet

Option 2: Add the new disk capacity to the **dockersys** disk.

a. Expand the PV capacity so that LVM can identify the new EVS capacity. */dev/vdb* specifies the physical volume where dockersys is located.

pvresize /dev/vdb

Information similar to the following is displayed:

Physical volume "/dev/vdb" changed 1 physical volume(s) resized or updated / 0 physical volume(s) not resized

b. Expand 100% of the free capacity to the logical volume. *vgpaas/ dockersys* specifies the logical volume used by the container engine. Ivextend -l+100%FREE -n *vgpaas/dockersys*

Information similar to the following is displayed:

Size of logical volume vgpaas/dockersys changed from <18.00 GiB (4607 extents) to <118.00 GiB (30208 extents). Logical volume vgpaas/dockersys successfully resized.

c. Adjust the size of the file system. /dev/vgpaas/dockersys specifies the file system path of the container engine. resize2fs /dev/vgpaas/dockersys

Information similar to the following is displayed:

Filesystem at /dev/vgpaas/dockersys is mounted on /var/lib/docker; on-line resizing required old_desc_blocks = 3, new_desc_blocks = 15 The filesystem on /dev/vgpaas/dockersys is now 20022002 blocks long

The filesystem on /dev/vgpaas/dockersys is now 30932992 blocks long.

d. Run the **lsblk** command to check the disk and partition space of the device and check whether the capacity has been expanded. If the new disk capacity was added to the dockersys, the capacity has been expanded.

# lsblk	
NAME	MAJ:MIN RM SIZE RO TYPE MOUNTPOINT
vda	8:0 0 50G 0 disk
└─vda1	8:1 0 50G 0 part /
vdb	8:16 0 200G 0 disk
-vgpaas-dockersys	253:0 0 118G 0 lvm /var/lib/docker # dockersys after
capacity expansion	
-vgpaas-thinpool tmeta	253:1 0 3G 0 lvm
└─vgpaas-thinpool	253:3 0 67G 0 lvm
'	
-vgpaas-thinpool tdata	253:2 0 67G 0 lvm
└─vgpaas-thinpool	253:3 0 67G 0 lvm
-vgpaas-kubernetes	253:4 0 10G 0 lvm /mnt/paas/kubernetes/kubelet
51	

----End

Expanding the kubelet Capacity

The kubelet space serves as a temporary storage location for kubelet components and emptyDir. You can follow the following steps to increase the kubelet capacity:

Step 1 Expand the capacity of a data disk on the EVS console. For details, see Expanding EVS Disk Capacity.

Only the storage capacity of EVS disks can be expanded. You need to perform the following operations to expand the capacity of logical volumes and file systems.

- Step 2 Log in to the CCE console and click the cluster name to access the cluster console. In the navigation pane, choose Nodes. In the right pane, click the Nodes tab, locate the row containing the target node, and choose More > Sync Server Data in the Operation column.
- **Step 3** Log in to the target node.
- **Step 4** Run **lsblk** to view the block device information of the node.

lsblk
NAME MAJ:MIN RM SIZE RO TYPE MOUNTPOINT
sda 8:0 0 50G 0 disk
__sda1 8:1 0 50G 0 part /
sdb 8:16 0 200G 0 disk #The data disk has been expanded to 200 GiB, but 50 GiB space
is not allocated.
__vgpaas-dockersys 253:0 0 140G 0 lvm /var/lib/containerd
__vgpaas-kubernetes 253:1 0 10G 0 lvm /mnt/paas/kubernetes/kubelet

- **Step 5** Perform the following operations on the node to add the new disk capacity to the kubelet space:
 - 1. Expand the PV capacity so that LVM can identify the new EVS capacity. /*dev/sdb* specifies the physical volume where kubelet is located. pvresize /*dev/sdb*

Information similar to the following is displayed:

Physical volume "/dev/sdb" changed 1 physical volume(s) resized or updated / 0 physical volume(s) not resized

2. Expand 100% of the free capacity to the logical volume. *vgpaas/kubernetes* specifies the logical volume used by kubelet. Ivextend -l+100%FREE -n *vgpaas/kubernetes*

Information similar to the following is displayed:

Size of logical volume vgpaas/kubernetes changed from <10.00 GiB (2559 extents) to <60.00 GiB (15359 extents).

Logical volume vgpaas/kubernetes successfully resized.

 Adjust the size of the file system. /dev/vgpaas/kubernetes specifies the file system path of the container engine. resize2fs /dev/vgpaas/kubernetes

Information similar to the following is displayed:

Filesystem at /dev/vgpaas/kubernetes is mounted on /mnt/paas/kubernetes/kubelet; on-line resizing required

old_desc_blocks = 2, new_desc_blocks = 8

The filesystem on /dev/vgpaas/kubernetes is now 15727616 blocks long.

Step 6 Run **lsblk** to view the block device information of the node.

Isblk
NAME MAJ:MIN RM SIZE RO TYPE MOUNTPOINT
sda 8:0 0 50G 0 disk
__sda1 8:1 0 50G 0 part /
sdb 8:16 0 200G 0 disk
__vgpaas-dockersys 253:0 0 140G 0 lvm /var/lib/containerd
_vgpaas-kubernetes 253:1 0 60G 0 lvm /mnt/paas/kubernetes/kubelet # Allocate the new disk to
the kubelet space.

```
----End
```

Expanding Capacity of the Disk Shared by Container Engine and kubelet

To expand the capacity of the disk shared by container engine and kubelet, perform the following operations:

Step 1 Expand the capacity of a data disk on the EVS console. For details, see Expanding EVS Disk Capacity.

Only the storage capacity of EVS disks can be expanded. You need to perform the following operations to expand the capacity of logical volumes and file systems.

- Step 2 Log in to the CCE console and click the cluster name to access the cluster console. In the navigation pane, choose Nodes. In the right pane, click the Nodes tab, locate the row containing the target node, and choose More > Sync Server Data in the Operation column.
- **Step 3** Log in to the target node.
- **Step 4** Run **lsblk** to view the block device information of the node.

Isblk
NAME MAJ:MIN RM SIZE RO TYPE MOUNTPOINT
sda 8:0 0 50G 0 disk
__sda1 8:1 0 50G 0 part /
sdb 8:16 0 120G 0 disk # The data disk has been expanded to 120 GiB, but 20 GiB space is not
allocated.
__vgpaas-share 253:0 0 100G 0 lvm /mnt/paas # Space used by the container engine and the kubelet
component

- **Step 5** Run the following commands on the node to add the new disk capacity to the shared disk:
 - Expand the PV capacity so that LVM can identify the new EVS capacity. /dev/sdb specifies the physical volume where the shared disk is located. pvresize /dev/sdb

Information similar to the following is displayed:

Physical volume "/dev/sdb" changed

- physical volume(s) resized or updated / 0 physical volume(s) not resized
 Expand 100% of the free capacity to the logical volume. *vgpaas/share*
- specifies the logical volume shared by the container engine and the kubelet component.

lvextend -l+100%FREE -n vgpaas/share

Information similar to the following is displayed:

Size of logical volume vgpaas/share changed from <100.00 GiB (25599 extents) to <120.00 GiB (30719 extents).

Logical volume vgpaas/share successfully resized.

3. Adjust the size of the file system. /*dev/vgpaas/share* specifies the file system path of the shared disk. resize2fs /*dev/vgpaas/share*

Information similar to the following is displayed:

Filesystem at /dev/vgpaas/share is mounted on /mnt/paas; on-line resizing required old_desc_blocks = 13, new_desc_blocks = 15 The filesystem on /dev/vgpaas/share is now 31456256 blocks long.

Step 6 Run **lsblk** to view the block device information of the node.

# lsblk	
NAME	MAJ:MIN RM SIZE RO TYPE MOUNTPOINT
sda	8:0 0 50G 0 disk
└─sda1	8:1 0 50G 0 part /

 sdb
 8:16
 0
 120G
 0 disk

 vgpaas-share
 253:0
 0
 120G
 0 lvm /mnt/paas # Space of the new disk used by the container engine and the kubelet component

----End

Expanding the Capacity of a Data Disk Used by Pod (basesize)

- **Step 1** Log in to the CCE console and click the cluster name to access the cluster console.
- **Step 2** Choose **Nodes** from the navigation pane.
- Step 3 Click the Nodes tab, locate the row containing the target node, and choose More > Reset Node in the Operation column.

NOTICE

Resetting a node may make the node-specific resources (such as local storage and workloads scheduled to this node) unavailable. Exercise caution when performing this operation to avoid impact on running services.

Step 4 Reconfigure node parameters.

If you need to adjust the container storage space, pay attention to the following configurations:

Storage Settings: Click **Expand** next to the data disk to configure the following parameter:

Space Allocation for Pods: indicates the base size of a pod. It is the maximum size that a workload's pods (including the container images) can grow to in the disk space. Proper settings can prevent pods from taking all the disk space available and avoid service exceptions. It is recommended that the value is less than or equal to 80% of the container engine space. This parameter is related to the node OS and container storage rootfs and is not supported in some scenarios.

For more information about container storage space allocation, see **Data Disk Space Allocation**.

- **Step 5** After the node is reset, log in to the node and check whether the container capacity has been expanded. The command output varies with the container storage rootfs.
 - Overlayfs: No independent thin pool is allocated. Image data is stored in dockersys. Run the following command to check whether the container capacity has been expanded: kubectl exec -it pod_name -- /bin/sh

df -h

If the information similar to the following is displayed, the overlay capacity has been expanded from 10 GiB to 15 GiB.

Filesystem	Size	Used	Avail	Use%	Mounted on
overlay	15G	104K	15G	1%	/
tmpfs	64M	0	64M	0%	/dev
tmpfs	3.6G	0 3	3.6G	0% /	sys/fs/cgroup
/dev/mapper/vgpaas-s	share	98G	4.0G	89G	5% /etc/hosts

• Device Mapper: A thin pool is allocated to store image data. Run the following command to check whether the container capacity has been expanded:

```
kubectl exec -it pod_name -- /bin/sh
df -h
```

If the information similar to the following is displayed, the thin pool capacity has been expanded from 10 GiB to 15 GiB.

```
Size Used Avail Use% Mounted on
Filesystem
/dev/mapper/vgpaas-thinpool-snap-84 15G 232M
                                            15G
                                                  2% /
tmpfs
                     64M 0 64M 0% /dev
tmpfs
                     3.6G
                           0 3.6G 0% /sys/fs/cgroup
/dev/mapper/vgpaas-kubernetes
                                              1% /etc/hosts
                             11G 41M
                                        11G
/dev/mapper/vgpaas-dockersys
                             20G 1.1G 18G
                                             6% /etc/hostname
```

----End

Expanding a PVC

Cloud storage:

- OBS and SFS: There is no storage restriction and capacity expansion is not required.
- EVS:
 - You can expand the capacity of automatically created pay-per-use volumes on the CCE console. The procedure is as follows:
 - i. Choose **Storage** in the navigation pane. In the right pane, click the **PVCs** tab. Click **More** in the **Operation** column of the target PVC and select **Scale-out**.
 - ii. Enter the capacity to be added and click **OK**.
 - For yearly/monthly-billed instances, expand the capacity on the EVS console and then change the capacity in the PVC.
- SFS Turbo:
 - If no subdirectory is used, the following situations may occur:
 - If the underlying SFS Turbo file systems are billed on the pay-per-use basis, you can expand their capacity on the CCE console.
 - If the underlying SFS Turbo file systems are billed on the yearly/ monthly basis, you can expand the capacity on the SFS Turbo console first and then adjust it in the PVC on the CCE console.
 - If a subdirectory is used, the following situations may occur:
 - If the expanded capacity does not surpass the underlying SFS Turbo file system's capacity, you can increase the PVC capacity on the CCE console.
 - If the expanded capacity exceeds the underlying SFS Turbo file system's capacity, you can expand the file system capacity on the SFS console first and then adjust the capacity in the PVC on the CCE console.
- Local PV: Before expanding the capacity of a local PV, check the available capacity of the storage pool for the node where the PVC is located. If the

expanded capacity of the local PV is less than or equal to the available capacity of the storage pool, directly adjust the PVC capacity. Otherwise, expand the storage pool and then change the capacity of the PVC.

11.2 Mounting Object Storage Across Accounts

Application Scenarios

- Cross-account data sharing. For example, multiple teams within a company need to share data, but each team uses a different account.
- Cross-account data migration and backup. When account A is about to be disabled, all data stored in the account needs to be transferred to a new account (account B).
- Data processing and analysis. For example, account B is an external data processor and needs to access raw data from account A to perform tasks such as big data analysis and machine learning.

By linking object storage across accounts, you can share data, lower storage and transmission expenses, and guarantee data security and consistency. This enables various teams or organizations to securely and conveniently access each other's data resources, eliminating the need for repeated storage and redundant transmission. Additionally, data is kept current and compliant, enhancing overall service efficiency and security.

Procedure

Assume that account B needs to access and use an OBS bucket of account A. For details, see **Figure 11-1** and **Table 11-2**.



Figure 11-1 Mounting an OBS bucket across accounts

Table 11-2 Process description

Procedure	Description
Step 1: Create an OBS Bucket Policy and ACL	Configure an OBS bucket policy and ACL using account A and grant account B required permissions like the read and write permissions.
Step 2: Create a Workload with an OBS Volume Mounted	Create a PV and a PVC based on the OBS bucket of account A using account B and mount the PVC to the required workload.
Step 3: Check the Pod Actions on the OBS Bucket	Check whether the pod created by account B has the required permissions based on the bucket policy.
Step 4: Clear Resources	Once you have studied this example, delete any associated resources to prevent incurring settlement fees.

Prerequisites

- The involved accounts are in the same region.
- You have created a cluster where the CCE Container Storage (Everest) add-on is installed. The add version must be 1.1.11 or later, and the cluster version must be 1.15 or later. If no cluster is available, create one by referring to **Buying a CCE Standard/Turbo Cluster**.
- An ECS with an EIP bound has been created in the same VPC as the cluster, and the ECS can access the cluster using kubectl. For details about how to access a cluster from an ECS, see **Connecting to a Cluster Using kubectl**.

Step 1: Create an OBS Bucket Policy and ACL

Configure an OBS bucket policy and ACL using account A and grant account B required permissions like the read and write permissions.

- **Step 1** Log in to the OBS console. In the navigation pane, choose **Object Storage**.
- Step 2 Click the name of the target bucket to go to the Objects page.
- **Step 3** In the navigation pane, choose **Permissions** > **Bucket Policies**. On the page displayed, click **Create**.
- **Step 4** Configure the parameters. In this example, only some mandatory parameters are described. You can keep the default values for other parameters. For details about the parameters, see **Bucket Policies**.

Table 11-3 Bucket	policy parameters
-------------------	-------------------

Parameter	Description	Example Value
Policy Name	Enter a name.	example01

Parameter	Description	Example Value
Effect	 Specify the behavior of a policy. Allow: The actions defined in the policy are allowed. Deny: The actions defined in the policy are denied. 	Allow
Principal	 Specify authorized accounts. (Multiple accounts can be selected.) For different types of authorized accounts, the OBS console provides different templates for authorizations. For details, see Creating a Bucket Policy with a Template. All accounts: Any account can execute the current bucket policy without identity authentication, which may pose data security risks. Current account: Grant permissions to a specific IAM account under the current account. Other accounts: Grant permissions to a specific IAM account under another account. 	Other accounts XXX(account ID)/XXX (IAM ID)
Resources	 Specify the authorized resources. Entire bucket (including the objects in it): Allow authorized accounts to perform certain actions on a bucket and the objects in it. Current bucket: Allow authorized accounts to perform certain actions on the current bucket. Specific objects: Allow authorized accounts to perform certain actions on the specified objects in a bucket. 	Entire bucket (including the objects in it)
Actions	 Specify actions. Use a template: Use a permission template preset on the OBS console. If you selected Bucket Read/Write, the Specified actions option will be selected by default in the Advanced Settings area. Customize: Customize the actions. 	Use a template > Bucket Read/ Write

Figure 11-2 Creating a bucket policy

isual Editor JS	ON
Policy Name	example01
€ Effect	Allow Deny
Ł Principal	 All accounts Current account ✓ Other accounts Account ID/* indicates that permissions are granted to all IAM users under the account. Learn how to find an Account ID and IAM User ID.
Resources	Entire bucket (including the objects in it) Current bucket Specified objects
	Bucket Read-Only Bucket Read/Write
Conditions (Optional)	Add Condition Conditions required for this policy to take effect. A condition is expressed as a key value pair. View configuration examples [2]
	Key \Leftrightarrow Condition Oper \Leftrightarrow Value \Leftrightarrow Operation
	No conditions added.
 Advanced Settings 	8

Step 5 In the navigation pane, choose Permissions > Bucket ACL. In the right pane, click Add under User Access. Enter the account ID of the authorized user, select Read and Write for Access to Bucket, select Read for Access to Objects, select Read and Write for Access to ACL, and click OK.

----End

Step 2: Create a Workload with an OBS Volume Mounted

Create a PV and a PVC based on the OBS bucket of account A using account B and mount the PVC to the required workload.

Step 1 Create a ConfigMap named **paas-obs-endpoint** and configure the region and endpoint of OBS.

vim *config.yaml*

The content is as follows: (For details about the parameters, see Table 11-4.)

```
apiVersion: v1
kind: ConfigMap
metadata:
name: paas-obs-endpoint # The value must be paas-obs-endpoint.
namespace: kube-system # The value must be kube-system.
data:
obs-endpoint: |
    {"<region_name>": "<endpoint_address>"}
```

Create the ConfigMap using config.yaml.

kubectl create -f *config.yaml*

 Table 11-4 ConfigMap parameters

ConfigMap	Description	Example Value
metadata.na me	ConfigMap name, which is fixed at paas-obs- endpoint and cannot be changed	paas-obs-endpoint
metadata.na mespace	Namespace, which is fixed at kube-system and cannot be changed	kube-system
data.obs- endpoint	Region names and endpoints are in key-value pairs. Replace <i><region_name></region_name></i> and <i><endpoint_address></endpoint_address></i> with specific values. If multiple values are needed, use commas (,) to separate them. For details about its value, see Regions and Endpoints .	{"ap-southeast-1": "https:// obs.ap- southeast-1.myhuaweicloud.com :443", "ap-southeast-3": "https:// obs.ap- southeast-3.myhuaweicloud.com :443"}

- **Step 2** Create a secret named **test-user**. (This secret is used to provide access credentials when volumes are mounted to CSI, and its name can be customized.)
 - 1. Obtain the AK. Go back to the management console, hover the cursor over the username in the upper right corner and choose **My Credentials** from the drop-down list.

In the navigation pane, choose **Access Keys**. On the page displayed, click **Create Access Key**.

Click **OK** and download the AK.

- 2. Encode the AK using Base64 and save the encoded AK and SK. If the AK obtained is **xxx** and the SK is **yyy**, run the following commands: echo -n xxx|base64 echo -n yyy|base64
- 3. Create a secret YAML file, for example, **test_user.yaml**. vim *test_user.yaml*

The content is as follows: (For details about the parameters, see Table 11-5.)

```
apiVersion: v1
data:
access.key: QUxPQUUU******
secret.key: aVMwZkduQ******
kind: Secret
metadata:
```

name: *test-user* namespace: default type: cfe/secure-opaque

Create a secret using test_user.yaml.

kubectl create -f *test_user.yaml*

Table 11-5 Secret parameters

Parameter	Description	Example Value
access.key	A Base64-encoded AK	QUxPQUUU******
secret.key	A Base64-encoded SK	aVMwZkduQ*****
type	Key type, which is fixed at cfe/ secure-opaque and cannot be changed	cfe/secure-opaque
	When this type is used, the data entered by users will be automatically encrypted.	

Step 3 Create a PV named **testing_abc** and mount the secret named **test_user** to the PV. vim *testing_abc.yaml*

The content is as follows: (For details about the parameters, see Table 11-6.)

```
kind: PersistentVolume
apiVersion: v1
metadata:
name: testing-abc
 annotations:
  pv.kubernetes.io/bound-by-controller: 'yes'
  pv.kubernetes.io/provisioned-by: everest-csi-provisioner
spec:
 capacity:
  storage: 1Gi
 mountOptions:
 - default_acl=bucket-owner-full-control
                                           #New OBS mounting parameters
 csi:
  driver: obs.csi.everest.io
  volumeHandle: obs-cce-test
                                         # Name of the OBS bucket to be mounted
  fsType: s3fs
                                  # obsfs indicates a parallel file system, and s3fs indicates an OBS bucket.
  volumeAttributes:
   everest.io/obs-volume-type: STANDARD # Bucket type, which can be STANDARD or WARM when an
OBS bucket is used
   everest.io/region: <region_name>
                                          # Region where the OBS bucket is located (Replace it with the
actual value.)
   storage.kubernetes.io/csiProvisionerIdentity: everest-csi-provisioner
                                      # AK/SK used for mounting the OBS bucket
  nodePublishSecretRef:
   name: test-user
   namespace: default
 accessModes:
  - ReadWriteMany
 persistentVolumeReclaimPolicy: Retain
                                           # PV reclaim policy
 storageClassName: csi-obs
                                        # csi-obs specifies an OBS storage class that is automatically
created. You can customize it as required.
volumeMode: Filesystem
```

Create the PV using testing_abc.yaml.

kubectl create -f *testing_abc.yaml*

Paramet er	Description	Example Value
mountOp tions.defa ult_acl	 Specify access control policies for a bucket and objects in the bucket. In this example, account A owns the bucket, and both account A and account B have the ability to upload data. private: The bucket or objects can only be fully accessed by the owner of the bucket. public-read: The owner of the bucket has complete control over both the bucket and its objects. While other users can read data from the bucket, they are unable to modify, delete, or upload any data within it. public-read-write: The owner of the bucket has complete control over the bucket and its objects. Other users can read and write data from and to the bucket. bucket-owner-read: Users who have uploaded objects to the bucket has complete control over the objects, while the bucket owner is only granted read permissions for said objects. This mode is usually used in cross-account sharing scenarios. bucket-owner-full-control: Users who have uploaded objects to the bucket are granted write permissions for those specific objects, but not read permissions by default. The bucket owner has complete control over the bucket are granted write permissions for those specific objects, but not read permissions by default. The bucket owner has complete control over all objects within the bucket. This mode is usually used in cross-account sharing scenarios. 	bucket-owner-full- control NOTE Due to the bucket policy that was configured using account A, account B has been granted read and write permissions for the entire bucket, including objects uploaded by both account A and B.
csi.nodeP	Specify the secret to be mounted.	test-user
ublishSec	name: name of a secret	default
	 namespace: namespace where the secret is in 	
csi.volum eHandle	Specify the name of the OBS bucket to be mounted.	obs-cce-test (OBS bucket name authorized by account A)

Table 11-6 PV parameters

Paramet er	Description	Example Value
csi.fsType	 Specify the file type. obsfs: Create an OBS parallel file system. s3fs: Create an OBS bucket. 	s3fs NOTICE The file type specified by the PV and the PVC must match in order for the PV to be bound to the corresponding PVC. If they do not match, the binding cannot occur.
accessMo des	 Specify the access mode of the storage volume. OBS supports only ReadWriteMany. ReadWriteOnce: A storage volume can be mounted to a single node in read-write mode. ReadWriteMany: A storage volume can be mounted to multiple nodes in read-write mode. 	ReadWriteMany
persistent VolumeR eclaimPol icy	 Specify the reclaim policy for the PV. Delete: When a PVC is deleted, both the PV and underlying storage resources will be deleted. If the everest.io/reclaim-policy: retain-volume-only annotation is added to the YAML file, the underlying storage resources will be retained. Retain: When a PVC is deleted, both the PV and underlying storage resources will be retained. Retained. You need to manually delete them. After the PVC is deleted, the PV is in the Released state and cannot be bound to a PVC again. 	Retain NOTE If multiple PVs use the same OBS volume, use Retain to prevent the underlying volume from being deleted with one of the PV.

Step 4 Create a PVC named **pvc-test-abc** and bind the new PV **testing_abc** to it.

vim *pvc_test_abc.yaml*

```
The file content is as follows:
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
 name: pvc-test-abc
 namespace: default
 annotations:
  csi.storage.k8s.io/node-publish-secret-name: test-user # Mount a secret.
  csi.storage.k8s.io/node-publish-secret-namespace: default # Namespace of the secret
   everest.io/obs-volume-type: STANDARD # Bucket type, which can be STANDARD or WARM when an
OBS bucket is used
                                     # File type. obsfs indicates a parallel file system, and s3fs indicates
  csi.storage.k8s.io/fstype: s3fs
an OBS bucket.
  volume.beta.kubernetes.io/storage-provisioner: everest-csi-provisioner
spec:
```

accessModes: - ReadWriteMany # The value must be **ReadWriteMany** for object storage. resources: requests: storage: 1Gi # Storage capacity of a PVC. This parameter is valid only for verification (fixed to 1, cannot be empty or 0). The value setting does not take effect for OBS buckets. storageClassName: *csi-obs* # **csi-obs** specifies an OBS storage class that is automatically created. You can customize it as required. volumeName: *testing-abc* # PV name

Create the PVC using pvc_test_abc.yaml.

kubectl create -f pvc_test_abc.yaml

Step 5 Create a workload and mount the PVC to it. The following uses an Nginx Deployment as an example.

vim *obs_deployment_example.yaml*

The file content is as follows:

apiVersion: apps/v1 kind: Deployment	
metadata: name: <i>obs-deployment-example</i>	# Workload name, which can be customized
namespace: default	,
spec:	
replicas: 1	
selector:	
matchLabels:	
app: <i>obs-deployment-example</i>	# Label, which can be customized
template:	
metadata:	
labels:	
app: <i>obs-deployment-example</i>	
spec:	
containers:	
- image: nginx	
name: container-0	
volumeMounts:	
- mountPath: /tmp	# PVC mount path, which can be customized as required
name: pvc-obs-example	
restartPolicy: Always	
ImagePullSecrets:	
- name: default-secret	
volumes.	
- Indiffe. pvc-obs-example	
claimNamo: pyc tast abc	# DV/C name
$p_{l-lesl-abc}$	

Create the workload named **obs-deployment-example** using **obs_deployment_example.yaml**.

kubectl create -f *obs_deployment_example.yaml*

Check whether the workload has been created.

kubectl get pod

If information similar to the following is displayed and the workload is in the **Running** state, the workload has been created.

NAMEREADYSTATUSRESTARTSAGEobs-deployment-example-6b4dfd7b57-frfxv1/1Running022h

----End

Step 3: Check the Pod Actions on the OBS Bucket

Check whether the pod created by account B has the required permissions based on the bucket policy.

Step 1 Check whether the pod can read and write objects in the OBS bucket created by account A and assume that a **test.txt** file is present in the OBS bucket.

Run the following command to access the created workload. (You can press **Ctrl** +**D** to exit the current workload.)

kubectl -n default exec -it obs-deployment-example-6b4dfd7b57-frfxv -c container-0 -- bash

Run the following command to check the pod actions on **test.txt**. **/tmp** specifies the PVC mount path.

ls -l /tmp/test.txt

If information similar to the following is displayed, the pod has the read and write permissions on the **test.txt** file, which is related to the bucket policy set by account A.

-rwxrwxrwx 1 root root 4 Sep 5 09:09 /tmp/test.txt

Step 2 Check whether the pod can read and write data from and to the objects uploaded by itself in the OBS bucket.

Create a **test01.txt** file in **/tmp** and write **test\n** into the file. echo -e "test\n" > /tmp/test01.txt

Run the following command to check the **test01.txt** content and check whether the pod can read and write new objects uploaded by itself: (Account A can check the new objects in the OBS bucket.)

cat /tmp/test01.txt

If information similar the following is displayed, the pod has the read and write permissions on the objects uploaded by itself.

test

----End

Step 4: Clear Resources

Once you have studied this example, delete any associated resources to prevent incurring settlement fees. If you plan to learn other examples, wait until they are finished before doing any clean-up.

Step 1 Run the following command to delete the workload:

kubectl delete -f obs_deployment_example.yaml

Information similar to the following is displayed:

deployment.apps "obs-deployment-example" deleted

Step 2 Run the following command to delete the PVC: kubectl delete -f *pvc_test_abc.yaml*

Information similar to the following is displayed:

persistentvolumeclaim "pvc-test-abc" deleted

Step 3 Run the following command to delete the PV: kubectl delete -f *testing_abc.yaml*

Information similar to the following is displayed:

persistentvolume "testing-abc" deleted

Step 4 Run the following command to delete the secret: kubectl delete -f *test_user.yaml*

Information similar to the following is displayed:

secret "test-user" deleted

Step 5 Run the following command to delete the ConfigMap: kubectl delete -f *config.yaml*

Information similar to the following is displayed:

configmap "paas-obs-endpoint" deleted

----End

Common Issues

If a workload fails to be created, locate the fault based on the error information in the pod events. For details, see **Table 11-7**. If the problem persists, **create a service ticket** to contact customer service for help.

Error	Possible Cause	Fault Locating
0/4 nodes are available: pod has unbound immediate PersistentVolum eClaims. preemption: 0/4 nodes are available: 4 Preemption is not helpful for scheduling.	The PVC is not bound to any PV.	 Run the following command to check the PVC status: kubectl get pvc If the PVC is in the Pending state, it is not bound to any PV. Check the PVC details and locate the cause of the binding failure. kubectl describe pvc <pvc_name></pvc_name> Modify the YAML file to rectify the fault if the fault is caused by any of the following reasons: The PVC is not bound to the proper PV. The PVC and PV parameters do not match. This includes fsType, StorageClass, accessModes, and storage. The StorageClass must be object storage, and accessModes must be set to ReadWriteMany because OBS buckets only support this mode. Additionally, the storage value requested by the PVC must be equal to or less than the storage value provided by the PV.

 Table 11-7 Locating the fault
Error	Possible Cause	Fault Locating
MountVolume.S etUp failed for volume <i>"obs-</i> <i>cce-example"</i> : rpc error: code = Unknown desc = failed to get secret(paas.long aksk), err: get secret(paas.long aksk) failed: get secret paas.longaksk from namespace kube-system failed: secrets "paas.longaksk" not found	The required secret cannot be found when the storage volume is mounte d to the workloa d.	 Check whether the mounted secret is present. kubectl get secret If it is present, the secret may be incorrectly configured. If it is not present, you can create one by referring to Step 2. Check the secret parameter configurations. The following shows the common issues: The access.key and secret.key parameters are not set to the AK and SK encoded using Base64. The type parameter is not set to cfe/ secure-opaque.
MountVolume.S etUp failed for volume "pv-obs- example": rpc error: code = Internal desc = [8032c354-4e1b -41b0-81ce-9d4 b3f8c49c9] get obsUrl failed before mount bucket obs-cce- example, get configMap paas- obs-endpoint from namespace kube-system failed: configmaps "paas-obs- endpoint" not found	The ConfigM ap that stores the OBS endpoin t is not present or is incorrect ly configur ed.	 Check whether the ConfigMap is present. kubectl get configmap -n kube-system If it is present, the ConfigMap may be incorrectly configured. If it is not present, you can create one by referring to Step 1. Check the ConfigMap parameter configurations. The following shows the common issues: The name parameter is not set to paas- obs-endpoint. The namespace parameter is not set to kube-system. The region name is not set to the region where the OBS bucket is located.

Background

The minimum capacity of an SFS Turbo file system is 500 GiB, and the SFS Turbo file system cannot be billed by usage. By default, the root directory of an SFS Turbo file system is mounted to a container which, in most case, does not require such a large capacity.

The Everest add-on allows you to dynamically create subdirectories in an SFS Turbo file system and mount these subdirectories to containers. In this way, an SFS Turbo file system can be shared by multiple containers to increase storage efficiency.

Notes and Constraints

- Only clusters of v1.15 or later are supported.
- The cluster must use the Everest add-on of version 1.1.13 or later.
- Kata containers are not supported.
- When the Everest add-on earlier than 1.2.69 or 2.1.11 is used, a maximum of 10 PVCs can be created concurrently at a time by using the subdirectory function. everest of 1.2.69 or later or of 2.1.11 or later is recommended.
- A subPath volume is a subdirectory of an SFS Turbo file system. Increasing the capacity of a PVC of this type only changes the resource range specified by the PVC, but does not change the total capacity of the SFS Turbo file system. If the SFS Turbo file system's total resource capacity is not enough, the available capacity of the subPath volume will be restricted. To fix this, you must increase the resource capacity of the SFS Turbo file system on the SFS Turbo console.

Deleting the subPath volume does not result in the deletion of the resources of the SFS Turbo file system.

Creating an SFS Turbo Volume of the subPath Type

Step 1 Create an SFS Turbo file system in the same VPC and subnet as the cluster.

Step 2 Create a YAML file of StorageClass, for example, **sfsturbo-subpath-sc.yaml**.

The following is an example:

apiVersion: storage.k8s.io/v1 allowVolumeExpansion: true kind: StorageClass metadata: name: sfsturbo-subpath-sc *# Storage class name* mountOptions: *#Mount options* - lock parameters: csi.storage.k8s.io/csi-driver-name: sfsturbo.csi.everest.io csi.storage.k8s.io/fstype: nfs everest.io/archive-on-delete: "true" everest.io/share-access-to: 7ca2dba2-1234-1234-626371a8fb3a

```
everest.io/share-expand-type: bandwidth
everest.io/share-export-location: 192.168.1.1:/sfsturbo/ # Mount directory configuration
everest.io/share-source: sfs-turbo
everest.io/share-volume-type: STANDARD
everest.io/volume-as: subpath
everest.io/volume-id: 0d773f2e-1234-1234-1234-de6a35074696 # ID of an SFS Turbo volume
provisioner: everest-csi-provisioner
reclaimPolicy: Delete
volumeBindingMode: Immediate
```

In this example:

- **name**: indicates the name of the StorageClass.
- **mountOptions**: (Optional) indicates the mount options.
 - In versions later than Everest 1.1.13 and earlier than Everest 1.2.8, only the **nolock** parameter can be configured. By default, the **nolock** parameter is used for the mount operation and does not need to be configured. If **nolock** is set to **false**, the **lock** field is used.
 - Starting from Everest 1.2.8, more mount options are supported. For details, see Configuring SFS Turbo Mount Options. Do not set nolock to true. Otherwise, the mount operation will fail. mountOptions:
 - vers=3
 - timeo=600 - nolock
 - nolock - hard
- **everest.io/volume-as**: This parameter is set to **subpath** to use the subPath volume.
- everest.io/share-access-to: This parameter is optional. In a subPath volume, set this parameter to the ID of the VPC where the SFS Turbo file system is located.
- everest.io/share-expand-type: This parameter is optional. If the type of the SFS Turbo file system is SFS Turbo Standard Enhanced or SFS Turbo Performance Enhanced, set this parameter to **bandwidth**.
- everest.io/share-export-location: This parameter indicates the mount directory. It consists of the SFS Turbo shared path and sub-directory. The shared path can be obtained on the SFS Turbo console. The sub-directory is user-defined. The PVCs created using the StorageClass are located in this sub-directory.
- everest.io/share-volume-type: This parameter is optional. It specifies the SFS Turbo file system type. The value can be STANDARD or PERFORMANCE. For enhanced types, this parameter must be used together with everest.io/shareexpand-type (whose value should be bandwidth).
- **everest.io/zone**: This parameter is optional. Set it to the AZ where the SFS Turbo file system is located.
- **everest.io/volume-id**: This parameter indicates the ID of the SFS Turbo volume. You can obtain the volume ID on the SFS Turbo page.
- everest.io/archive-on-delete: If this parameter is set to true and Delete is selected for Reclaim Policy, the original documents of the PV will be archived to the directory named archived-{\$PV name.timestamp} before the PVC is deleted. If this parameter is set to false, the SFS Turbo subdirectory of the corresponding PV will be deleted. The default value is true, indicating that the original documents of the PV will be archived to the directory named archived-{\$PV name.timestamp} before the PVC is deleted.

Step 3 Run kubectl create -f sfsturbo-subpath-sc.yaml.

Step 4 Create a PVC YAML file named sfs-turbo-test.yaml.

The following is an example:

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
name: sfs-turbo-test # PVC name
namespace: default
spec:
accessModes:
- ReadWriteMany
resources:
requests:
storage: 50Gi
storageClassName: sfsturbo-subpath-sc # Storage class name
volumeMode: Filesystem
```

In this example:

- name: indicates the name of the PVC.
- storageClassName: specifies the name of the StorageClass.
- storage: In a subPath volume, modifying the value of this parameter does not impact the resource capacity of the SFS Turbo file system. A subPath volume is essentially a file path within an SFS Turbo file system. As a result, increasing the capacity of the subPath volume in a PVC does not lead to an increase in the resources of the SFS Turbo file system.

NOTE

The capacity of a subPath volume is restricted by the overall resource capacity of the corresponding SFS Turbo file system. If the resources of the SFS Turbo file system are inadequate, you can adjust the resource capacity via the SFS Turbo console.

Step 5 Run kubectl create -f sfs-turbo-test.yaml.

----End

Creating a Deployment and Mounting an Existing Volume

Step 1 Create a YAML file for the Deployment, for example, deployment-test.yaml.

```
The following is an example:
apiVersion: apps/v1
kind: Deployment
metadata:
 name: test-turbo-subpath-example # Name of the created workload
 namespace: default
 generation: 1
 labels:
  appgroup: "
spec:
 replicas: 1
 selector:
  matchLabels:
   app: test-turbo-subpath-example
 template:
  metadata:
   labels:
     app: test-turbo-subpath-example
  spec:
   containers:
```

 image: nginx:latest # Image of the workload name: container-0 volumeMounts:
- mountPath: /tmp #Mount path in a container
name: pvc-sfs-turbo-example
restartPolicy: Always
imagePullSecrets:
- name: default-secret
volumes:
- name: pvc-sfs-turbo-example
persistentVolumeClaim:
claimName: sfs-turbo-test # Name of an existing PVC

In this example:

- **name**: indicates the name of the created workload.
- image: specifies the image used by the workload.
- mountPath: indicates the mount path of the container. In this example, the volume is mounted to the /tmp directory.
- claimName: indicates the name of an existing PVC.
- **Step 2** Create the Deployment.

kubectl create -f deployment-test.yaml

----End

Dynamically Creating a subPath Volume for a StatefulSet

Step 1 Create a YAML file for a StatefulSet, for example, statefulset-test.yaml.

The following is an example:

```
apiVersion: apps/v1
kind: StatefulSet
metadata:
 name: test-turbo-subpath # Name of the created workload
 namespace: default
 generation: 1
 labels:
  appgroup: "
spec:
 replicas: 2
 selector:
  matchLabels:
   app: test-turbo-subpath
 template:
  metadata:
    labels:
     app: test-turbo-subpath
    annotations:
     metrics.alpha.kubernetes.io/custom-endpoints: '[{"api":"","path":"","port":"","names":""}]'
     pod.alpha.kubernetes.io/initialized: 'true'
  spec:
    containers:
     - name: container-0
      image: 'nginx:latest' # Image of the workload
      resources: {}
      volumeMounts:
       - name: sfs-turbo-160024548582479676
         mountPath: /tmp # Mount path in a container
      terminationMessagePath: /dev/termination-log
      terminationMessagePolicy: File
      imagePullPolicy: IfNotPresent
    restartPolicy: Always
```

```
terminationGracePeriodSeconds: 30
  dnsPolicy: ClusterFirst
  securityContext: {}
  imagePullSecrets:
    - name: default-secret
  affinity: {}
  schedulerName: default-scheduler
volumeClaimTemplates:
 - metadata:
   name: sfs-turbo-160024548582479676
   namespace: default
   annotations: {}
  spec:
   accessModes:
     - ReadWriteMany
   resources:
     requests:
      storage: 10Gi
   storageClassName: sfsturbo-subpath-sc # Enter the name of a self-managed storage class.
serviceName: wwww
podManagementPolicy: OrderedReady
updateStrategy:
type: RollingUpdate
revisionHistoryLimit: 10
```

In this example:

- **name**: indicates the name of the created workload.
- image: specifies the image used by the workload.
- mountPath: indicates the mount path of the container. In this example, the volume is mounted to the /tmp directory.
- spec.template.spec.containers.volumeMounts.name and spec.volumeClaimTemplates.metadata.name: must be consistent because they have a mapping relationship.
- **storageClassName**: specifies the name of an on-premises StorageClass.
- **Step 2** Create the StatefulSet.

kubectl create -f statefulset-test.yaml

----End

11.4 Changing the Storage Class Used by a Cluster of v1.15 from FlexVolume to CSI Everest

In clusters later than v1.15.11-r1, CSI (the everest add-on) has taken over all functions of fuxi FlexVolume (the storage-driver add-on) for managing container storage. You are advised to use CSI Everest.

To migrate your storage volumes, create a static PV to associate with the original underlying storage, and then create a PVC to associate with this static PV. When you upgrade your application, mount the new PVC to the original mounting path to migrate the storage volumes.

Services will be interrupted during the migration. Therefore, properly plan the migration and back up data.

Procedure

- **Step 1** (Optional) Back up data to prevent data loss in case of exceptions.
- **Step 2** Configure a YAML file of the PV in the CSI format according to the PV in the FlexVolume format and associate the PV with the existing storage.

To be specific, run the following commands to configure the pv-example.yaml file, which is used to create a PV.

touch pv-example.yaml

vi pv-example.yaml

Configuration example of a PV for an EVS volume:	
apiVersion: v1	
kind: PersistentVolume	
metadata:	
labels:	
failure-domain.beta.kubernetes.io/region: ap-southeast-1	
failure-domain.beta.kubernetes.io/zone: <zone name=""></zone>	
annotations:	
pv.kubernetes.io/provisioned-by: everest-csi-provisioner	
name: pv-evs-example	
spec:	
accessModes:	
- ReadWriteOnce	
capacity:	
storage: 10Gi	
CSI:	
driver: disk.csi.everest.io	
TS Type: ext4	
volumeAttributes:	
everest.io/aisk-mode: SCSI	
everest.io/disk-volume-type: SAS	
volumenaliue. V532ubua-0540-4702-a742-410002002002	
etorsaclassicativoumeneetaminouty. Delete	
storageClassivanie, CSI-USK	

Parameter	Description
failure- domain.beta.kuber netes.io/region	Region where the EVS disk is located. Use the same value as that of the FlexVolume PV.
failure- domain.beta.kuber netes.io/zone	AZ where the EVS disk is located. Use the same value as that of the FlexVolume PV.
name	Name of the PV, which must be unique in the cluster.
storage	EVS volume capacity in the unit of Gi. Use the value of spec.capacity.storage of the FlexVolume PV.

Table	11-8	EVS	volume	confia	uration	parameters
			· · · · · · · · · ·	coning	aracioni	parameters

Parameter	Description
driver	Storage driver used to attach the volume. Set the driver to disk.csi.everest.io for the EVS volume.
volumeHandle	Volume ID of the EVS disk. Use the value of spec.flexVolume.options.volumeID of the FlexVolume PV.
everest.io/disk- mode	EVS disk mode. Use the value of spec.flexVolume.options.disk-mode of the FlexVolume PV.
everest.io/disk- volume-type	EVS disk type. Currently, high I/O (SAS) and ultra-high I/O (SSD) are supported. Use the value of kubernetes.io/ volumetype in the storage class corresponding to spec.storageClassName of the FlexVolume PV.
storageClassName	Name of the Kubernetes storage class associated with the storage volume. Set this field to csi-disk for EVS disks.

Configuration example of a PV for an SFS volume:

```
apiVersion: v1
.
kind: PersistentVolume
metadata:
name: pv-sfs-example
 annotations:
  pv.kubernetes.io/provisioned-by: everest-csi-provisioner
spec:
accessModes:
 - ReadWriteMany
 capacity:
 storage: 10Gi
 csi:
  driver: nas.csi.everest.io
  fsType: nfs
  volumeAttributes:
   everest.io/share-export-location: sfs-nas01.ap-southeast-1.myhuaweicloud.com:/share-436304e8
   storage.kubernetes.io/csiProvisionerIdentity: everest-csi-provisioner
  volumeHandle: 682f00bb-ace0-41d8-9b3e-913c9aa6b695
 persistentVolumeReclaimPolicy: Delete
 storageClassName: csi-nas
```

Pay attention to the fields in bold and red. The parameters are described as follows:

Parameter	Description
name	Name of the PV, which must be unique in the cluster.
storage	File storage size in the unit of Gi. Use the value of spec.capacity.storage of the FlexVolume PV.
driver	Storage driver used to attach the volume. Set the driver to nas.csi.everest.io for the file system.

Table 11-9 SFS volume configuration parameters

Parameter	Description
everest.io/share- export-location	Shared path of the file system. Use the value of spec.flexVolume.options.deviceMountPath of the FlexVolume PV.
volumeHandle	File system ID. Use the value of spec.flexVolume.options.volumeID of the FlexVolume PV.
storageClassName	Name of the Kubernetes storage class. Set this field to csi- nas .

Configuration example of a PV for an OBS volume:

apiVersion: v1
kind: PersistentVolume
metadata:
name: pv-obs-example
annotations:
pv.kubernetes.io/provisioned-by: everest-csi-provisioner
spec:
accessModes:
- ReadWriteMany
capacity:
storage: 1Gi
csi:
driver: obs.csi.everest.io
fsType: s3fs
volumeAttributes:
everest.io/obs-volume-type: STANDARD
everest.io/region: ap-southeast-1
storage.kubernetes.io/csiProvisionerIdentity: everest-csi-provisioner
volumeHandle: obs-normal-static-pv
persistentVolumeReclaimPolicy: Delete
storageClassName: csi-obs

Parameter	Description
name	Name of the PV, which must be unique in the cluster.
storage	Storage capacity, in the unit of Gi. Set this parameter to the fixed value 1Gi .
driver	Storage driver used to attach the volume. Set the driver to obs.csi.everest.io for the OBS volume.

Table 11-10 OBS volume	e configuration	parameters
------------------------	-----------------	------------

Parameter	Description
fsТуре	File type. Value options are obsfs or s3fs . If the value is s3fs , an OBS bucket is created and mounted using s3fs. If the value is obsfs , an OBS parallel file system is created and mounted using obsfs. Set this parameter according to the value of spec.flexVolume.options.posix of the FlexVolume PV. If the value of spec.flexVolume.options.posix is true , set this parameter to obsfs . If the value is false , set this parameter to s3fs .
everest.io/obs- volume-type	Storage class, including STANDARD (standard bucket) and WARM (infrequent access bucket). Set this parameter according to the value of spec.flexVolume.options.storage_class of the FlexVolume PV. If the value of spec.flexVolume.options.storage_class is standard , set this parameter to STANDARD . If the value is standard_ia , set this parameter to WARM .
everest.io/region	Region where the OBS bucket is located. Use the value of spec.flexVolume.options.region of the FlexVolume PV.
volumeHandle	OBS bucket name. Use the value of spec.flexVolume.options.volumeID of the FlexVolume PV.
storageClassName	Name of the Kubernetes storage class. Set this field to csi-obs .

Configuration example of a PV for an SFS Turbo volume:

apiVersion: v1
kind: PersistentVolume
metadata:
name: pv-efs-example
annotations:
pv.kubernetes.io/provisioned-by: everest-csi-provisioner
spec:
accessModes:
- ReadWriteMany
capacity:
storage: 10Gi
csi:
driver: sfsturbo.csi.everest.io
fsType: nfs
volumeAttributes:
everest.io/share-export-location: 192.168.0.169:/
storage.kubernetes.io/csiProvisionerIdentity: everest-csi-provisioner
volumeHandle: 8962a2a2-a583-4b7f-bb74-fe76712d8414
persistentVolumeReclaimPolicy: Delete
storageClassName: csi-sfsturbo

	-
Parameter	Description
name	Name of the PV, which must be unique in the cluster.
storage	File system size. Use the value of spec.capacity.storage of the FlexVolume PV.
driver	Storage driver used to attach the volume. Set it to sfsturbo.csi.everest.io .
everest.io/share- export-location	Shared path of the SFS Turbo volume. Use the value of spec.flexVolume.options.deviceMountPath of the FlexVolume PV.
volumeHandle	SFS Turbo volume ID. Use the value of spec.flexVolume.options.volumeID of the FlexVolume PV.
storageClassName	Name of the Kubernetes storage class. Set this field to csi-sfsturbo for SFS Turbo volumes.

Table 11-11 SFS Turbo volume configuration parameters

Step 3 Configure a YAML file of the PVC in the CSI format according to the PVC in the FlexVolume format and associate the PVC with the PV created in **Step 2**.

To be specific, run the following commands to configure the pvc-example.yaml file, which is used to create a PVC.

touch pvc-example.yaml

vi pvc-example.yaml

Configuration example of a PVC for an EVS volume:

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
 labels:
  failure-domain.beta.kubernetes.io/region: ap-southeast-1
  failure-domain.beta.kubernetes.io/zone: <zone name>
 annotations:
  everest.io/disk-volume-type: SAS
  volume.beta.kubernetes.io/storage-provisioner: everest-csi-provisioner
 name: pvc-evs-example
 namespace: default
spec:
 accessModes:
 - ReadWriteOnce
 resources:
  requests:
   storage: 10Gi
 volumeName: pv-evs-example
 storageClassName: csi-disk
```

Parameter	Description
failure- domain.beta.kuber netes.io/region	Region where the cluster is located. Use the same value as that of the FlexVolume PVC.
failure- domain.beta.kuber netes.io/zone	AZ where the EVS disk is deployed. Use the same value as that of the FlexVolume PVC.
everest.io/disk- volume-type	Storage class of the EVS disk. The value can be SAS or SSD . Set this parameter to the same value as that of the PV created in Step 2 .
name	PVC name, which must be unique in the namespace. The value must be unique in the namespace. (If the PVC is dynamically created by a stateful application, the value of this parameter must be the same as the name of the FlexVolume PVC.)
namespace	Namespace to which the PVC belongs. Use the same value as that of the FlexVolume PVC.
storage	Requested capacity of the PVC, which must be the same as the storage size of the existing PV.
volumeName	Name of the PV. Set this parameter to the name of the static PV in Step 2 .
storageClassName	Name of the Kubernetes storage class. Set this field to csi- disk for EVS disks.

Table 11-12 PVC configuration	parameters for an EVS volume
-------------------------------	------------------------------

Configuration example of a PVC for an SFS volume:

apiVersion: v1 kind: PersistentVolumeClaim metadata: annotations: volume.beta.kubernetes.io/storage-provisioner: everest-csi-provisioner name: pvc-sfs-example namespace: default spec: accessModes: - ReadWriteMany resources: requests: storage: 10Gi storageClassName: csi-nas volumeName: pv-sfs-example

Parameter	Description
name	PVC name, which must be unique in the namespace. The value must be unique in the namespace. (If the PVC is dynamically created by a stateful application, the value of this parameter must be the same as the name of the FlexVolume PVC.)
namespace	Namespace to which the PVC belongs. Use the same value as that of the FlexVolume PVC.
storage	Storage capacity, in the unit of Gi. The value must be the same as the storage size of the existing PV.
storageClassName	Set this field to csi-nas .
volumeName	Name of the PV. Set this parameter to the name of the static PV in Step 2 .

Fable 11-13 PVC of	configuration	parameters f	or an	SFS volume
---------------------------	---------------	--------------	-------	------------

Configuration example of a PVC for an OBS volume:

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
 annotations:
  volume.beta.kubernetes.io/storage-provisioner: everest-csi-provisioner
  everest.io/obs-volume-type: STANDARD
 csi.storage.k8s.io/fstype: s3fs
 name: pvc-obs-example
namespace: default
spec:
 accessModes:
 - ReadWriteMany
 resources:
  requests:
   storage: 1Gi
 storageClassName: csi-obs
volumeName: pv-obs-example
```

Parameter	Description
everest.io/obs- volume-type	OBS volume type, which can be STANDARD (standard bucket) and WARM (infrequent access bucket). Set this parameter to the same value as that of the PV created in Step 2 .
csi.storage.k8s.io/ fstype	File type, which can be obsfs or s3fs . The value must be the same as that of fsType of the static OBS volume PV.

Parameter	Description
name	PVC name, which must be unique in the namespace. The value must be unique in the namespace. (If the PVC is dynamically created by a stateful application, the value of this parameter must be the same as the name of the FlexVolume PVC.)
namespace	Namespace to which the PVC belongs. Use the same value as that of the FlexVolume PVC.
storage	Storage capacity, in the unit of Gi. Set this parameter to the fixed value 1Gi .
storageClassName	Name of the Kubernetes storage class. Set this field to csi-obs .
volumeName	Name of the PV. Set this parameter to the name of the static PV created in Step 2 .

Configuration example of a PVC for an SFS Turbo volume:

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
annotations:
volume.beta.kubernetes.io/storage-provisioner: everest-csi-provisioner
name: pvc-efs-example
namespace: default
spec:
accessModes:
- ReadWriteMany
resources:
requests:
storage: 10Gi
storageClassName: csi-sfsturbo
volumeName: pv-efs-example
```

Parameter	Description
name	PVC name, which must be unique in the namespace. The value must be unique in the namespace. (If the PVC is dynamically created by a stateful application, the value of this parameter must be the same as the name of the FlexVolume PVC.)
namespace	Namespace to which the PVC belongs. Use the same value as that of the FlexVolume PVC.
storageClassName	Name of the Kubernetes storage class. Set this field to csi-sfsturbo .

 Table 11-15 PVC configuration parameters for an SFS Turbo volume

Parameter	Description
storage	Storage capacity, in the unit of Gi. The value must be the same as the storage size of the existing PV.
volumeName	Name of the PV. Set this parameter to the name of the static PV created in Step 2 .

Step 4 Upgrade the workload to use a new PVC.

For Deployments

 Run the kubectl create -f commands to create a PV and PVC. kubectl create -f pv-example.yaml kubectl create -f pvc-example.yaml

NOTE

Replace the example file name **pvc-example.yaml** in the preceding commands with the names of the YAML files configured in **Step 2** and **Step 3**.

 Go to the CCE console. On the workload upgrade page, click Upgrade > Advanced Settings > Data Storage > Cloud Storage.

Advanced Settings							
Lifecycle Health Check Environment Variables Data Storage Security Context Log Policies							
The local and cloud volumes must be mounted to the container for persistent data storage.							
Local Volume Cloud Volume							
Name	Туре	Sub-Type	Storage Capac	subPath ⑦	Container Path	Permission	Operation
				No data available.			
Add Cloud Volume							

- 3. Uninstall the old storage and add the PVC in the CSI format. Retain the original mounting path in the container.
- 4. Click Submit.
- 5. Wait until the pods are running.

For StatefulSets that use existing storage

1. Run the **kubectl create -f** commands to create a PV and PVC.

kubectl create -f pv-example.yaml

kubectl create -f pvc-example.yaml

NOTE

Replace the example file name **pvc-example.yaml** in the preceding commands with the names of the YAML files configured in **Step 2** and **Step 3**.

2. Run the **kubectl edit** command to edit the StatefulSet and use the newly created PVC.

kubectl edit sts sts-example -n xxx

30	pod. alpna. Rubernetes. 10/ initialized: true
	spec:
	volumes:
	- name: cce-efs-import-kjxmtzqn-z65j
	persistentVolumeClaim:
	<pre>claimName: pvc-csi-sfsturbo-f2ed93a7-468c-49c3-9a8b-9ded5c6e1533-1</pre>
	conteinere:

D NOTE

Replace **sts-example** in the preceding command with the actual name of the StatefulSet to upgrade. **xxx** indicates the namespace to which the StatefulSet belongs.

3. Wait until the pods are running.

NOTE

The current console does not support the operation of adding new cloud storage for StatefulSets. Use the kubectl commands to replace the storage with the newly created PVC.

For StatefulSets that use dynamically allocated storage

- Back up the PV and PVC in the flexVolume format used by the StatefulSet.
 kubectl get pvc xxx -n {namespaces} -oyaml > pvc-backup.yaml
 kubectl get pv xxx -n {namespaces} -oyaml > pv-backup.yaml
- 2. Change the number of pods to **0**.
- 3. On the storage page, disassociate the flexVolume PVC used by the StatefulSet.
- 4. Run the **kubectl create -f** commands to create a PV and PVC.

kubectl create -f pv-example.yaml

kubectl create -f pvc-example.yaml

NOTE

Replace the example file name **pvc-example.yaml** in the preceding commands with the names of the YAML files configured in **Step 2** and **Step 3**.

5. Change the number of pods back to the original value and wait until the pods are running.

NOTE

The dynamic allocation of storage for StatefulSets is achieved by using **volumeClaimTemplates**. This field cannot be modified by Kubernetes. Therefore, data cannot be migrated by using a new PVC.

The PVC naming rule of the **volumeClaimTemplates** is fixed. When a PVC that meets the naming rule exists, this PVC is used.

Therefore, disassociate the original PVC first and then create a PVC with the same name in the CSI format.

6. (Optional) Recreate the stateful application to ensure that a CSI PVC is used when the application is scaled out. Otherwise, FlexVolume PVCs are used in scaling out.

• Run the following command to obtain the YAML file of the StatefulSet:

kubectl get sts xxx -n {namespaces} -oyaml > sts.yaml

• Run the following command to back up the YAML file of the StatefulSet:

cp sts.yaml sts-backup.yaml

 Modify the definition of volumeClaimTemplates in the YAML file of the StatefulSet.

vi sts.yaml

Configuration example of volumeClaimTemplates for an EVS volume:

```
volumeClaimTemplates:

- metadata:

name: pvc-161070049798261342

namespace: default

creationTimestamp: null

annotations:

everest.io/disk-volume-type: SAS

spec:

accessModes:

- ReadWriteOnce

resources:

requests:

storage: 10Gi

storageClassName: csi-disk
```

The parameter value must be the same as the PVC of the EVS volume created in **Step 3**.

Configuration example of volumeClaimTemplates for an SFS volume:

```
volumeClaimTemplates:

- metadata:

name: pvc-161063441560279697

namespace: default

creationTimestamp: null

spec:

accessModes:

- ReadWriteMany

resources:

requests:

storage: 10Gi

storageClassName: csi-nas
```

The parameter value must be the same as the PVC of the SFS volume created in **Step 3**.

Configuration example of volumeClaimTemplates for an OBS volume:

```
volumeClaimTemplates:

- metadata:

name: pvc-161070100417416148

namespace: default

creationTimestamp: null

annotations:

csi.storage.k8s.io/fstype: s3fs

everest.io/obs-volume-type: STANDARD

spec:

accessModes:

- ReadWriteMany

resources:

requests:

storage: 1Gi

storageClassName: csi-obs
```

The parameter value must be the same as the PVC of the OBS volume created in **Step 3**.

• Delete the StatefulSet.

kubectl delete sts xxx -n {namespaces}

• Create the StatefulSet.

kubectl create -f sts.yaml

Step 5 Check service functions.

- 1. Check whether the application is running properly.
- 2. Checking whether the data storage is normal.

NOTE

If a rollback is required, perform **Step 4**. Select the PVC in FlexVolume format and upgrade the application.

Step 6 Uninstall the PVC in the FlexVolume format.

If the application functions normally, unbind the PVC in the FlexVolume format on the storage management page.

You can also run the kubectl command to delete the PVC and PV of the FlexVolume format.

Before deleting a PV, change the persistentVolumeReclaimPolicy of the PV to **Retain**. Otherwise, the underlying storage will be reclaimed after the PV is deleted.

If the cluster has been upgraded before the storage migration, PVs may fail to be deleted. You can remove the PV protection field **finalizers** to delete PVs.

kubectl patch pv {pv_name} -p '{"metadata":{"finalizers":null}}'

----End

11.5 Using Custom Storage Classes

Background

When using storage resources in CCE, the most common method is to specify **storageClassName** to define the type of storage resources to be created when creating a PVC. The following configuration shows how to use a PVC to apply for an SAS (high I/O) EVS disk (block storage).

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
name: pvc-evs-example
namespace: default
annotations:
everest.io/disk-volume-type: SAS
spec:
accessModes:
- ReadWriteOnce
resources:
requests:
storage: 10Gi
storageClassName: csi-disk
```

To specify the EVS disk type, you can configure the **everest.io/disk-volume-type** field. The value **SAS** is used as an example here, indicating the high I/O EVS disk type. Or you can choose **SSD** (ultra-high I/O).

This configuration method may not work if you want to:

- Set storageClassName only, which is simpler than specifying the EVS disk type by using everest.io/disk-volume-type.
- Avoid modifying YAML files or Helm charts. Some users switch from self-built or other Kubernetes services to CCE and have written YAML files of many applications. In these YAML files, different types of storage resources are specified by different StorageClassNames. When using CCE, they need to modify a large number of YAML files or Helm charts to use storage resources, which is labor-consuming and error-prone.
- Set the default storageClassName for all applications to use the default storage class. In this way, you can create storage resources of the default type without needing to specify storageClassName in the YAML file.

Solution

This section describes how to set a custom storage class in CCE and how to set the default storage class. You can specify different types of storage resources by setting **storageClassName**.

• For the first scenario, you can define custom storageClassNames for SAS and SSD EVS disks. For example, define a storage class named **csi-disk-sas** for creating SAS disks. The following figure shows the differences before and after you use a custom storage class.



- For the second scenario, you can define a storage class with the same name as that in the existing YAML file without needing to modify **storageClassName** in the YAML file.
- For the third scenario, you can set the default storage class as described below to create storage resources without specifying **storageClassName** in YAML files.

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
name: pvc-evs-example
namespace: default
spec:
accessModes:
- ReadWriteOnce
resources:
requests:
storage: 10Gi
```

Creating a StorageClass Using a YAML File

As of now, CCE provides StorageClasses such as csi-disk, csi-nas, and csi-obs by default. When defining a PVC, you can use a **StorageClassName** to automatically create a PV of the corresponding type and automatically create underlying storage resources.

Run the following kubectl command to obtain the StorageClasses that CCE supports. Use the CSI add-on provided by CCE to create a StorageClass.

# kubectl get sc		
NAME PROVISIONER	AGE	
csi-disk everest-csi-provisioner	17d	# EVS disk
csi-disk-topology everest-csi-provisioner	17d	# EVS disks created with a delay
csi-nas everest-csi-provisioner	17d	# SFS 1.0
csi-sfs everest-csi-provisioner	17d	# A general purpose file system (SFS 3.0 Capacity
Oriented)		
csi-obs everest-csi-provisioner	17d	# OBS
csi-sfsturbo everest-csi-provisioner	17d	# SFS Turbo
csi-local everest-csi-provisioner	17d	# Local PV
csi-local-topology everest-csi-provisioner	· 17d	# Local PV created with a delay
csi-disk-dss everest-csi-provisioner	2d21h	# DSS

Each StorageClass contains the default parameters used for dynamically creating a PV. The following is an example of StorageClass for EVS disks:

kind: StorageClass
apiVersion: storage.k8s.io/v1
metadata:
name: csi-disk
provisioner: everest-csi-provisioner
parameters:
csi.storage.k8s.io/csi-driver-name: disk.csi.everest.io
csi.storage.k8s.io/fstype: ext4
everest.io/disk-volume-type: SAS
everest.io/passthrough: 'true'
reclaimPolicy: Delete
allowVolumeExpansion: true
volumeBindingMode: Immediate

Table 11-16 Key parameters

Parameter	Description
provisioner	Specifies the storage resource provider, which is the Everest add- on for CCE. Set this parameter to everest-csi-provisioner .
parameters	Specifies the storage parameters, which vary with StorageClasses. For details, see Table 11-17.

Parameter	Description		
reclaimPolicy	Specifies the value of persistentVolumeReclaimPolicy for creating a PV. The value can be Delete or Retain . If reclaimPolicy is not specified when a StorageClass object is created, the value defaults to Delete .		
	• Delete : When a PVC is deleted, its associated underlying storage resources will be deleted and the PV resources will be removed. Exercise caution if you select this option.		
	• Retain : When a PVC is deleted, both of the PV and its associated underlying storage resources will be retained and the PV is marked as released. If you manually delete the PV afterwards, the underlying storage resources will not be deleted. To bind the PV to a new PVC, you need to remove the original binding information from the PV.		
allowVolume Expansion	Specifies whether the PV of this StorageClass supports dynamic capacity expansion. The default value is false . Dynamic capacity expansion is implemented by the underlying storage add-on. This is only a switch.		
volumeBindi ngMode Specifies the volume binding mode, which is the time v is dynamically created. The value can be Immediate or WaitForFirstConsumer .			
	• Immediate: After a PVC is created, the storage resources and PV will be created and associated with the PVC without delay.		
	• WaitForFirstConsumer: After a PVC is created, it will not be immediately bound to a PV. Instead, the storage resources and PV will be generated and bound to the PVC only after the pod that requires the PVC is scheduled.		
mountOptio ns	This field must be supported by the underlying storage. If this field is not supported but is specified, the PV creation will fail.		

Table 11-17 Parameters

Volum e Type	Parameter	Mandat ory	Description
EVS	csi.storage.k8s.io/ csi-driver-name	Yes	Driver type. If an EVS disk is used, the parameter value is fixed at disk.csi.everest.io .
EVS	csi.storage.k8s.io/ fstype	Yes	If an EVS disk is used, the parameter value can be ext4 .

Volum e Type	Parameter	Mandat ory	Description
EVS	everest.io/disk- volume-type	Yes	 EVS disk type. All letters are in uppercase. SAS: high I/O SSD: ultra-high I/O GPSSD: general-purpose SSD ESSD: extreme SSD GPSSD2: general-purpose SSD v2, which is supported when the Everest version is 2.4.4 or later and the everest.io/disk-throughput annotations are configured
EVS	everest.io/ passthrough	Yes	The parameter value is fixed at true , which indicates that the EVS device type is SCSI . No other parameter values are allowed.
EVS	everest.io/disk- iops	No	 Preconfigured IOPS, which is supported only by general-purpose SSD v2 and extreme SSD v2 EVS disks. The IOPS of general-purpose SSD v2 EVS disks ranges from 3000 to 128000, and the maximum value is 500 times of the capacity (GiB). If the IOPS of general-purpose SSD v2 disks is greater than 3000, extra IOPS will be billed. For details, see Price Calculator. The IOPS of extreme SSD v2 disks ranges from 100 to 256000, and the maximum value is 1000 times of the capacity (GiB). The IOPS of extreme SSD v2 disks will be billed separately. For details, see Price Calculator.
EVS	everest.io/disk- throughput	No	Preconfigured throughput, which is supported only by general-purpose SSD v2 EVS disks. The value ranges from 125 MiB/s to 1000 MiB/s. The maximum value is a quarter of IOPS. If the throughput is greater than 125 MiB/s, extra throughput will be billed. For details, see Price Calculator .

Volum e Type	Parameter	Mandat ory	Description
SFS	csi.storage.k8s.io/ csi-driver-name	Yes	Driver type. If SFS is used, the parameter value is fixed at nas.csi.everest.io .
SFS	csi.storage.k8s.io/ fstype	Yes	If SFS is used, the value can be nfs .
SFS	everest.io/share- access-level	Yes	The parameter value is fixed at rw , indicating that the SFS data is readable and writable.
SFS	everest.io/share- access-to	Yes	VPC ID of the cluster.
SFS	everest.io/share- is-public	No	The parameter value is fixed at false , indicating that the file is shared to private. When you use a general purpose file system (SFS 3.0 Capacity-Oriented), there is no need to configure this parameter.
SFS	everest.io/sfs- version	No	This parameter is only required for general purpose file systems (SFS 3.0 Capacity-Oriented) and its value is fixed at sfs3.0 .
SFS Turbo	csi.storage.k8s.io/ csi-driver-name	Yes	Driver type. If SFS Turbo is used, the parameter value is fixed at sfsturbo.csi.everest.io .
SFS Turbo	csi.storage.k8s.io/ fstype	Yes	If SFS Turbo is used, the value can be nfs .
SFS Turbo	everest.io/share- access-to	Yes	VPC ID of the cluster.
SFS Turbo	everest.io/share- expand-type	No	Extension type. The default value is bandwidth , indicating an enhanced file system. This parameter does not take effect.
SFS Turbo	everest.io/share- source	Yes	The parameter value is fixed at sfs- turbo .
SFS Turbo	everest.io/share- volume-type	No	SFS Turbo StorageClass. The default value is STANDARD , indicating standard and standard enhanced editions. This parameter does not take effect.

Volum e Type	Parameter	Mandat ory	Description
SFS Turbo	everest.io/ reclaim-policy	No	 Whether to retain subdirectories when deleting a PVC. This parameter must be used with reclaimPolicy. This parameter is available only when the PV reclaim policy is Delete. Options: retain-volume-only: When a PVC is deleted, the PV is deleted, but its associated subdirectories are retained. delete: When a PVC is deleted, the PV and its associated subdirectories are also deleted. NOTE When a subdirectory is deleted, only the absolute path of the subdirectory
			specified in the PVC is deleted. The parent directory remains unchanged.
OBS	csi.storage.k8s.io/ csi-driver-name	Yes	Driver type. If OBS is used, the parameter value is fixed at obs.csi.everest.io .
OBS	csi.storage.k8s.io/ fstype	Yes	 Instance type, which can be obsfs or s3fs. obsfs: a parallel file system s3fs: object bucket
OBS	everest.io/obs- volume-type	Yes	 OBS StorageClass. If fsType is set to s3fs, standard buckets (STANDARD) and infrequent access buckets (WARM) are supported. This parameter is invalid when fsType is set to obsfs.
Local PV	csi.storage.k8s.io/ csi-driver-name	Yes	Driver type. If a local PV is used, the parameter value is fixed at local.csi.everest.io .
Local PV	csi.storage.k8s.io/ fstype	Yes	File system type. The value can only be ext4 .
Local PV	volume-type	Yes	Volume type. The value can only be persistent .
DSS	csi.storage.k8s.io/ csi-driver-name	Yes	Driver type. If DSS is used, the parameter value is fixed at disk.csi.everest.io .

Volum e Type	Parameter	Mandat ory	Description
DSS	csi.storage.k8s.io/ fstype	Yes	File system type. The value can only be ext4 .
DSS	everest.io/disk- volume-type	Yes	Dedicated storage, which must be specified in uppercase. • SAS: high I/O • SSD: ultra-high I/O
DSS	everest.io/ passthrough	Yes	The parameter value is fixed at true , which indicates that the DSS device type is SCSI. No other values are allowed.
DSS	everest.io/ resource-type	Yes	Resource type. The parameter value is fixed at dss .

Custom Storage Classes

You can customize a high I/O storage class in a YAML file. For example, the name **csi-disk-sas** indicates that the disk type is SAS (high I/O).

apiVersion: storage.k8s.io/v1 kind: StorageClass	
metadata:	
name: csi-disk-sas	# Name of the high I/O storage class, which can be customized.
parameters:	
csi.storage.k8s.io/csi-driver-name: c csi.storage.k8s.io/fstype: ext4	disk.csi.everest.io
everest.io/disk-volume-type: SAS everest.io/passthrough: "true" provisioner: everest-csi-provisioner	# High I/O EVS disk type, which cannot be customized.
reclaimPolicy: Delete volumeBindingMode: Immediate	
allowVolumeExpansion: true	# true indicates that capacity expansion is allowed.

For an ultra-high I/O storage class, you can set the class name to **csi-disk-ssd** to create SSD EVS disk (ultra-high I/O).

apiVersion: storage.k8s.io/v1	
kind: StorageClass	
metadata:	
name: csi-disk-ssd	# Name of the ultra-high I/O storage class, which can be customized.
parameters:	
csi.storage.k8s.io/csi-driver-name:	: disk.csi.everest.io
csi.storage.k8s.io/fstype: ext4	
everest.io/disk-volume-type: SSD	# Ultra-high I/O EVS disk type, which cannot be customized.
everest.io/passthrough: "true"	
provisioner: everest-csi-provisioner	
reclaimPolicy: Delete	
volumeBindingMode: Immediate	
allowVolumeExpansion: true	

reclaimPolicy: indicates the recycling policies of the underlying cloud storage. The value can be **Delete** or **Retain**.

• **Delete**: When a PVC is deleted, both the PV and the EVS disk are deleted.

 Retain: When a PVC is deleted, the PV and underlying storage resources are not deleted. Instead, you must manually delete these resources. After that, the PV resource is in the Released state and cannot be bound to the PVC again.

NOTE

The reclamation policy configured here has no impact on the SFS Turbo storage and the SFS Turbo resources will not be reclaimed.

If high data security is required, you are advised to select **Retain** to prevent data from being deleted by mistake.

After the definition is complete, run the **kubectl create** commands to create storage resources.

kubectl create -f sas.yaml
storageclass.storage.k8s.io/csi-disk-sas created
kubectl create -f ssd.yaml
storageclass.storage.k8s.io/csi-disk-ssd created

Query the storage class again. Two more types of storage classes are displayed in the command output, as shown below.

# kubectl get sc	
NAME PROVISIONER	AGE
csi-disk everest-csi-provisioner	17d
csi-disk-sas everest-csi-provisioner	2m28s
csi-disk-ssd everest-csi-provisioner	16s
csi-disk-topology everest-csi-provisioner	17d
csi-nas everest-csi-provisioner	17d
csi-obs everest-csi-provisioner	17d
csi-sfsturbo everest-csi-provisioner	17d

Other types of storage resources can be defined in the similar way. You can use kubectl to obtain the YAML file and modify it as required.

|--|

•	The storage
	# kubectl get sc csi-nas -oyaml
	nilu. Julageciass
	apiversioni, sociage.kos.io/vir
	name rsi-nas
	novisioner: everest-csi-provisioner
	parameters:
	csi.storage.k8s.io/csi-driver-name: nas.csi.everest.io csi.storage.k8s.io/fstype: nfs everest.io/share-access-level: rw everest.io/share-access.to: 5e3864c6-e78d-4400-b6fd-de09d432c632 # ID of the VPC to which the
	cluster belongs everest.io/share-is-public: 'false' everest.io/zone: xxxxx # AZ reclaimPolicy: Delete allowVolumeExpansion: true volumeBindingMode: Immediate
•	Object storage
	<pre># kubectl get sc csi-obs -oyaml kind: StorageClass apiVersion: storage.k8s.io/v1 metadata: name: csi-obs provisioner: everest-csi-provisioner parameters: csi.storage.k8s.io/csi-driver-name: obs.csi.everest.io</pre>
	csi.storage.k8s.io/fstype: s3fs # Object storage type. s3fs indicates an object bucket, and obsfs indicates a parallel file system.

everest.io/obs-volume-type: STANDARD # Storage class of the OBS bucket reclaimPolicy: Delete volumeBindingMode: Immediate

Specifying an Enterprise Project for Storage Classes

CCE allows you to specify an enterprise project when creating EVS disks and OBS PVCs. The created storage resources (EVS disks and OBS) belong to the specified enterprise project. The enterprise project can be the enterprise project to which the cluster belongs or the default enterprise project.

If you do not specify any enterprise project, the enterprise project in StorageClass is used by default. The created storage resources by using the csi-disk and csi-obs storage classes of CCE belong to the default enterprise project.

If you want the storage resources created from the storage classes to be in the same enterprise project as the cluster, you can customize a storage class and specify the enterprise project ID, as shown below.

NOTE

To use this function, the everest add-on must be upgraded to 1.2.33 or later.

kind: StorageClass apiVersion: storage.k8s.io/v1 metadata: name: csi-disk-epid #Customize a storage class name. provisioner: everest-csi-provisioner parameters: csi.storage.k8s.io/csi-driver-name: disk.csi.everest.io csi.storage.k8s.io/fstype: ext4 everest.io/disk-volume-type: SAS everest.io/enterprise-project-id: 86bfc701-9d9e-4871-a318-6385aa368183 #Specify the enterprise project ID. everest.io/passthrough: 'true' reclaimPolicy: Delete allowVolumeExpansion: true volumeBindingMode: Immediate

Specifying a Default Storage Class

You can specify a storage class as the default class. In this way, if you do not specify **storageClassName** when creating a PVC, the PVC is created using the default storage class.

For example, to specify **csi-disk-ssd** as the default storage class, edit your YAML file as follows:

apiVersion: storage.k8s.io/v1 kind: StorageClass metadata: name: csi-disk-ssd annotations: storageclass.kubernetes.io/is-default-class: "true" # Specifies the default storage class in a cluster. A cluster can have only one default storage class. parameters: csi.storage.k8s.io/csi-driver-name: disk.csi.everest.io csi.storage.k8s.io/fstype: ext4 everest.io/disk-volume-type: SSD everest.io/passthrough: "true" provisioner: everest-csi-provisioner reclaimPolicy: Delete volumeBindingMode: Immediate allowVolumeExpansion: true

Delete the created csi-disk-ssd disk, run the **kubectl create** command to create a csi-disk-ssd disk again, and then query the storage class. The following information is displayed.

# kubectl delete sc csi-disk-ssd							
storageclass.storage.k8s.io "csi-disk-ssd" deleted							
# kubectl create -f	ssd.yaml						
storageclass.storag	e.k8s.io/csi-disk-ssd created						
# kubectl get sc							
NAME	PROVISIONER	AGE					
csi-disk	everest-csi-provisioner	17d					
csi-disk-sas	everest-csi-provisioner	114m					
csi-disk-ssd (default) everest-csi-provisioner 9							
csi-disk-topology	everest-csi-provisioner	17d					
csi-nas	everest-csi-provisioner	17d					
csi-obs	everest-csi-provisioner	17d					
csi-sfsturbo	everest-csi-provisioner	17d					

Verification

When creating a PVC, you have the following configuration methods:

- Specify a storage class explicitly.
- Use the default storage class set up in the cluster if you do not specify a storage class.

Below are the steps to check the results of both configuration methods.

The following steps are used to verify whether the storage class **csi-disk-sas** defined in "Custom Storage Classes" can be used to create a PVC.

 Create a YAML file named sas-disk.yaml. You can change the file name as needed. This file uses csi-disk-sas to create a PVC. vim sas-disk.yaml

The file content is as follows: apiVersion: v1 kind: PersistentVolumeClaim metadata: name: sas-disk spec: accessModes: - ReadWriteOnce resources: requests: storage: 10Gi storageClassName: csi-disk-sas

2. Create a PVC.

kubectl create -f sas-disk.yaml

Information similar to the following is displayed:

- persistentvolumeclaim/sas-disk created
- 3. Check the PVC information. kubectl get pvc

If information similar to the following is displayed, a PV has been associated with the PVC and the storage class is **csi-disk-sas**.

NAME **STATUS** VOLUME CAPACITY ACCESS MODES **STORAGECLASS** AGE sas-disk **Bound** pvc-6e2f37f9-7346-4419-82f7-b42e79f7964c 10Gi RWO **csi-disk-sas** 24s

4. Check the PV information. kubectl get pv

If information similar to the following is displayed, the PV has been automatically created.

NAME CAPACITY ACCESS MODES RECLAIM POLICY STATUS CLAIM **STORAGECLASS** REASON AGE pvc-6e2f37f9-7346-4419-82f7-b42e79f7964c 10Gi RWO Delete Bound default/ sas-disk **csi-disk-sas** 30s

The result shows that the custom storage class **csi-disk-sas** can be used properly.

The following steps are used to verify whether a PVC can be created without specifying a storage class.

1. Create a YAML file named **sas-disk.yaml**. You can change the file name as needed. When the file is used to create a PVC, no storage class is specified. vim *sas-disk.yaml*

The file content is as follows: apiVersion: v1 kind: PersistentVolumeClaim metadata: name: ssd-disk spec: accessModes: - ReadWriteOnce resources: requests: storage: 10Gi

2. Create a PVC.

kubectl create -f sas-disk.yaml

Information similar to the following is displayed:

persistentvolumeclaim/sas-disk created

3. Check the PVC information. kubectl get pvc

If information similar to the following is displayed, a PV has been bound to the PVC and the default storage class of the EVS disk is used.

```
NAME STATUS VOLUME CAPACITY ACCESS MODES
STORAGECLASS AGE
sas-disk Bound pvc-6e2f37f9-7346-4419-82f7-b42e79f7964c 10Gi RWO csi-disk-sas
16m
ssd-disk Bound pvc-4d2b059c-0d6c-44af-9994-f74d01c78731 10Gi RWO csi-disk-
ssd 10s
```

4. Check the PV information.

kubectl get pv

If information similar to the following is displayed, the PV has been automatically created and the default storage class of the EVS disk is used.

NAME		CAPACITY	ACCESS	MODES	RECLAIM POLICY	STATUS	
CLAIM	STORAGECL	ASS REASC	ON AGE				
pvc-4d2b059c-0d6	6c-44af-9994-f7	4d01c78731	10Gi	RWO	Delete	Bound	
default/ssd-disk	csi-disk-sso	l 159	5				
pvc-6e2f37f9-734	6-4419-82f7-b4	2e79f7964c	10Gi	RWO	Delete	Bound	default/
sas-disk csi	-disk-sas	17m					

The result shows that if no storage class is specified, the system will automatically use the default storage class.

11.6 Scheduling EVS Disks Across AZs Using csi-disktopology

Background

EVS disks cannot be attached to a node deployed in another AZ. For example, the EVS disks in AZ 1 cannot be attached to a node in AZ 2. If the storage class csidisk is used for StatefulSets, when a StatefulSet is scheduled, a PVC and a PV are created immediately (an EVS disk is created along with the PV), and then the PVC is bound to the PV. However, when the cluster nodes are located in multiple AZs, the EVS disk created by the PVC and the node to which the pods are scheduled may be in different AZs. As a result, the pods fail to be scheduled.



Solution

CCE provides a storage class named **csi-disk-topology**, which is a late-binding EVS disk type. When you use this storage class to create a PVC, no PV will be created in pace with the PVC. Instead, the PV is created in the AZ of the node where the pod will be scheduled. An EVS disk is then created in the same AZ to ensure that the EVS disk can be attached and the pod can be successfully scheduled.



Failed Pod Scheduling Due to csi-disk Used in Cross-AZ Node Deployment

Create a cluster with three nodes in different AZs.

Use the csi-disk storage class to create a StatefulSet and check whether the workload is successfully created.

apiVersion: apps/v1	
metadata	
neldudid:	
spec.	# Name of the headless Service
renlicas: 4	# Name of the headless service
selector:	
matchl abels:	
app: nginx	
template:	
metadata:	
labels:	
app: nginx	
spec:	
containers:	
- name: container-0	
image: nginx:alpine	
resources:	
limits:	
cpu: 600m	
memory: 200Mi	
requests:	
cpu: 600m	
memory: 200Mi	
volumeMounts:	# Storage mounted to the pod
- name: data	
mountPath: /usr/share/nginx/ht	ml # Mount the storage to /usr/share/nginx/html .
imagePullSecrets:	
- name: default-secret	
volumeClaim Templates:	
- metadata:	
name: data	
annotations:	
everest.io/uisk-voluitie-type. SAS	
spec.	
- ReadWriteOnce	
- Read WhiteOnce	
requests:	
storage: 1Gi	
storageClassName: csi-disk	

The StatefulSet uses the following headless Service.

apiVersion: v1	
kind: Service metadata: name: nginx	# Object type (Service)
labels:	
app: nginx	
spec:	
ports:	
- name: nginx	# Name of the port for communication between pods
port: 80	# Port number for communication between pods
selector:	
app: nginx	# Select the pod whose label is app:nginx.
clusterIP: None	# Set this parameter to None , indicating the headless Service.

After the creation, check the PVC and pod status. In the following output, the PVC has been created and bound successfully, and a pod is in the Pending state.

kubectl get nyc -owide

STA	TUS VC	OLUME			CAPACITY	ACCES	S MODES	STORAG	ECLASS
	ODE								
x-0 Bo	und pv	/c-04e259	985-fc9	3-4254-92a	1-1085ce19d31	e 1Gi	RWO	CS	i-disk
ystem									
x-1 Bo	und pv	/c-0ae633	36b-a2e	ea-4ddc-8f6	3-cfc5f9efe189	1Gi	RWO	csi-	disk
ystem									
x-2 Bo	und pv	/c-aa46f4	52-cc5	b-4dbd-825	a-da68c858720	d 1Gi	RWO	CS	i-disk
ystem									
x-3 Bo	und p	vc-3d60e	532-ff3	1-42df-9e78	3-015cacb18a0b	o 1Gi	RWO	csi	-disk
ystem									
get pod	-owide								
READY	STATU	S REST	ARTS	AGE IP	NODE	N	OMINATE	D NODE	READINESS
		_							
1/1 R	unning	0	2m25s	172.16.0.1	2 192.168.0.1	21 <nc< td=""><td>one></td><td><none></none></td><td></td></nc<>	one>	<none></none>	
1/1 R	unning	0	2m8s	172.16.0.13	36 192.168.0.2	11 <nc< td=""><td>one></td><td><none></none></td><td></td></nc<>	one>	<none></none>	
1/1 R	unning	0	111s	172.16.1.7	192.168.0.240) <non< td=""><td>e> ·</td><td><none></none></td><td></td></non<>	e> ·	<none></none>	
0/1 P	ending	0	95s	<none></none>	<none></none>	<none></none>	> <r< td=""><td>none></td><td></td></r<>	none>	
	STA .UMEM ⁴ <-0 Bo ystem x-1 Bo ystem x-2 Bo ystem x-3 Bo ystem get pod READY 1/1 R 1/1 R 1/1 R 1/1 R	STATUS VC .UMEMODE <-0 Bound pv ystem <-1 Bound pv ystem <-2 Bound pv ystem x-3 Bound p ystem get pod -owide READY STATU 1/1 Running 1/1 Running 1/1 Running 0/1 Pending	STATUS VOLUME .UMEMODE k-0 Bound pvc-04e259 ystem x-1 Bound pvc-0ae633 ystem x-2 Bound pvc-aa46f4 ystem x-3 Bound pvc-3d60e ystem get pod -owide READY STATUS RESTAND 1/1 Running 0 1/1 Running 0 1/1 Running 0 1/1 Running 0 1/1 Running 0 1/1 Running 0 1/1 Running 0	STATUS VOLUME .UMEMODE x-0 Bound pvc-04e25985-fc9 ystem x-1 Bound pvc-0ae6336b-a26 ystem x-2 Bound pvc-aa46f452-cc5 ystem x-3 Bound pvc-3d60e532-ff3 ystem get pod -owide READY STATUS RESTARTS 1/1 Running 0 2m25s 1/1 Running 0 2m8s 1/1 Running 0 111s 0/1 Pending 0 95s	STATUS VOLUME .UMEMODE k-0 Bound pvc-04e25985-fc93-4254-92a ystem k-1 Bound pvc-0ae6336b-a2ea-4ddc-8f6. ystem k-2 Bound pvc-0ae6336b-a2ea-4ddc-8f6. ystem k-2 Bound pvc-aa46f452-cc5b-4dbd-825. ystem x-3 Bound pvc-3d60e532-ff31-42df-9e78 ystem get pod -owide READY STATUS RESTARTS 1/1 Running 0 2m25s 172.16.0.1 1/1 Running 0 2m8s 172.16.0.1 1/1 Running 0 111s 172.16.1.7 0/1 Pending 0 95s <none></none>	STATUS VOLUME CAPACITY .UMEMODE .UMEMODE	STATUS VOLUME CAPACITY ACCESS .UMEMODE .UMEMODE .	STATUS VOLUME CAPACITY ACCESS MODES .UMEMODE .UMEMODE . . .	STATUS VOLUME CAPACITY ACCESS MODES STORAG UMEMODE <-0 Bound pvc-04e25985-fc93-4254-92a1-1085ce19d31e 1Gi RWO cs ystem <-1 Bound pvc-0ae6336b-a2ea-4ddc-8f63-cfc5f9efe189 1Gi RWO csi- ystem <-2 Bound pvc-aa46f452-cc5b-4dbd-825a-da68c858720d 1Gi RWO csi- ystem <-3 Bound pvc-3d60e532-ff31-42df-9e78-015cacb18a0b 1Gi RWO csi- ystem get pod -owide READY STATUS RESTARTS AGE IP NODE NOMINATED NODE 1/1 Running 0 2m25s 172.16.0.12 192.168.0.121 <none> <none> 1/1 Running 0 2m8s 172.16.0.136 192.168.0.211 <none> <none> 1/1 Running 0 111s 172.16.1.7 192.168.0.240 <none> <none> 0/1 Pending 0 95s <none> <none> <none></none></none></none></none></none></none></none></none></none>

The event information of the pod shows that the scheduling fails due to no available node. Two nodes (in AZ 1 and AZ 2) do not have sufficient CPUs, and the created EVS disk is not in the AZ where the third node (in AZ 3) is located. As a result, the pod cannot use the EVS disk.

Check the AZ where the EVS disk created from the PVC is located. It is found that data-nginx-3 is in AZ 1. In this case, the node in AZ 1 has no resources, and only the node in AZ 3 has CPU resources. As a result, the scheduling fails. Therefore, there should be a delay between creating the PVC and binding the PV.

Storage Class for Delayed Binding

If you check the cluster storage class, you can see that the binding mode of csidisk-topology is **WaitForFirstConsumer**, indicating that a PV is created and bound when a pod uses the PVC. That is, the PV and the underlying storage resources are created based on the pod information.

# kubectl get s	torageclass				
NAME	JAME PROVISIONER RECLAIMPOLICY VOLUMEBINDINGMODE				
ALLOWVOLUM	EEXPANSION AGE				
csi-disk	everest-csi-provisioner	Delete	Immediate	true	156m
csi-disk-topolog	gy everest-csi-provisioner	Delete	WaitForFirstCo	nsumer true	
156m					
csi-nas	everest-csi-provisioner	Delete	Immediate	true	156m
csi-obs	everest-csi-provisioner	Delete	Immediate	false	156m
csi-sfsturbo	everest-csi-provisioner	Delete	Immediate	true	156m

VOLUMEBINDINGMODE is displayed if your cluster is v1.19. It is not displayed in clusters of v1.17 or v1.15.

You can also view the binding mode in the csi-disk-topology details.

# kubectl describe	e sc csi-disk-topology
Name:	csi-disk-topology
IsDefaultClass:	No
Annotations:	<none></none>
Provisioner:	everest-csi-provisioner
Parameters:	csi.storage.k8s.io/csi-driver-name=disk.csi.everest.io,csi.storage.k8s.io/
fstype=ext4,evere	st.io/disk-volume-type=SAS,everest.io/passthrough=true
AllowVolumeExpa	nsion: True
MountOptions:	<none></none>
ReclaimPolicy:	Delete
VolumeBindingMo	ode: WaitForFirstConsumer
Events:	<none></none>

Create PVCs of the csi-disk and csi-disk-topology classes. Observe the differences between these two types of PVCs.

```
csi-disk
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
name: disk
 annotations:
  everest.io/disk-volume-type: SAS
spec:
 accessModes:
 - ReadWriteOnce
 resources:
  requests:
   storage: 10Gi
 storageClassName: csi-disk
                                # StorageClass
csi-disk-topology
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
name: topology
 annotations:
  everest.io/disk-volume-type: SAS
spec:
 accessModes:
 - ReadWriteOnce
 resources:
  requests:
   storage: 10Gi
 storageClassName: csi-disk-topology # StorageClass
```

View the PVC details. As shown below, the csi-disk PVC is in Bound state and the csi-disk-topology PVC is in Pending state.

```
# kubectl create -f pvc1.yaml
persistentvolumeclaim/disk created
# kubectl create -f pvc2.yaml
persistentvolumeclaim/topology created
# kubectl get pvc
NAME
            STATUS VOLUME
                                                    CAPACITY ACCESS MODES
STORAGECLASS
                  AGE
          Bound pvc-88d96508-d246-422e-91f0-8caf414001fc 10Gi
disk
                                                                      RWO
                                                                                 csi-disk
18s
topology
            Pending
                                                               csi-disk-topology 2s
```

View details about the csi-disk-topology PVC. You can see that "waiting for first consumer to be created before binding" is displayed in the event, indicating that the PVC is bound after the consumer (pod) is created.

kubectl describe pvc topology Name: topology Namespace: default StorageClass: csi-disk-topology Status: **Pending** Volume: Labels: <none> Annotations: everest.io/disk-volume-type: SAS Finalizers: [kubernetes.io/pvc-protection] Capacity: Access Modes: VolumeMode: Filesystem Used By: <none> Events: Type Reason From Age Message ------------_____

Normal WaitForFirstConsumer 5s (x3 over 30s) persistentvolume-controller waiting for first consumer to be created before binding

Create a workload that uses the PVC. Set the PVC name to topology.

apiVersion: apps/v1	
kind: Deployment	
metadata:	
name: nginx-deployment	
spec:	
selector:	
matchLabels:	
app: nginx	
replicas: 1	
template:	
metadata:	
labels:	
app: nginx	
spec:	
containers:	
- image: nginx:alpine	
name: container-0	
volumeMounts:	
- mountPath: /tmp	# Mount path
name: topology-example	
restartPolicy: Always	
volumes:	
- name: topology-example	
persistentvolumeClaim:	# D) (C
claimName: topology	# PVC name

After the PVC is created, check the PVC details. You can see that the PVC is bound successfully.

kubectl describe pvc topology Name: topology Namespace: default StorageClass: csi-disk-topology Status: Bound Used By: nginx-deployment-fcd9fd98b-x6tbs Events: Type Reason Age From Message -------------____ Normal WaitForFirstConsumer 84s (x26 over 7m34s) persistentvolumecontroller waiting for first consumer to be created before binding Normal **Provisioning** 54s everest-csi-provisioner_everest-csicontroller-7965dc48c4-5k799_2a6b513e-f01f-4e77-af21-6d7f8d4dbc98 External provisioner is provisioning volume for claim "default/topology" Normal ProvisioningSucceeded 52s everest-csi-provisioner_everest-csicontroller-7965dc48c4-5k799 2a6b513e-f01f-4e77-af21-6d7f8d4dbc98 Successfully provisioned volume pvc-9a89ea12-4708-4c71-8ec5-97981da032c9

Using csi-disk-topology in Cross-AZ Node Deployment

The following uses csi-disk-topology to create a StatefulSet with the same configurations used in the preceding example.

volumeClaimTemplates: - metadata: name: data annotations: everest.io/disk-volume-type: SAS spec: accessModes: - ReadWriteOnce resources: requests: storage: 1Gi storageClassName: csi-disk-topology

After the creation, check the PVC and pod status. As shown in the following output, the PVC and pod can be created successfully. The nginx-3 pod is created on the node in AZ 3.

# kubectl get pvc -owide			
NAME STATUS VOLUME CAPACITY	ACCESS N	MODES	
STORAGECLASS AGE VOLUMEMODE			
data-nginx-0 Bound pvc-43802cec-cf78-4876-bcca-e041618f2470	1Gi	RWO	csi-disk-
topology 55s Filesystem			
data-nginx-1 Bound pvc-fc942a73-45d3-476b-95d4-1eb94bf19f1f	1Gi	RWO	csi-disk-
topology 39s Filesystem			
data-nginx-2 Bound pvc-d219f4b7-e7cb-4832-a3ae-01ad689e364	e 1Gi	RWO	csi-disk-
topology 22s Filesystem			
data-nginx-3 Bound pvc-b54a61e1-1c0f-42b1-9951-410ebd326a4	d 1Gi	RWO	csi-disk-
topology 9s Filesystem			
# kubectl get pod -owide			
NAME READY STATUS RESTARTS AGE IP NODE	NOM	INATED NODE	E READINESS
GATES			
nginx-0 1/1 Running 0 65s 172.16.1.8 192.168.0.240	<none></none>	<none></none>	
nginx-1 1/1 Running 0 49s 172.16.0.13 192.168.0.121	<none></none>	<none></none>	
nginx-2 1/1 Running 0 32s 172.16.0.137 192.168.0.211	<none></none>	<none></none>	•
nginx-3 1/1 Running 0 19s 172.16.1.9 192.168.0.240	<none></none>	<none></none>	

11.7 Automatically Collecting JVM Dump Files That Exit Unexpectedly Using a General Purpose File System (SFS 3.0 Capacity-Oriented)

If you are using Java to develop services, you may encounter an out of memory (OOM) problem if the JVM heap space is insufficient. To address this issue, you can use general purpose file systems (SFS 3.0 Capacity-Oriented) to store logs and mount the file systems to the relevant directories in containers. In the event of a JVM OOM, general purpose file systems (SFS 3.0 Capacity-Oriented) can record logs.

Prerequisites

- A CCE standard cluster has been created. For details, see **Buying a CCE Standard/Turbo Cluster**.
- Before using general purpose file systems (SFS 3.0 Capacity-Oriented) for CCE container storage, you need to configure a VPC endpoint to communicate

with the general purpose file systems (SFS 3.0 Capacity-Oriented). For details, see **Configure a VPC Endpoint**.

Procedure

Step 1 Create a PVC based on a general purpose file system (SFS 3.0 Capacity-Oriented).

cat << EOF | kubectl apply -f apiVersion: v1 kind: PersistentVolumeClaim metadata: name: jvm-sfs-pvc namespace: default annotations: {} spec: accessModes: - ReadWriteMany resources: requests: storage: 10Gi storageClassName: csi-sfs EOF

Step 2 Create a Deployment to simulate Java OOM and dump the generated dump files to the PV associated with the general purpose file system (SFS 3.0 Capacity-Oriented).

```
cat << EOF | kubectl apply -f -
apiVersion: apps/v1
kind: Deployment
metadata:
 name: java-application
 namespace: default
spec:
 selector:
  matchLabels:
   app: java-application
 template:
  metadata:
   labels:
    app: java-application
  spec:
   containers:

    name: java-application

    image: swr.cn-east-3.myhuaweicloud.com/container/java-oom-demo:v1 #The image in this document
is only an example.
    imagePullPolicy: Always
    env:

    name: POD_NAME # Use metadata.name as the value of the POD_NAME environment variable.

      valueFrom:
       fieldRef:
         apiVersion: v1
         fieldPath: metadata.name
     - name: POD_NAMESPACE #Use metadata.namespace as the value of the POD_NAMESPACE
environment variable.
      valueFrom:
       fieldRef:
         apiVersion: v1
         fieldPath: metadata.namespace
    aras:
     - java
                            #Run the Java command.
                               #Configure the minimum heap size of the heap memory.
     - -Xms80m
                               #Configure the maximum heap size of the heap memory.
     - -Xmx80m
     - -XX:HeapDumpPath=/mnt/oom/logs #Heap memory dump path when OOM occurs
     - -XX:+HeapDumpOnOutOfMemoryError #Capture the heap OOM error.
     - Mycode
                              # Execute the application in the example image.
    volumeMounts:
     - name: java-oom-pv
     mountPath: "/mnt/oom/logs" # The container that is created using the example image
```
<pre>ises /mnt/oom/logs as the log mount directory. subPathExpr: default iava_application # Create a subdirectory in the \$(POD_NAMESPACE)</pre>						
(POD_NAME) format and generate OOM dump files in the subdirectory.						
imagePullSecrets:						
- name: default-secret						
volumes:						
- name: java-oom-pv						
persistentVolumeClaim:						
claimName: jvm-sfs-pvc #PVC using the SFS file system, named jvm-sfs-pvc						
EOF						

NOTE

In the example provided, the parameter settings are tailored to align with the image used, demonstrating the process of log collection through SFS volume mounting. It is necessary to customize these parameter settings based on the specific requirements of the actual service image in use.

Step 3 Wait until the container automatically restarts due to OOM.

kubectl -n default get pod

Information similar to the following is displayed:

NAME READY STATUS RESTARTS AGE java-application-84dc6f897f-hc9q7 1/1 Running 1 (31s ago) 97s

- **Step 4** Obtain the files generated by the Java program due to OOM.
 - 1. Log in to the CCE console and click the cluster name to access the cluster console. In the navigation pane, choose **Storage**, click the **PVCs** tab, locate the row containing **jvm-sfs-pvc**, and click the name of the associated PV.
 - 2. After the system automatically switches to the row containing the corresponding PV, click the name of the associated storage volume.
 - 3. After the system automatically switches to the SFS console, copy the mounting command.

< pvc-1633b380-aa7	'6-46ac			
Basic Information	Basic Info			
Permissions Management	Name	pvc-1633b380-aa76-46ac-8200-81a4e54de312	Region	$-10^{-1} (100 {\rm gm}^{-1})$
Limits Management	Used Capacity	0 Byte	AZ	AZ1
Taos	File System Classes	Standard	Protocol Type	NFS
	Created	Jul 22, 2024 12:18:55	Enterprise Project	default
	Mount Point Info			
	Mount in Linux	# mount -t nfs -o vers=3,retrans=1,timeo=600,noresvport,nolock,proto=tcp sfs.l:	z01.cn-north-7.myhuaweicloud.com:/pvc-1633b380-aa76-46ac-8200-81a4e5	i4de312 /localfolder 🗇
		Note: localfolder indicates the local directory created by you.		
• Note: Before running the preceding command on an ECS, make sure that the rfs-utils has been installed. Hore information about mounting/3				×

4. Log in to a node, create a mount point, and run the mount command to mount the SFS volume to the node. mkdir /test-ivm

mount -t nfs -o vers=3,timeo=600,noresvport,nolock,proto=tcp ***.com:/pvc-4ea9137e-4101-4610a4d2-9f8bb37043a1 /test-jvm

5. Check the files in the mounted file system. The dump file **java_pid1.hprof** is present in the directory. To identify the line of code that triggers an OOM error, download **java_pid1.hprof** to the local host and use Eclipse Memory Analyzer Tool (MAT) to further analyze JVM stack information.

----End

11.8 Deploying Storage Volumes in Multiple AZs

Application Scenarios

- Deploying services in specific AZs within a cluster that has nodes running in multiple AZs
- Preventing faults caused by a lack of resources in a single AZ with multi-AZ deployment

Deploying storage volumes in multiple AZs reduces application interruptions during rollout and ensures the stability of key systems and applications in case of any faults.

Prerequisites

- You have created a cluster with the CCE Container Storage (Everest) add-on installed and the cluster version is 1.21 or later. If no cluster is available, create one by referring to **Buying a CCE Standard/Turbo Cluster**.
- Nodes in the cluster must be in at least three different AZs. If the current nodes are not in three different AZs, you need to create new nodes or node pools in AZs that have no such resources deployed.

Procedure

- Step 1 Use kubectl to access the cluster. For details, see Connecting to a Cluster Using kubectl.
- Step 2 Create a StorageClass YAML file.

vi *storageclass.yaml*

Enter the following content in the **storageclass.yaml** file: (The following shows only a template for StorageClass configuration. You can modify it as required.)

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
name: test-disk-topology-alltype
provisioner: everest-csi-provisioner
parameters:
csi.storage.k8s.io/csi-driver-name: disk.csi.everest.io
csi.storage.k8s.io/fstype: ext4
everest.io/disk-volume-type: SAS #A high I/O EVS disk
everest.io/passthrough: "true"
reclaimPolicy: Delete
allowVolumeExpansion: true
volumeBindingMode: WaitForFirstConsumer
```

Table 11-18 Sto	orageClass para	meters
-----------------	-----------------	--------

Parameter	Description
provisioner	Specifies the storage resource provider, which is the Everest add-on for CCE. Set this parameter to everest-csi- provisioner .

Parameter	Description		
parameters	 Specifies the storage parameters, which vary with storage types. NOTICE everest.io/disk-volume-type indicates the cloud disk type, which can be any of the following: SAS: high I/O SSD: ultra-high I/O GPSSD: general purpose SSD ESSD: extreme SSD GPSSD2: general purpose SSD v2, which is supported when the Everest version is 2.4.4 or later and the everest.io/disk-iops and everest.io/disk-iops 		
	 ESSD2: extreme SSD v2, which is supported when the Everest version is 2.4.4 or later and the everest.io/disk-iops annotation is configured. 		
reclaimPolicy	Specifies the value of persistentVolumeReclaimPolicy for creating a PV. The value can be Delete or Retain . If reclaimPolicy is not specified when a StorageClass object is created, the value defaults to Delete .		
	• Delete : indicates that a dynamically provisioned PV will be automatically deleted when the PVC is deleted.		
	• Retain : indicates that a dynamically provisioned PV will be retained when the PVC is deleted.		
allowVolumeEx pansion	Specifies whether the PV of this StorageClass supports dynamic capacity expansion. The default value is false . Dynamic capacity expansion is implemented by the underlying storage add-on. This is only a switch.		
volumeBinding Mode	Specifies when a PV is dynamically provisioned. The value can be Immediate or WaitForFirstConsumer .		
	• Immediate: The PV is dynamically provisioned when a PVC is created.		
	• WaitForFirstConsumer: The PV is dynamically provisioned when the PVC is used by the workload.		

Step 3 Create the StorageClass.

kubectl create -f storageclass.yaml

Step 4 Create a StatefulSet YAML file.

vi *statefulset.yaml*

Enter the following content in the **statefulset.yaml** file: (The following shows only a template for the standard StatefulSet configuration. You can customize it as required.)

apiVersion: apps/v1 kind: StatefulSet metadata: name: nginx spec: replicas: 3 serviceName: "nginx" selector: matchLabels: app: nginx template: metadata: labels: app: nginx spec: topologySpreadConstraints: - labelSelector: # Used to search for matched pods and count the pods that match the label selector to determine the number of pods in the corresponding topology domain. matchLabels: app: nginx maxSkew: 1 # Maximum difference between the numbers of matched pods in any two topology domains in a given topology type. topologyKey: topology.kubernetes.io/zone # Key of a node label whenUnsatisfiable: DoNotSchedule # How the scheduler processes pods when the pods do not meet the spread constraints containers: - image: nginx:latest name: nginx env: - name: NGINX_ROOT_PASSWORD value: "nginx" volumeMounts: - name: disk-csi mountPath: /var/lib/nginx imagePullSecrets: - name: default-secret tolerations: - key: "app" operator: "Exists" effect: "NoSchedule" volumeClaimTemplates: # EVS disks are automatically created based on the specified number of replicas for quick expansion. - metadata: name: disk-csi spec: accessModes: ["ReadWriteOnce"] # EVS disks can only be mounted to and accessed by a single node in read/write mode, that is, ReadWriteOnce. storageClassName: test-disk-topology-alltype resources: requests: storage: 40Gi

Tal	ble	<u>1</u>	1-19	StatefulSe	t parameters
-----	-----	----------	------	------------	--------------

Parameter	Description
topologySpreadCon straints	Specifies the topology spread constraints, which are used to control how pods are spread across a cluster among topology domains, such as regions, AZs, nodes, and other custom topology domains. For details, see Pod Topology Spread Constraints .
topologySpreadCon straints.labelSelecto r	Used to search for matched pods. The number of pods that match this label selector is counted to determine the number of pods in the corresponding topology domain.

Parameter	Description	
topologySpreadCon straints.maxSkew	Specifies the maximum difference between the numbers of matched pods in any two topology domains in a given topology type. The value must be greater than 0 and is used to indicate how much uneven distribution of pods is allowed.	
topologySpreadCon straints.topologyKey	Specifies the key of a node label. If two nodes use this key and have the same label value, the scheduler treats them as being in the same topology domain and tries to schedule an equal number of pods to each domain.	
topologySpreadCon straints.whenUnsati sfiable	Specifies how the scheduler processes pods when the pods do not meet the spread constraints. The value can be:	
	• DoNotSchedule (Default): If a pod does not meet the spread constraints, it will not be scheduled.	
	• ScheduleAnyway : If a pod does not meet the spread constraints, it will be scheduled to the node with the minimum skew preferentially.	
volumeClaimTempla tes	Specifies that EVS disks are automatically created based on the specified number of replicas for quick expansion.	

Step 5 Create the StatefulSet.

kubectl create -f statefulset.yaml

----End

Verification

The following shows how to verify that the dynamically created PVs are in different AZs along with the pods.

Step 1 View the new PVs.

kubectl get pv

The command output is as follows: (The first three PVs are dynamically created with the pods.)

NAME	CAPACITY	ACCESS	MODES RECLA	IM POLICY	STATUS CLAIM	
STORAGECLA	ASS	VOLU	MEATTRIBUTESC	LASS REA	ASON AGE	
pvc-699eda7	'5 40Gi	RWO	Delete	Bound	default/disk-csi-nginx-	0 test-disk-topology-
alltype <un< td=""><td>set></td><td></td><td>132m</td><td></td><td></td><td></td></un<>	set>		132m			
pvc-6c68f5a	7 40Gi	RWO	Delete	Bound	default/disk-csi-nginx-	1 test-disk-topology-
alltype <un< td=""><td>set></td><td></td><td>131m</td><td></td><td></td><td></td></un<>	set>		131m			
pvc-8f74ce3a	a 40Gi	RWO	Delete	Bound	default/disk-csi-nginx-	2 test-disk-topology-
alltype <un< td=""><td>set></td><td></td><td>131m</td><td></td><td></td><td></td></un<>	set>		131m			
pvc-f738f8aa	10Gi	RWO	Delete	Bound	default/pvc c	si-disk
<unset></unset>		6d4h				

Step 2 Check the AZs where the PVs are located based on the PV names. kubectl describe pv *pvc-699eda75 pvc-6c68f5a7 pvc-8f74ce3a* | grep zone

The command output is as follows: (The three PVs are in different AZs to enable multi-AZ deployment of storage volumes.)

Labels:	failure-domain.beta.kubernetes.io/zone= cn-east-3d
Term 0:	failure-domain.beta.kubernetes.io/zone in [cn-east-3d]
Labels:	failure-domain.beta.kubernetes.io/zone= cn-east-3b
Term 0:	failure-domain.beta.kubernetes.io/zone in [cn-east-3b]
Labels:	failure-domain.beta.kubernetes.io/zone= cn-east-3c
Term 0:	failure-domain.beta.kubernetes.io/zone in [cn-east-3c]

----End

12 Container

12.1 Recommended Configurations for Workloads

When deploying a workload in a CCE cluster, you need to configure the workload based on the actual service scenarios and environments to ensure that the workload can run stably and reliably. This section provides some recommended configurations and suggestions for workload deployment.

Specifying Pod Resources (Requests and Limits)

Requests and limits need to be configured based on the actual service scenarios. The requests are used for the scheduler to check available resources and record the allocated resources on each node. The allocated resources on a node are the sum of container requests defined in all pods on the node. You can calculate the available resources on a node using the following formula: Available resources on a node using the following formula: Available resources on a node. If there are not enough available resources on a node to accommodate a pod's requests, the pod will not be scheduled on that node.

If the requests are not configured, the scheduler cannot determine the resource usage on a node and cannot schedule pods to suitable nodes. This can lead to a situation where a node becomes overloaded with a large number of pods, potentially causing issues with the node and impacting the actual services. It is recommended that you configure requests for all containers so that the scheduler can accurately monitor the resource usage on nodes and make appropriate scheduling decisions.

The following shows an example of how to configure the request and limit for an Nginx pod. The request specifies that the pod requires 0.5 CPU cores and 128 MiB of memory. During running, the pod can use resources beyond the request, but it cannot exceed the resource limit of 1 CPU core and 256 MiB of memory.

apiVersion: v1 kind: Pod metadata: name: nginx-test spec: containers: - name: container-1

```
image: nginx
resources: # Resource declaration
resources:
limits:
cpu: 1000m
memory: 256Mi
requests:
cpu: 500m
memory: 128Mi
imagePullSecrets:
- name: default-secret
```

Configuring a Graceful Exit Period

The graceful exit period (**terminationGracePeriodSeconds**) is the time between a failed pod triggering the termination process and the pod being forcefully stopped. By default, if this parameter is not specified, the grace period is set to 30 seconds, with a minimum value of 1 second. During this grace period, the pod can be gracefully shut down, allowing it to perform operations such as saving its status, completing ongoing tasks, and closing network connections. It is crucial to configure **terminationGracePeriodSeconds** properly to ensure a smooth, orderly termination of an application.

If you want a pod to wait for 60 seconds before termination, allowing the pod to be properly cleared, you can include the following parameters in the pod definition:

kind: Deployment apiVersion: apps/v1 metadata: name: nginx spec: replicas: 1 selector. matchLabels: app: nginx version: v1 template: metadata: labels: app: nginx version: v1 spec: containers: - name: container-1 image: nginx imagePullSecrets: - name: default-secret terminationGracePeriodSeconds: 60

Configuring Tolerations

Tolerations allow pods to be scheduled on nodes even if there are taints present. For example, if an application heavily relies on the local state of a node, you may want it to remain on that node for an extended period during a network partition, waiting for the network to recover and avoiding eviction.

Sometimes, the Kubernetes node controller automatically adds taints to nodes. It is recommended that you add tolerations for the built-in taints **node.kubernetes.io/not-ready** (indicating the node is not ready) and **node.kubernetes.io/unreachable** (indicating the node controller cannot access the node). In the following example, a node has added the preceding tolerations, and the pod will continue running on the node for 300 seconds before being evicted.

apiVersion: v1 kind: Pod metadata: name: nginx-test spec: containers: - name: container-1 image: nginx imagePullSecrets: - name: default-secret tolerations: - key: node.kubernetes.io/not-ready operator: Exists effect: NoExecute tolerationSeconds: 300 - key: node.kubernetes.io/unreachable operator: Exists effect: NoExecute tolerationSeconds: 300

Configuring a Rolling Update

In Kubernetes, the **strategy** field in a workload determines how resources like Deployments, StatefulSets, and DaemonSets are updated. To maintain service continuity during a workload upgrade, you can use rolling updates to control the number of available pods, minimizing downtime. For example, in a Deployment with multiple pods, you can specify the maximum number of old pods that can be unavailable and the maximum number of new pods that can be started and running until the update is complete. Rolling updates ensure service stability and availability while smoothly transitioning applications to new versions.

In the following example, a rolling update policy is configured, where both **maxUnavailable** and **maxSurge** are set to **25%**. This means that up to 25% of old pods can be unavailable and up to 25% of new pods can be started during the update.

kind: Deployment apiVersion: apps/v1 metadata: name: nginx spec: replicas: 10 selector: matchl abels: app: nginx version: v1 template: metadata: labels: app: nginx version: v1 spec: containers: - name: container-1 image: nginx imagePullSecrets: - name: default-secret strategy: type: RollingUpdate rollingUpdate: maxUnavailable: 25% maxSurge: 25%

Configuring a Restart Policy

The **restartPolicy** parameter is used to define the behavior after a pod terminates. You can customize the policy based on your specific services to enable automatic restarts when a pod exits.

The following is an example of configuring an Nginx pod to always restart automatically:

apiVersion: v1 kind: Pod metadata: name: nginx-test spec: containers: - name: nginx image: nginx restartPolicy: Always imagePullSecrets: - name: default-secret

The options of restartPolicy include:

- Always: The pod always restarts automatically after any termination.
- **OnFailure**: The pod automatically restarts if it exits with an error. (The process exit status is not 0).
- Never: The pod never restarts.

Configuring a Liveness Probe and a Readiness Probe

A liveness probe checks whether a pod is normal. In Kubernetes, if a pod is in the **Running** state, it does not mean that the pod can provide services properly. The pod may fail to provide services due to problems in processes such as deadlock. You can configure a liveness probe to avoid similar problems and restart the pod in a timely manner to restore your service.

A readiness probe detects whether a pod is ready to receive Service requests. If the pod is faulty, the readiness probe avoids forwarding new traffic to the pod.

apiVersion: v1 kind: Pod metadata: name: tomcat spec: containers: - name: tomcat image: tomcat livenessProbe: httpGet: path: /index.jsp port: 8080 initialDelaySeconds: 3 periodSeconds: 3 readinessProbe: httpGet: path: /index.jsp port: 8080 imagePullSecrets: - name: default-secret

12.2 Properly Allocating Container Computing Resources

If a node has sufficient memory resources, a container on this node can use more memory resources than requested, but no more than limited. If the memory allocated to a container exceeds the upper limit, the container is stopped first. If the container continuously uses memory resources more than limited, the container is terminated. If a stopped container is allowed to be restarted, kubelet will restart it, but other types of run errors will occur.

Scenario 1

The node's memory has reached the memory limit reserved for the node. As a result, OOM killer is triggered.

Solution

You can either scale up the node or migrate the pods on the node to other nodes.

Scenario 2

The upper limit of resources configured for the pod is too small. When the actual usage exceeds the limit, OOM killer is triggered.

Solution

Set a higher upper limit for the workload.

Example

A pod will be created and allocated memory that exceeds the limit. As shown in the following configuration file of the pod, the pod requests 50 MiB memory and the memory limit is set to 100 MiB.

Example YAML file (memory-request-limit-2.yaml):

```
apiVersion: v1
kind: Pod
metadata:
name: memory-demo-2
spec:
 containers:
 - name: memory-demo-2-ctr
  image: vish/stress
  resources:
   requests:
    memory: 50Mi
   limits:
    memory: "100Mi"
  args:
  - -mem-total
  - 250Mi
  - -mem-alloc-size
  - 10Mi
  - -mem-alloc-sleep
  - 1s
```

The **args** parameters indicate that the container attempts to request 250 MiB memory, which exceeds the pod's upper limit (100 MiB).

Creating a pod:

kubectl create -f memory-request-limit-2.yaml --namespace=mem-example

Viewing the details about the pod:

kubectl get pod memory-demo-2 --namespace=mem-example

In this stage, the container may be running or be killed. If the container is not killed, repeat the previous command until the container is killed.

NAME READY STATUS RESTARTS AGE memory-demo-2 0/1 OOMKilled 1 24s

Viewing detailed information about the container:

kubectl get pod memory-demo-2 --output=yaml --namespace=mem-example

This output indicates that the container is killed because the memory limit is exceeded.

```
lastState:
terminated:
```

```
containerID: docker://7aae52677a4542917c23b10fb56fcb2434c2e8427bc956065183c1879cc0dbd2
exitCode: 137
finishedAt: 2020-02-20T17:35:12Z
reason: OOMKilled
startedAt: null
```

In this example, the container supports automatic restart, so kubelet will restart it upon termination. You can run the following command multiple times to see how the container is stopped and restarted:

kubectl get pod memory-demo-2 --namespace=mem-example

The preceding command output indicates how the container is killed and started back and forth:

\$ kubectl get pod memory-demo-2 --namespace=mem-exampleNAMEREADYSTATUSRESTARTSAGEmemory-demo-20/1OOMKilled137s\$ kubectl get pod memory-demo-2 --namespace=mem-exampleNAMEREADYSTATUSRESTARTSAGEmemory-demo-21/1Running240s

Viewing the historical information of the pod:

kubectl describe pod memory-demo-2 --namespace=mem-example

The following command output indicates that the pod is repeatedly killed and started.

... Normal Created Created container with id 66a3a20aa7980e61be4922780bf9d24d1a1d8b7395c09861225b0eba1b1f8511 ... Warning BackOff Back-off restarting failed container

12.3 Upgrading Pods Without Interrupting Services

Application Scenarios

In a Kubernetes cluster, applications can be accessed externally through Deployments and LoadBalancer Services. When an application is updated or

upgraded, new pods are created in the Deployment. These new pods will gradually replace the old ones. During this process, services may be interrupted.

Solution

To prevent an application upgrade from interrupting services, configure Deployments and Services as follows:

- In a Deployment, upgrade pods in the Rolling upgrade mode. In this mode, pods are updated one by one, not all at once. In this way, you can control the update speed and the number of concurrent pods to ensure that services are not interrupted during the upgrade. For example, you can configure the maxSurge and maxUnavailable parameters to control the number of new pods created and the number of old pods deleted concurrently. Ensure that there is always a workload that can provide services during the upgrade.
- There are two types of service affinity in a LoadBalancer:
 - **Cluster-level** service affinity (**externalTrafficPolicy: Cluster**). In this mode, if there is no pod deployed on a node, the request is forwarded to pods on another node. During the cross-node forwarding, the source IP address may be lost.
 - Node-level service affinity (externalTrafficPolicy: Local). In this mode, requests are directly forwarded to the node where the pod resides. Cross-node forwarding is not involved. Therefore, the source IP address can be preserved. However, if the node where the pod resides changes during the rolling upgrade, the ELB backend server will change accordingly, which may cause service interruption. In this case, you can upgrade pods in place. This ensures that there is at least one pod running properly on the ELB backend node.

The following table lists the solution for ensuring service continuity during a pod upgrade.

Scenario	Service	Deployment	
The source IP address does not need to be preserved.	Select the Cluster-level service affinity.	Select Rolling upgrade for Upgrade Mode , configure a graceful termination, and enable Liveness probe and Ready probe .	
The source IP address needs to be preserved.	Select the Node-level service affinity.	Select Rolling upgrade for Upgrade Mode, configure a graceful termination, enable Liveness probe and Ready probe, and add Node Affinity policies. (Ensure that there is at least one pod running on each node during the update.)	

Procedure

In this example, there are 200 replicas in the workload, and the workload is exposed through the LoadBalance Service. When upgrading workloads associated with LoadBalancer Services or ingresses, cross-service calling is necessary, so it is important to carefully configure the rolling upgrade parameters.

- **Step 1** Log in to the CCE console and click the cluster name to access the cluster console. In the navigation pane, choose **Workloads**.
- **Step 2** In the workload list, click **Upgrade** in the **Operation** column of the workload to be upgraded. The **Upgrade Workload** page is displayed.
 - Enable the liveness probe and ready probe. In the Container Settings area, click Health Check and enable Liveness probe and Ready probe. In this example, TCP is selected for Check Method. Configure the parameters based on your requirements. Parameters like Period (s), Delay (s), and Timeout (s) must be properly configured. Some applications take a long time to start. A small value of these parameters will lead to repeated restart.

In this example, the ready probe delay is set to **20** to control the interval for rolling workloads in batches.

Container Settings			
Container Information	Container - 1		+ Add Container
	Basic Info Lifecycle Health Check Environment Variables Data Storage	How do I configure health check for containent?() Leveness probe (backs whether the containent responds properly and whether a restart is required) Enable Check Method: MTIP: Or O: Command Port Mandatory Entre a conted port number The	atrod (s) 10 atrod (s) 0 1 1
	Context Logging	Sa	xcess Threshold 1
		 Ready probe (checks whether the container is ready to respond to requests) 	
		Enable 💽 Per	triod (s) 10
		Check Method O HTTP TCP O Command	liay (s) 20
		Port 80 Tim	meout (s) 1
		Sa	access Threshold 1
		Fai	ailure Threshold 3

Figure 12-1 Enabling the liveness probe and ready probe

2. Configure a rolling upgrade. In the **Advanced Settings** area, click **Upgrade** and select **Rolling upgrade** for **Upgrade Mode**. This ensures that the instances of the old versions are gradually replaced with the ones of the new versions.

In this example, **maxUnavailable** is set to **2%**, and **maxSurge** is set to **2%** to control the workload rolling step. This, works with the ready probe delay, enables eight workloads to be upgraded every 20 seconds.

Figure 12-2 Configuring a rolling upgrade

Advanced Settings				
Upgrade	💡 You can configu	re replace or rolling upgrade policies to upgrade your workload.		
Toleration	Upgrade Mode	Rolling upgrade Reptace upgrade Old pods will be analysisty replaced with new pods. During the upgrade, services will be eventy distributed to both old and new pods to ensur	e service continuity.	
Labels and Annotations	Max. Unavailable	2] [% v]	Max. Surge	2
DNS	Pods	Maximum number of unavailable pods allowed in a rolling upgrade. If the number is equal to the total number of pods, services may be interrupted. (Winimum number of allve pods = Total pods – Maximum number of unavailable pods)		Maximum number of pods that can be created over the desired number of pods in each rolling upgrade
Network Configuration	Min. Ready Seconds	Minimum number of seconds for which a newly created pod should be ready without any of its containers crasting, for it is be considered analytic.	Revision History Limit	10
	Max. Upgrade Duration (s)	600	Scale-In Time Window (s)	60 Time window (0-9999s) for pre-stop commands to finish execution before a worldvad is forcibly deletec value is 30s.

- 3. Configure a graceful termination.
 - a. In the **Container Settings** area, click **Lifecycle** and configure pre-stop processing. Configure this parameter to the time required for the Service to process all remaining requests, most of which are persistent connection requests. You can, for example, set the workload to hibernate for 30s after receiving a deletion request so that the workload can have sufficient time to process the remaining requests to ensure proper service running.
 - b. In the Advanced Settings area, click Upgrade. Configure Scale-In Time Window (terminationGracePeriodSeconds) to specify the waiting time for command execution before the container is stopped. The scale-in time window must be greater than the pre-stop processing time. Add 30s to the command execution time before the container is stopped. If, for example, the pre-stop processing time is 30s, the scale-in time window should be 60s.

Figure 12-3 Entering the pre-stop command

Container Settings								
Container Information	Container - 1	ontainer - 1						
	Eask Info Lifecycle Heath Check Environment Variables Data Storage Security Context Logging	How do I set con Startup Command The pre-stop hook is completion and exits (C Processing Method Command	taker lifegole hoats? (3 Ped-Start Pre-Step caldo immediatily before a container is stopper completion)	d (Terminuted), not when the container runs to Detete Detete Detete	Examples	Bash Boary HTTP Request Anthan C < C while from do actor helic, since 10, done C		
	Logging		sleep 30; +	Delete				

 Add node affinity policies. Add this kind of policy when Node-level is selected for a Service's Service Affinity. In the Advanced Settings area, click Scheduling and add Node Affinity policies. When adding a scheduling policy, specify the nodes that the workload requires affinity.

Figure 12-4 Adding node affinity policies

Add Scheduling Policy				
O Multiple policies can be added for a rule and trigger sch	eduling if all met.			
Label Key	Operator (?)	Label Value	Operation	
Add Policy (Specify Node Specify AZ)				

Step 3 After the configuration is complete, click **Upgrade Workload**.

On the **Pods** tab, after a newly created pod is displayed, stop the old one. This ensures that there is always a pod running in the workload.

----End

12.4 Modifying Kernel Parameters Using a Privileged Container

Prerequisites

To access a Kubernetes cluster from a client, you can use the Kubernetes command line tool kubectl. For details, see **Connecting to a Cluster Using kubectl**.

Procedure

- **Step 1** Create a DaemonSet on the backend, select the Nginx image, enable the privileged container, configure the lifecycle, and specify **hostNetwork: true**.
 - 1. Create a DaemonSet file. vi daemonset.yaml

An example YAML file is provided as follows:

NOTICE

The **spec.spec.containers.lifecycle** field indicates the command that will be run after the container is started.

kind: DaemonSet
motodoto:
name: daomonsot tost
labole:
name: daemonset test
spec:
spec.
matchl abels
name: daemonset_test
template:
metadata:
labels:
name: daemonset-test
spec.
hostNetwork: true
containers
- name: daemonset-test
image: nginx:alpine-perl
command:
- "/bin/sh"
args:
- "-c"
- while :; do time=\$(date);done
imagePullPolicy: IfNotPresent
lifecycle:
postStart:
exec:
command:

- sysctl
 "-w"
 net.ipv4.tcp_tw_reuse=1
 securityContext:
 privileged: true
 imagePullSecrets:
 name: default-secret
- 2. Create the DaemonSet. kubectl create –f daemonSet.yaml
- **Step 2** Check whether the DaemonSet has been created.

kubectl get daemonset {daemonset_name}

In this example, run the following command:

kubectl get daemonset daemonset-test

Information similar to the following is displayed:

 NAME
 DESIRED
 CURRENT
 READY
 UP-T0-DATE
 AVAILABLE
 NODE SELECTOR
 AGE

 daemonset-test
 2
 2
 2
 2
 <node>
 2h

Step 3 Obtain the IDs of the DaemonSet pods on the nodes.

kubectl get pod | grep {daemonset_name}

In this example, run the following command:

kubectl get pod | grep *daemonset-test*

Information similar to the following is displayed:

daemonset-test-mqdpv	1/1	Running	0	2h
daemonset-test-n56vm	1/1	Running	0	2h

Step 4 Access the container.

kubectl exec -it *{pod_name}* -- /bin/sh

In this example, run the following command:

kubectl exec -it daemonset-test-mqdpv -- /bin/sh

Step 5 Check whether the configured command is executed after the container is started. sysctl -a |grep net.ipv4.tcp_tw_reuse

If the following information is displayed, the system parameters are modified successfully:

net.ipv4.tcp_tw_reuse=1

----End

12.5 Using Init Containers to Initialize an Application

Concepts

An init container is a type of container that starts and exits before the application containers start. If there are multiple init containers, they will be started in the defined sequence. The data generated in the init containers can be used by the application containers because storage volumes in a pod are shared. Init containers can be used in multiple Kubernetes resources, such as Deployments, DaemonSets, and jobs. They perform initialization before application containers are started.

Application Scenarios

Before deploying a service, you can use an init container to make preparations before the service pod is deployed. After the preparations are complete, the init container runs to completion and exits, and the container to be deployed will be started.

- Scenario 1: Wait for other modules to be ready. For example, an application contains two containerized services: web server and database. The web server service needs to access the database service. However, when the application is started, the database service may have not been started. Therefore, web server may fail to access database. To solve this problem, you can use an init container in the pod where web server is running to check whether database is ready. The init container runs to completion only when database is accessible. Then, web server is started and initiates a formal access request to database.
- Scenario 2: Initialize the configuration. For example, the init container can check all existing member nodes in the cluster and prepare the cluster configuration information for the application container. After the application container is started, it can be added to the cluster using the configuration information.
- **Other scenarios**: For example, a pod is registered with a central database and application dependencies are downloaded.

For details, see **Init Containers**.

Procedure

Step 1 Edit the YAML file of the init container workload.

vi deployment.yaml

An example YAML file is provided as follows:

apiVersion: apps/v1 kind: Deployment metadata: name: mysql spec: replicas: 1 selector: matchLabels: name: mysql template: metadata: labels: name: mysal spec: initContainers: - name: getresource image: busybox command: ['sleep 20'] containers: name: mysal image: percona:5.7.22 imagePullPolicy: Always

```
ports:

- containerPort: 3306

resources:

limits:

memory: "500Mi"

cpu: "500Mi"

requests:

memory: "500Mi"

cpu: "250m"

env:

- name: MYSQL_ROOT_PASSWORD

value: "mysql"
```

Step 2 Create an init container workload. kubectl create -f deployment.yaml

Information similar to the following is displayed:

deployment.apps/mysql created

Step 3 Query the created Docker container on the node where the workload is running. docker ps -a|grep mysql

The init container will exit after it runs to completion. The query result **Exited (0)** shows the exit status of the init container.



----End

12.6 Setting Time Zone Synchronization

Case Scenarios

- Scenario 1: Setting Time Zone Synchronization Between Pods and Nodes
- Scenario 2: Setting Time Zone Synchronization Between Java Container Logs, Pods, and Nodes
- Scenario 3: Setting Time Zone Synchronization Between Workloads and Nodes

Scenario 1: Setting Time Zone Synchronization Between Pods and Nodes

- **Step 1** Log in to the CCE console.
- Step 2 In the Basic Info area of the Create Workload page, enable Time Zone Synchronization so that the same time zone will be used for both the pod and the node. After completing other settings, create the workload.

Basic Info					
Workload Type	Deployment	tatefulSet	DaemonSet	Job	Cron Job
	A Switching the workload typ	pe will require you to confi	gure workload parameter	s again.	
Workload Name	Enter a workload name.				
Namespace	default		✓ Q Create Name	espace	
Pods	─ 2 +				
Burst to CCI	Supports the rapid and elastic cre reduce the consumption caused b	eation of Pods to the cloud by cluster expansion. Lear	container instance CCI s n more	ervice in short-term high loa	ad scenarios to
	1 The CCE burst elastic eng enabled. Install	gine (connected to CCI) pl	ug-in is not installed in the	e cluster, and CCI elastic be	earing cannot be
Synchronization	Allows containers to use the sar	me time zone as the node	where they run. (This fun	ction is realized by the local	disks mounted to
Time Zone Synchronization	Allows containers to use the sar	ne time zone as the node	where they run. (This fun	ction is realized by the local	disks mounted to

Figure 12-5 Enabling the time zone synchronization

Step 3 Log in to the node and obtain the time zone. date -R

Information similar to the following is displayed:

the containers. Do not modify or delete the local disks.)

Sat, 12 Apr 2025 16:58:47 +0800

Step 4 Check whether the time zone of the pod is the same as that of the node.

Run the following kubectl command to obtain the pod name:

kubectl get pod

Run the kubectl command to access the pod. In the command, *<pod_name>* specifies the pod name obtained in the previous step.

kubectl exec -it <pod_name> -- /bin/bash

Obtain the time zone of the pod.

date -R

Information similar to the following is displayed:

Sat, 12 Apr 2025 16:59:23 +0800

----End

Scenario 2: Setting Time Zone Synchronization Between Java Container Logs, Pods, and Nodes

After time zone synchronization between a pod and the host node is enabled, the standard time inside the pod (obtained by running **date -R**) aligns with the node time. However, a Java application uses the default time zone, leading to discrepancies between the time recorded in logs and the pod time. The following uses a Java application that runs in a Tomcat container as an example to describe how to synchronize the time zone in the Java container logs with the node time zone.

- **Step 1** Log in to the CCE console.
- Step 2 In the Basic Info area of the Create Workload page, enable Time Zone Synchronization so that the same time zone will be used for both the pod and the node.

Figure 12-6	Enabling	the tim	e zone	synchronization
-------------	----------	---------	--------	-----------------

Basic Info					
Workload Type	Deployment	StatefulSet	DaemonSet	Job	Cron Job
	A Switching the worklo	ad type will require you to	o configure workload parame	ters again.	
Workload Name	Enter a workload name.				
Namespace	default		✓ Q Create Na	amespace	
Pods	- 2 +				
Burst to CCI	Supports the rapid and elas reduce the consumption car	tic creation of Pods to the used by cluster expansion	cloud container instance CC . Learn more	I service in short-term h	igh load scenarios to
	1 The CCE burst elast enabled. Install	ic engine (connected to C	CI) plug-in is not installed in	the cluster, and CCI ela	stic bearing cannot be
Time Zone					
Synchronization	Allows containers to use the	ie same time zone as the	node where they run. (This f	unction is realized by the	e local disks mounted to
	the containers. Do not mo	dify or delete the local disl	(S.)		

- **Step 3** In the **Container Settings** area, click **Environment Variables** and add the following environment variable:
 - Environment variable name: CATALINA_OPTS
 - Environment variable value: -Duser.timezone=GMT+08

Container Settings								
Container Information	Container - 1				+ Add Container			
	Basic Info Environment variables affect the way a numming container will behave. You can modify created variables as required affect diploying the workload. How Do I Add Environment Variables? Lifecycle							
	Health Check	Add Veriable Batch Edit Custom Variables						
	Environment	nt Type Variable Name Variable Value/Reference						
	Data	Custom ~	CATALINA_OPTS	-Duser.timezone=GMT+08	Delete			
	Storage							
	Context							
	Logging							

- **Step 4** After completing the configuration, create the workload.
- Step 5 In the workload list, locate the row containing the workload, click View Log in the Operation column, and check whether the time zone in the logs is the same as that of the pod and node. For details about how to check the time zone of a node and pod, see Scenario 1: Setting Time Zone Synchronization Between Pods and Nodes.

Х

View Log		
The currently displayed log content is the standard output log of the container, which does not have persistence and high-level oper-capabilities. If you need to use more complete log capabilities, you can use Logging ⁽²⁾ function. If the user has enabled AOM to col to AOM to view more load logs. View Logs in AOM ⁽²⁾	ation and maintenance llect standard output, you can	ו go
Download V		
Pod test-849848c998-8rbwh v Container container-1 v Rows 200 v Q Scheduled R	efresh 🔵 30s 🛛 🖉 Full so	creen
Apr 12, 2025 4:53:12 PM org.apache.catalina.core.AprLifecycleListener init INFO: Loaded APR based Apache Tomcat Native library 1.2.10 using APR version 1.5.1. Apr 12, 2025 4:53:13 PM org.apache.catalina.core.AprLifecycleListener init INFO: APR capabilities: IPv6 [true], sendfile [true], accept filters [false], random [true]. Apr 12, 2025 4:53:13 PM org.apache.catalina.core.AprLifecycleListener initializeSSL INFO: OpenSSL successfully initialized with version OpenSSL 1.0.2 j 26 Sep 2016 Apr 12, 2025 4:53:13 PM org.apache.coyote.httpl1.HttplAprProtocol init INFO: Initializing Coyote HTTP/1.1 on http=8080 Apr 12, 2025 4:53:13 PM org.apache.coyote.ajp.AjpAprProtocol init INFO: Initializing Coyote AJP/1.3 on ajp-8009 Apr 12, 2025 4:53:13 PM org.apache.catalina.startup.Catalina load INFO: Initialization processed in 2897 ms Apr 12, 2025 4:53:14 PM org.apache.catalina.core.StandardService start INFO: Starting service Catalina		

----End

Scenario 3: Setting Time Zone Synchronization Between Workloads and Nodes

- Method 1: Set the time zone to CST when creating a container image.
- Method 2: If you do not want to modify the container, when creating a workload on the CCE console, mount the **/etc/localtime** directory of the local host to the **/etc/localtime** directory of the container.

Example:

kind: Deployment
apiVersion: apps/v1
metadata:
name: test
namespace: default
spec:
replicas: 2
selector:
matchLabels:
app: test
template:
metadata:
labels:
app: test
spec:
volumes:
- name: vol-162979628557461404
hostPath:
path: /etc/localtime
type: "
containers:
- name: container-0
image: 'nginx:alpine'
volumeMounts:
- name: vol-162979628557461404
readOnly: true
mountPath: /etc/localtime
imagePullPolicy: IfNotPresent

imagePullSecrets: - name: default-secret

12.7 Configuration Suggestions on Container Network Bandwidth Limit

Application Scenarios

Containers on the same node share the host network bandwidth. Limiting the network bandwidth of containers can effectively prevent mutual interference between containers and improve container network stability.

Notes and Constraints

Specificati ons	Tunnel Network	VPC Network	Cloud Native Network 2.0	Cloud Native Network 2.0 + DataPlane V2
Supported cluster versions	All versions	Clusters of v1.19.10 and later	Clusters of v1.19.10 and later	v1.27.16-r10, v1.28.15-r0, v1.29.10-r0, v1.30.6-r0, or later
Egress bandwidth limitation	Supported	Supported	Supported	Supported
Ingress bandwidth limitation	Supported	Supported	Supported	Not supported
Scenarios where bandwidth limitation is not supported	None	None	 Pod access to cloud service CIDR blocks such as 100.125.0.0/16 Pod health check 	 Pod access to cloud service CIDR blocks such as 100.125.0.0/ 16 Pod health check
Bandwidth limitation range	Only the rate Gbit/s is supp and 1 Gbit/s. and the maxi	limit in the unit o ported, for example The minimum val mum value is 4.29	The minimum value is 1 Kbit/s, and the maximum value is 1 Pbit/s.	

The following table lists the bandwidth limitation specifications of pods.

- After DataPlane V2 network acceleration is enabled, pods on the nodes running Huawei Cloud EulerOS 2.0 use EDT to limit the egress bandwidth. The ingress bandwidth limitation is not supported. In other network modes, a Token Bucket Filter (TBF) qdisc is used to limit the bandwidth.
- Pod bandwidth limitation applies to regular containers (runC as the container runtime), not secure containers (Kata Containers as the container runtime).
- Pod bandwidth limitation does not apply to hostNetwork pods.

Procedure

Step 1 Edit a YAML file for a workload.

vi deployment.yaml

Set the network bandwidth for the pod in spec.template.metadata.**annotations** to limit the network traffic of the container. For details about the network bandwidth limit fields, see **Table 12-1**.

If the parameters are not specified, the network bandwidth is not limited by default.

An example is as follows:

apiVersion: apps/v1
kind: Deployment
metadata:
name: nginx
spec:
replicas: 1
selector:
matchLabels:
app: nginx
strategy:
type: RollingUpdate
template:
metadata:
labels:
app: nginx
annotations:
Ingress bandwidth
kubernetes.io/ingress-bandwidth: 100M
Egress bandwidth
kubernetes.io/egress-bandwidth: 1G
spec:
containers:
- image: nginx
imagePullPolicy: Always
name: nginx
imagePullSecrets:
 name: default-secret

Table 12-1 Fields for limiting the network bandwidth of pods

Field	Description	Mandatory
kubernetes.io/ingress- bandwidth	Ingress bandwidth for a pod. Value range: 1k-1P. If this field is set to a value greater than 32 Gbit/s, the actual ingress bandwidth that a pod can use is 32 Gbit/s.	No

Field	Description	Mandatory
kubernetes.io/egress- bandwidth	Egress bandwidth for a pod. Value range: 1k-1P. If this field is set to a value greater than 32 Gbit/s, the actual egress bandwidth that a pod can use is 32 Gbit/s.	No

Step 2 Create a workload.

kubectl create -f deployment.yaml

Information similar to the following is displayed:

deployment.apps/nginx created

----End

12.8 Configuring the /etc/hosts File of a Pod Using hostAliases

Application Scenarios

If DNS or other related settings are inappropriate, you can use **hostAliases** to overwrite the resolution of the hostname at the pod level when adding entries to the **/etc/hosts** file of the pod.

Procedure

- Step 1 Use kubectl to connect to the cluster.
- Step 2 Create the hostaliases-pod.yaml file.

vi hostaliases-pod.yaml

The field in bold in the YAML file indicates the image name and tag. You can replace the example value as required.

```
apiVersion: v1
kind: Pod
metadata:
name: hostaliases-pod
spec:
 hostAliases:
 - ip: 127.0.0.1
  hostnames:
  - foo.local
  - bar.local
 - ip: 10.1.2.3
  hostnames:
  - foo.remote
  - bar.remote
 containers:
   - name: cat-hosts
    image: tomcat:9-jre11-slim
    lifecycle:
     postStart:
```

exec: command: - cat - /etc/hosts imagePullSecrets: - name: default-secret

Table 12-2 pod field description

Parameter	Mandatory	Description
apiVersion	Yes	API version number
kind	Yes	Type of the object to be created
metadata	Yes	Metadata definition of a resource object
name	Yes	Name of a pod
spec	Yes	Detailed description of the pod. For details, see Table 12-3 .

Table 12-3 spec field description

Parameter	Mandatory	Description
hostAliases	Yes	Host alias
containers	Yes	For details, see Table 12-4 .

Table 12-4 containers field description

Parameter	Mandatory	Description	
name	Yes	Container name	
image	Yes	Container image name	
lifecycle	No	Lifecycle	

Step 3 Create a pod.

kubectl create -f hostaliases-pod.yaml

If information similar to the following is displayed, the pod is created.

pod/hostaliases-pod created

Step 4 Query the pod status.

kubectl get pod hostaliases-pod

If the pod is in the **Running** state, the pod is successfully created.

		CTATUC	DECTADE	
NAME	READY	SIATUS	RESTART	S AGE
hostaliases-pod	1/1	Runnina	0	16m

Step 5 Check whether the hostAliases functions properly.

docker ps |grep hostaliases-pod

docker exec -ti Container ID /bin/sh

root@hostaliases	s-pod:/# cat /eto	c/hosts
# Kubernetes-mar	aged hosts file.	_
127.0.0.1	localhost	
::1 localhos	t ip6-localhost	<pre>ip6-loopback</pre>
fe00::0 ip6-loca	alnet	
fe00::0 ip6-mcas	stprefix	
fe00::1 ip6-allr	odes	
fe00::2 ip6-allr	routers	
10.0.0.25	hostaliases-pod	
# Entries added	by HostAliases.	
127.0.0.1	foo.local	bar.local
10.1.2.3	foo.remote	bar.remote

----End

12.9 Configuring Domain Name Resolution for CCE Containers

This section describes how to configure domain name resolution for CCE containers.

Service

 Create a Service before you create a workload (Deployment or ReplicaSet). When the Kubernetes starts a container, it provides environment variables that point to all the Services that are running when the container is started. For example, if a Service named foo exists, all containers will obtain the following variables when they are initialized.
 FOO_SERVICE_HOST=<the host the Service is running on> FOO_SERVICE_PORT=<the port the Service is running on>

Therefore, you must create a Service first. Otherwise, the environment variables do not take effect. This restriction does not apply to DNS.

- CCE clusters provide the coredns add-on as the DNS server. The DNS server monitors the Kubernetes APIs for the new Services and creates a set of DNS records for each Service. If DNS is enabled throughout the cluster, all pods will be able to automatically resolve the names of Services.
- Do not specify a hostPort for a pod unless necessary. When a pod is bound to a hostPort, the number of locations to which the pod can be scheduled will be limited because each <*hostIP*, *hostPort*, *protocol*> must be unique. If you do

not specify **hostIP** and **protocol**, Kubernetes uses 0.0.0.0 as the default host IP address and TCP as the default protocol.

If you only need to access the port for debugging, you can use apiserver proxies or kubectl port-forward.

If you want to open the pod port on the node, consider using the NodePort Service before using hostPort.

- Do not use hostNetwork. The reason is the same as that of using hostPort.
- When kube-proxy load balancing is not required, use headless Services (ClusterIP set to None) for service discovery.

DNS

By default, CCE provides a DNS add-on Service named coredns to automatically assign DNS domain names for other Services. If it is running in the cluster, run the following command to check the status:

kubectl get services coredns --namespace=kube-systemNAMETYPECLUSTER-IPEXTERNAL-IPPORT(S)AGEkube-dnsClusterIP10.0.0.10<none>53/UDP,53/TCP8m

If it is not running, run the **kubec describe** command to check the reason why the pod is not started. Assume that there is a Service that has a permanent IP address and a DNS server (coredns cluster add-on) that assigns domain name to the IP address. In this way, any pod in the cluster can communicate with the Service. You can run another application for testing. Enable a new pod, access it, and run the **curl** command to check whether the domain name of the Service can be properly resolved. In some cases, the **curl** command cannot be executed due to the DNS search principles and configuration.

When a pod is created on the CCE console, not all dnsConfig configurations are opened and some default values of the pod domain name resolution parameters are used. You need to know well the default configurations. A typical case is **ndots**. If the number of dots is within the **ndots** threshold range, the domain name is considered as an internal domain name of the Kubernetes cluster and the **..svc.cluster.local** suffix is added to the domain name.

DNS Search Principles and Rules

DNS configuration file: /etc/resolv.conf

nameserver 10.247.x.x search default.svc.cluster.local svc.cluster.local cluster.local options ndots:3

Parameters:

- nameserver: domain name resolution server
- **search**: domain name suffix search rule. More search configurations indicate more matching times for domain name resolution. For example, if three suffixes are matched, at least six search operations are required because both IPv4 and IPv6 addresses need to be checked.
- **options**: domain name resolution option. Multiple KV values are available. A typical case is **ndots**. If the number of dots in the domain name to be accessed exceeds the value of **ndots**, the domain name is considered as a

complete domain name and is directly parsed. If the number of dots is less than the value of **ndots**, the suffix **..svc.cluster.local** will be added.

Parameters in Kubernetes dnsConfig

- nameservers: a list of IP addresses that will be used as DNS servers for the pod. A maximum of three IP addresses can be specified. If pod's dnsPolicy is set to None, the list must contain at least one IP address, otherwise this property is optional. The servers listed will be combined to the base nameservers generated from the specified DNS policy with duplicate addresses removed.
- **searches**: a list of DNS search domains for hostname lookup in the pod. This property is optional. When specified, the provided list will be merged into the base search domain names generated from the chosen DNS policy. Duplicate domain names are removed. Kubernetes allows for at most 6 search domains.
- **options**: an optional list of objects where each object may have a **name** property (required) and a **value** property (optional). The contents in this property will be merged to the options generated from the specified DNS policy. Duplicate entries are removed.

For details, see **DNS for Services and Pods**.

Pod DNS Policies

DNS policies can be configured on a per-pod basis. Supported DNS policies are **Default**, **ClusterFirst**, and **None**.

- **Default**: The DNS configuration of a pod inherits from the host. That is, the DNS configuration of the pod is the same as that of the node.
- ClusterFirst: Unlike the Default policy, the ClusterFirst policy writes kubedns (or CoreDNS) information to the DNS configuration of a pod in advance. ClusterFirst is the default pod policy. If PodPolicy is not specified for the pod, dnsPolicy is preset to ClusterFirst. However, ClusterFirst is mutually exclusive with HostNetwork=true. If HostNetwork is set to true, the ClusterFirst policy will be forcibly changed to the Default policy.
- None: This policy will clear the DNS configuration preset for a pod. If **dnsPolicy** is set to **None**, Kubernetes does not load any DNS configuration that is determined by its own logic in advance for the pod. Therefore, if you want to set **dnsPolicy** to **None**, you are advised to set **dnsConfig** to describe custom DNS parameters. This setting ensures that the pod has DNS configuration.

DNS configuration scenarios are provided as follows:

Scenario 1: Using a custom DNS

The following example allows you to use a custom DNS to resolve the application domain name configuration in pods. After application migration, you do not need to modify the configuration.

apiVersion: v1 kind: Pod metadata: namespace: default name: dns-example spec:

```
containers:

- name: test

image: nginx

dnsPolicy: "None"

dnsConfig:

nameservers:

- 1.2.3.4

searches:

- ns1.svc.cluster.local

- my.dns.search.suffix

options:

- name: ndots

value: "2"

- name: edns0
```

Scenario 2: Using the Kubernetes DNS add-on CoreDNS

The DNS service of Kubernetes is preferentially used for domain name resolution. If the resolution fails, the DNS service of an external cascading system is used for domain name resolution.

apiVersion: v1 kind: Pod metadata: namespace: default name: dns-example spec: containers: - name: test image: nginx dnsPolicy: ClusterFirst

Scenario 3: Using the public network domain name resolution

This mode applies to the scenario where the domain names in pods are to be accessed from public networks. In this case, the applications in the pods resolve domain names from an external DNS.

apiVersion: v1 kind: Pod metadata: namespace: default name: dns-example spec: containers: - name: test image: nginx dnsPolicy: Default

Scenario 4: Using hostNetwork

If **hostNetwork: true** is used to configure the pod network, the network interfaces on the host node are exposed to the application running in the pod. All network interfaces on the LAN where the host node is located can be used to access the application.

```
apiVersion: extensions/v1beta1
kind: Deployment
metadata:
name: nginx
spec:
template:
metadata:
labels:
app: nginx
spec:
hostNetwork: true
```

dnsPolicy: ClusterFirstWithHostNet containers: - name: nginx image: nginx:1.7.9 ports: - containerPort: 80

If **dnsPolicy: ClusterFirstWithHostNet** is not added, even if the pod uses the DNS of the host node by default, it cannot access other pods in the Kubernetes cluster through the Service.

CoreDNS Configuration

1. Configuring the CoreDNS ConfigMap

The default CoreDNS configuration file is as follows:

```
Corefile: |

.:53 {

errors

health

kubernetes cluster.local in-addr.arpa ip6.arpa {

pods insecure

upstream

fallthrough in-addr.arpa ip6.arpa

}

prometheus :9153

forward . /etc/resolv.conf

cache 30

loop

reload

loadbalance
```

Parameters:

- **error**: Errors are recorded in stdout.
- health: The CoreDNS running status report can be obtained from http:// localhost:8080/health.
- **kubernetes**: The CoreDNS returns a DNS query response based on the IP addresses of the Kubernetes Service and pod.
- **prometheus**: The measurement standard of CoreDNS can be found in the metrics in the format of http://localhost:9153/Prometheus. You can obtain monitoring data in Prometheus format from http://localhost:9153/metrics.
- **proxy** and **forward**: Any query that is not in the Kubernetes cluster domain is forwarded to the predefined resolver (/etc/resolv.conf). If the domain name cannot be resolved locally, query the upper-level address. By default, the **/etc/ resolv.conf** configuration of the host machine is used.
- cache: The front-end cache is enabled.
- **loop**: Simple forwarding loops are detected. If a loop is detected, the CoreDNS process is stopped.
- **reload**: The changed Corefile can be automatically reloaded. After editing the ConfigMap, wait for two minutes for the modification to take effect.
- **loadbalance**: This is a round-robin DNS load balancer that randomizes the order of A, AAAA, and MX records in the answer.

2. Configuring an external DNS server

Some services are not in the Kubernetes environment and need to be accessed through the DNS. The name suffix suffix is **carey.com**.

```
carey.com:53 {
     errors
     cache 30
     proxy . 10.150.0.1
}
Complete configuration file:
Corefile: |
 .:53 {
    errors
    health
    kubernetes cluster.local in-addr.arpa ip6.arpa {
      pods insecure
      upstream
      fallthrough in-addr.arpa ip6.arpa
    }
    prometheus :9153
    forward . /etc/resolv.conf
    cache 30
    loop
    reload
    loadbalance
 }
 carey.com:53 {
    errors
    cache 30
    proxy . 10.150.0.1
 }
```

CCE supports stub domains, which are more flexible and convenient than directly editing ConfigMaps. With stub domains, you do not need to pay attention to the domain name resolution configuration of pods.

12.10 Using Dual-Architecture Images (x86 and Arm) in CCE

Background

CCE allows you to create x86 and Arm nodes in the same cluster. Due to different underlying architectures, Arm images (applications) cannot run on x86 nodes, and vice versa. As a result, workloads may fail to be deployed in the clusters containing x86 and Arm nodes.

Solution

To address this issue, use either of the following methods:

• Set the service affinity when you create a workload so that the pod can be scheduled to an Arm node when the Arm-based image is used or to an x86 node when the x86-based image is used.



• Build a dual-architecture image that supports both x86 and Arm architectures. When a pod is scheduled to an Arm node, the Arm variant in the image is pulled. When a pod is scheduled to an x86 node, the x86 variant in the image is pulled. A dual-architecture image has two variants but has one unified access path. When deploying a workload, you only need to specify one image path without configuring the service affinity. In this case, the workload description file is simpler and easier to maintain.



Affinity Configuration Description

When creating a node, CCE automatically adds the **kubernetes.io/arch** label to the node. This label specifies the node architecture.

kubernetes.io/arch=amd64

The value **amd64** indicates the x86 architecture, and **arm64** indicates the Arm architecture.

When creating a workload, you can configure the node affinity to schedule pods to nodes using the corresponding architecture.

You can use **nodeSelector** in the YAML file to configure the architecture.

```
apiVersion: apps/v1
kind: Deployment
metadata:
name: test
spec:
selector:
matchLabels:
```

```
app: test
template:
 metadata:
  labels:
   app: test
 spec:
  nodeSelector:
   kubernetes.io/arch: amd64
  containers:

    name: container0

   image: swr.ap-southeast-1.myhuaweicloud.com/test-namespace/defaultbackend-linux-amd64:1.5
   resources:
     limits:
      cpu: 250m
      memory: 512Mi
     requests:
      cpu: 250m
      memory: 512Mi
  imagePullSecrets:
  - name: default-secret
```

Building a Dual-Architecture Image

NOTE

To create a dual-architecture image, ensure that the Docker client version is later than 18.03.

The essence of building a dual-architecture image is to build images based on the x86 and Arm architectures separately and then build the dual-architecture image manifest.

For example, the **defaultbackend-linux-amd64:1.5** and **defaultbackend-linux-arm64:1.5** are images based on the x86 and Arm architectures, respectively.

Upload the two images to SWR. For details about how to upload an image, see **Uploading an Image Through a Container Engine Client**.

Add a tag to the original amd64 image defaultbackend-linux-amd64:1.5. docker tag defaultbackend-linux-amd64:1.5 swr.ap-southeast-1.myhuaweicloud.com/test-namespace/ defaultbackend-linux-amd64:1.5 # Add a tag to the original arm64 image defaultbackend-linux-arm64:1.5. docker tag defaultbackend-linux-arm64:1.5 swr.ap-southeast-1.myhuaweicloud.com/test-namespace/ defaultbackend-linux-arm64:1.5 # Push the amd64 image to the image repository. docker push swr.ap-southeast-1.myhuaweicloud.com/test-namespace/defaultbackend-linux-amd64:1.5 # Push the arm64 image to the image repository. docker push swr.ap-southeast-1.myhuaweicloud.com/test-namespace/defaultbackend-linux-arm64:1.5 Create a dual-architecture **manifest** file and upload it. # Enable DOCKER CLI EXPERIMENTAL. export DOCKER_CLI_EXPERIMENTAL=enabled # Create the **manifest** image file. docker manifest create --amend --insecure swr.ap-southeast-1.myhuaweicloud.com/test-namespace/ defaultbackend:1.5 swr.ap-southeast-1.myhuaweicloud.com/test-namespace/defaultbackend-linuxarm64:1.5 swr.ap-southeast-1.myhuaweicloud.com/test-namespace/defaultbackend-linux-amd64:1.5 # Add arch information to the manifest image file. docker manifest annotate swr.ap-southeast-1.myhuaweicloud.com/test-namespace/defaultbackend:1.5 swr.ap-southeast-1.myhuaweicloud.com/test-namespace/defaultbackend-linux-amd64:1.5 --arch amd64 docker manifest annotate swr.ap-southeast-1.myhuaweicloud.com/test-namespace/defaultbackend:1.5

docker manifest annotate swr.ap-southeast-1.mynuaweicloud.com/test-namespace/defaultbackend:1.5 swr.ap-southeast-1.myhuaweicloud.com/test-namespace/defaultbackend-linux-arm64:1.5 --arch arm64 # Push the **manifest** image file to the image repository. docker manifest push an --insecure swr.ap-southeast-1 myhuaweicloud.com/test-namespace/

docker manifest push -p --insecure swr.ap-southeast-1.myhuaweicloud.com/test-namespace/ defaultbackend:1.5

In this way, you only need to use the image path **swr.apsoutheast-1.myhuaweicloud.com/test-namespace/defaultbackend:1.5** when creating a workload.

- When a pod is scheduled to an x86 node, the swr.apsoutheast-1.myhuaweicloud.com/test-namespace/defaultbackend-linuxamd64:1.5 image is pulled.
- When a pod is scheduled to an Arm node, the **swr.ap**southeast-1.myhuaweicloud.com/test-namespace/defaultbackend-linuxarm64:1.5 image is pulled.

12.11 Locating Container Faults Using the Core Dump File

Application Scenarios

A core dump is when the Linux OS saves the memory status to a file after a program crashes or stops unexpectedly. You can analyze the file to locate the fault.

Generally, when a service application crashes, its container exits and is reclaimed and destroyed. Therefore, container core files need to be permanently stored on the host or cloud storage. This topic describes how to configure container core dumps.

Notes and Constraints

When a container core dump is persistently stored to OBS (parallel file system or object bucket), the default mount option **umask=0** is used. As a result, although the core dump file is generated, the core dump information cannot be written to the core file. You can configure the OBS mount option **umask=0077** to store core dump files to OBS. For details, see **Configuring OBS Mount Options**.

Enabling Core Dump on a Node

Log in to the node, run the following command to enable core dump, and set the path and format for storing core files:

echo "/tmp/cores/core.%h.%e.%p.%t" > /proc/sys/kernel/core_pattern

%h, %e, %p, and %t are placeholders, which are described as follows:

- %h: hostname (or pod name). You are advised to configure this parameter.
- %e: program file name. You are advised to configure this parameter.
- %p: (Optional) process ID
- %t: (Optional) time of the core dump

After the core dump function is enabled by running the preceding command, the generated core file is named in the format of **core.{***Host name***}.{***Program file name***}.{***Process ID***}.{***Time***}.**

You can also configure a pre-installation or post-installation script to automatically run this command when creating a node.

D NOTE

EulerOS 2.3 Systemd has a **bug** that affects container core dump. To use core dump, perform the following operations:

- 1. In the /usr/lib/systemd/system/docker.service file on the node, change the value of LimitCORE to infinity.
- 2. Restart Docker.
- 3. Redeploy service containers.

Permanently Storing Core Dumps

A core file can be stored in your host (using a hostPath volume) or cloud storage (using a PVC). The following is an example YAML file for using a hostPath volume. apiVersion: v1 kind: Pod metadata:

name: coredump spec: volumes: - name: coredump-path hostPath: path: /home/coredump containers: - name: ubuntu image: ubuntu:12.04 command: ["/bin/sleep","3600"] volumeMounts: - mountPath: /tmp/cores name: coredump-path

Create a pod:

kubectl create -f pod.yaml

Verification

1. Obtain the pod name:

kubectl get pod

Information similar to the following is displayed:

NAME READY STATUS RESTARTS AGE coredump 1/1 Running 0 56s

2. Access the container: kubectl exec -it *coredump* -- /bin/sh

After accessing the container, run the following command to trigger a segmentation fault of the current shell terminal:

kill -s SIGSEGV \$\$

Information similar to the following is displayed:

command terminated with exit code 139

 Log in to the node and check whether the core file has been generated in / home/coredump. ls /home/coredump

If information similar to the following is displayed, the core file has been generated.

core.coredump.bash.18.1650438992
12.12 Configuring Parameters to Delay the Pod Startup in a CCE Turbo Cluster

Application Scenarios

In a CCE Turbo cluster, the routing rules of the peer pod may take effect slowly in some specific scenarios like cross-VPC and private line interconnections. To avoid this problem, configure parameters to delay the pod startup.

You can also create enterprise routers to connect to the peer VPCs. For details, see **Connecting a Cluster to the Peer VPC Through an Enterprise Router**.

Notes and Constraints

Only the CCE Turbo clusters of the following versions support the configuration of this parameter:

- v1.19: v1.19.16-r40 or later
- v1.21: v1.21.11-r0 or later
- v1.23: v1.23.9-r0 or later
- v1.25: v1.25.4-r0 or later

Using kubectl

You can add annotations to a workload to configure whether to delay the pod startup.

apiVersion: apps/v1
kind: Deployment
metadata:
name: nginx
spec:
replicas: 10
selector:
matchLabels:
app: nginx
strategy:
rollingUpdate:
maxSurge: 25%
maxUnavailable: 25%
type: RollingUpdate
template:
metadata:
labels:
app: nginx
annotations:
cni.yangtse.io/readiness-delay-seconds: "20"
spec:
containers:
- name: container-0
image: nginx:alpine
resources:
limits:
cpu: 100m
memory: 200Mi
requests:
cpu: 100m
memory: 200Mi

In this example, the Deployment needs to be configured to access the IP addresses of the cross-VPC VMs. The maximum number of replicas of this Deployment is **10** and the maximum rolling upgrade surge is **25%**, which is, the maximum number of pods to be upgraded concurrently is 13. In the annotation, the pod startup delay is set to **20s**. This ensures that the cross-VPC network can be accessed normally after the pod is started.

Table 12	- 5 Con	figuring	the	annotation
----------	----------------	----------	-----	------------

Annotation	Default Value	Description	Value Range
cni.yangtse.io/readiness- delay-seconds	None	Indicates the waiting time for the pod health check.	0-60

12.13 Automatically Updating a Workload Version Using SWR Triggers

Application Scenarios

CCE and SWR work together to enable automatic application updates. Whenever images are updated, the applications that are built from them can be automatically updated. This could be realized by adding a trigger to the desired images.

Prerequisites

A containerized application has been created on CCE and a container image has been deployed in the application.

If no workload is created, log in to the CCE console and create one. For details, see **Creating a Deployment** or **Creating a StatefulSet**.

Procedure

- **Step 1** Log in to the SWR console.
- **Step 2** In the navigation pane, choose **My Images** and click the name of the target image.
- **Step 3** Click the **Triggers** tab and click **Add Trigger**. On the page displayed, configure the parameters by referring to **Table 12-6** and click **OK**.

Figure 12-7 Adding a trigger

Add Trigger	Add Trigger									
1 Trigger au	to application updates when their corre	sponding images are updated.								
Name	Enter a trigger name.									
Condition 🧿	All ~									
Action	Update container image									
Status	Enable Disable									
Туре	CCE									
Target	Cluster	Namespace	Application	Container						
	-Select V	-Select V	-Select V	-Select V						



Table 12-6 Trigger

Parameter	Description
Name	The name of a trigger. The name can contain 1 to 64 characters, and must start with a letter. Only letters, digits, underscores (_), and hyphens (-) are allowed. The name cannot end with an underscore or hyphen. Consecutive underscores or hyphens are not allowed and an underscore cannot be placed payt to a hyphen

Parameter	Description
Condition	The following trigger conditions are supported:
	 All: Deployment is triggered when any image tags are generated or updated.
	• Specified : Deployment is triggered when a specific image tag is generated or updated.
	• RegEx : Deployment is triggered when an image tag that matches the regular expression is generated or updated. The regular expression rules are as follows:
	 - *: matches any field that does not contain the path separator /.
	 **: matches any field that contains the path separator /.
	 - ?: matches any single character except /.
	- {option 1, option 2,}: matches multiple options.
Action	Currently, only operation of updating images will be triggered. You need to specify the application to be updated and the container of the application.
Status	Select Enable .
Туре	Select CCE .
Target	Select the application whose image is to be updated and its container.

----End

Example 1: The trigger condition is All.

A Deployment named **Nginx** is created using the Nginx v1 image. The Deployment provides service to external systems with a welcome page displaying **Hello**, **SWR**!

$\leftarrow \rightarrow \mathbf{C} \mathbf{A}$:9990
--	-------

Hello, SWR!

1. Add a trigger to the Nginx image.

Set **Name** to **All_tags**, **Condition** to **All**, and select the application and all its containers that use the Nginx image.

2. The Nginx v2 image is pushed to SWR. The welcome page of the Deployment created using this new image should display **Hello, SoftWare Repository for Container!**

Figure 12-8 Image tag v2

Tags	Select the image tags you want to delete.	³ ermissions	Sharing	Triggers	Retention	Synchronization Records			
	chronize Delete								
Q E	nter an image tag.								00
	Tag ⊖			Size		Image Pull Command	Updated \varTheta	Operation	
	v2			160.9 MB		docker pull swr.cn-north-7.myhuaweic	Apr 07, 2024 14:51:04 GMT+08:00	Scan Sync More ~	

3. Check whether the Deployment is triggered successfully.

On the **Triggers** tab, locate the trigger and click **Records** to check whether the trigger is successful.

Figure 12-9 Result

Records			×
			Q
Тад	Result	Triggered	Operation
v2	Successful	May 09, 2024 21:23:33 GMT+08:00	View Details

The welcome page of the Deployment displays Hello, SoftWare Repository for Container!

←	\rightarrow	С	A	:9990
---	---------------	---	---	-------

Hello, SoftWare Repository for Container!

Example 2: The trigger condition is RegEx.

A Deployment named **nginx** is created using the Nginx image v0. The Deployment provides service to external systems with a welcome page displaying **Hello, SWR!**



Hello, SWR!

1. Add a trigger to the Nginx image.

Set **Name** to **Tags_regular_expression**, **Condition** to **RegEx**, enter the **^v2.*** regular expression, and select the application and all its containers that use the Nginx image.

×

Figure 12-10 Selecting RegEx

Add Trigger				
1 Trigger au	to application updates when their corres	sponding images are updated.		
Name	Tags_regular_expression			
Condition ⑦	RegEx ~	^v2.*		
Action	Update container image			
Status	Enable Disable			
Туре	CCE CCI			
Target	Cluster	Namespace	Application	Container
	test-turbo-v25-001 🗸	default 🗸	nginx V	nginx v

2. Push the Nginx image v1 to SWR. The welcome page of the Deployment created using this new image should display **Hello**, **SWR!** (v1).

Figure 12-11 Image tag v1

Tags Description Pull/Push Permissions Sharing Tr	iggers Retention Synchronia	ration Records		
Synchronize Delete				
Q.1				× Q 🛞
□ Tag ⊕	Size	Image Pull Command	Updated O	Operation
□ v1	9.0 MB	docker pull swr.on-north-7.myhuaweicloud.com/cosign/nginx.v1	May 30, 2024 09:47:26 GMT+08:00	Scan Sync More Y
Total Records: 1 10 \checkmark (1) \rightarrow				

On the **Triggers** tab, click \checkmark to check the result. The workload redeployment of the Nginx image v1 is not triggered.

The workload access page still displayed Hello, SWR!.



Hello, SWR!

3. Push the Nginx image v2 to SWR. The welcome page of the Deployment created using this new image should display **Hello**, **SWR! (v2)**.

Figure 12-12 Image tag v2

Tags Description Pull/Push Permissions S	iharing Triggers Retention	Synchronization Records		
Synchronize Delete				
C. Enter an image tag.				
□ Teg ⊖	Size	Image Pull Command	Updated O	Operation
□ v2	9.0 MB	docker pull swr.cn-north-7.myhuaweicloud.com/cosign/ngirx: v2	May 30, 2024 09:47:37 GMT+08:00	Scan Sync More *
v1	9.0 MB	docker pull switch-north-7.myhuaweicloud.com/cosign/ngint:v1	May 30, 2024 09:47:26 GMT+08:00	Scan Sync More ~

4. Check whether the Deployment is triggered successfully.

On the **Triggers** tab, click \checkmark to check the result. As shown in Figure 12-13, only the deployment of the Nginx image v2 is triggered.

Figure 12-13 Result

My Images 7 nginx						Q
			Tag	Result	Triggered	Operation
Name	nginx		v2	 Successful 	May 30, 2024 09:49:59 GMT+08:00	View Details
Туре	Private					
Tags	2					
Space Used	18.0 MB					
Tags Desc	ription Pull/Push Permissions	Sharing Triggers Ret	ention Syn			
Add Trigger	Export ~					
Q. Select a prop	erty or enter a keyword.					
Name 🖯	Status 😔	Condition Type \ominus	Conditio			
🗹 test-ngiro	e Enabled	All	-			
Total Records: 1	10 ✓ < (1) →					

The welcome page of the Deployment displays Hello, SWR! (v2).



12.14 Effective Troubleshooting in Kubernetes with Temporary Containers

Pods are the basic building blocks in Kubernetes. Pods are disposable and replaceable. Once a pod is created, containers cannot be added to it. Ephemeral containers are special containers in Kubernetes. They are used to temporarily create containers in running pods for debugging.

In most cases, container exceptions can be debugged using the **kubectl exec** or **kubectl logs** command to access the container and diagnose faults. However, the **exec** command may fail if containers are in a **Crash** state or if their images lack debugging tools. In such situations, ephemeral containers offer a practical solution. These containers can be injected into a running pod to inspect its status and execute commands, aiding in the resolution of issues that are difficult to replicate.

Prerequisites

- The cluster version should be v1.23 or later and the ephemeral containers are enabled for the cluster by default.
- kubectl has been installed. You have been granted the required cluster access permissions. For details, see Accessing a Cluster Using kubectl.
- You have prepared a debugging tool image. The images of the preset tool packages are recommended.
 - container-trouble-shooting: a public image provided by the Huawei Cloud container team. This image is preconfigured with rich network diagnosis tools, performance diagnosis, and development environments, such as GDB, Python, Delve, strace, tcpdump, traceroute, telnet, Nmap, bind-utils, iPerf3, net-tools, ethtool, iftop, pstack, GCC, Golang, and perf.

Image address: swr-gallery.swr-pro.myhuaweicloud.com/library/container-trouble-shooting:v1

nicolaka/netshoot: a network diagnosis tool set (tcpdump, netstat, curl, and more)

Assigning Permissions (RBAC)

To ensure only minimum privileges are assigned, allowing a user or service account to modify only the temporary containers of pods:

Step 1 Create a **role-test.yaml** file and grant only the permissions to modify the temporary containers to the user.

vi role-test.yaml

The content is as follows: apiVersion: rbac.authorization.k8s.io/v1 kind: ClusterRole metadata: name: ephemeral-debugger rules: - apiGroups: [""] resources: ["pods/ephemeralcontainers"] verbs: ["update", "patch"] # Only the temporary containers of the pods can be modified. apiVersion: rbac.authorization.k8s.io/v1 kind: ClusterRoleBinding metadata: name: debugger-binding subjects: - kind: User name: "xxx" # User ID apiGroup: rbac.authorization.k8s.io roleRef: kind: ClusterRole name: ephemeral-debugger apiGroup: rbac.authorization.k8s.io

For details about how to obtain a user ID, see **Obtaining Account, IAM User, Group, Project, Region, and Agency Information**.

Step 2 Create the preceding RBAC configurations.

kubectl create -f role-test.yaml

----End

Injecting a Temporary Container

Step 1 Inject a temporary container into a pod and enter the interactive shell: kubectl debug *<pod-name>* -it --image=swr-gallery.swr-pro.myhuaweicloud.com/library/container-troubleshooting:v1 --target=*<target-container>*

The parameters in this command are described as follows:

- -it: Access the interactive terminal.
- --image: Specify the image of the temporary container.
- --target: (Optional) Share the process namespace of the target container with the temporary container.

For example, to debug the **nginx** container in the **myapp** pod, run the following command:

kubectl debug myapp -it --image=swr-gallery.swr-pro.myhuaweicloud.com/library/container-trouble-shooting:v1 --target=nginx

You can also debug containers using a pod by running the following command:

kubectl debug myapp -it --image=swr-gallery.swr-pro.myhuaweicloud.com/library/container-trouble-shooting:v1 --share-processes --copy-to=myapp-debug

In the preceding command:

- --share-processes: enables the temporary container to share the namespace with the pod, which allows you to view other container processes in the pod in the temporary container.
- --copy-to: creates a pod replica, copies the temporary container to the new pod, and specifies the name of the new pod.

After these commands are executed, information similar to the following is displayed: Pod/myapp created

```
Defaulting container name to debugger.
If you do not see a command prompt, try pressing Enter.
/ #
```

Step 2 Start the debugging and use various debugging tools to check the container status. For details about typical scenarios, see Typical Problem Diagnosis Scenarios.

----End

Typical Problem Diagnosis Scenarios

• Network Problem Check

Scenario: A pod is unable to access external or internal services.

- a. Use a debugging tool image to examine the network: kubectl debug *myapp*-it --image=swr-gallery.swr-pro.myhuaweicloud.com/library/containertrouble-shooting:v1
- b. Run the following command within the temporary container to check the network:

tcpdump -i eth0 port 80 # Capture HTTP traffic. netstat -tuln # Check the listening status of the port. dig my-service.namespace.svc.cluster.local # DNS curl -v http://backend:8080 # Test service connectivity.

• File System Check

Scenario: Logs from the main container are missing, or the configuration file is faulty.

- a. Create a temporary container and mount the file system (shared volumeMounts) of the main container. kubectl debug *myapp* -it --image=busybox --target=nginx
- b. Check the files. Is /var/log/nginx # View the log directory. cat /etc/nginx/nginx.conf # Verify the configuration file content.

• Process/Performance Analysis

Scenario: Abnormal CPU or memory usage is observed. Tools such as htop, strace, and perf are used.

- a. Create a temporary container and share the process namespace of the target container with it using the **--target** parameter. kubectl debug *myapp*-it --image=alpine --target=nginx
- b. Run the following command within the temporary container to check the process:

View the process tree.
ps aux
Monitor resource usage.
top -H
Trace system calls.
strace -p 1 # PID 1 is the main Nginx process.

12.15 Developing and Testing a Microservice Locally Using Telepresence

Kubernetes applications are typically composed of multiple separate services, each running in its own container. This can complicate development and debugging on a remote Kubernetes cluster, as you may need to **access a shell on a running container** to execute debugging tools.

In a typical Kubernetes development workflow, developers must repeat the following steps:

- 1. Modify the code.
- 2. Build a new image.
- 3. Push the image to the repository.
- 4. Update the Deployment.
- 5. Wait for pods to restart.
- 6. Verify the changes.

This process can take several minutes or even longer, which greatly slows down iteration.

Telepresence is a tool that injects Traffic Agents as sidecar containers into workload pods. These Traffic Agents function as proxies, rerouting network traffic between a CCE cluster and your local environment. This setup enables you to develop and test your applications locally using tools such as your preferred debugger and integrated development environment (IDE), as if your local environment were part of the CCE cluster.

This section describes how to develop and debug services running on a CCE cluster using Telepresence. For details, see **Code and debug an application locally**.

Prerequisites

You have created a CCE cluster and accessed the cluster using kubectl in your local environment. For details, see **Accessing a Cluster Using kubectl**.

Installing Telepresence

Step 1 Install the Telepresence client in your local environment.

- Download the Telepresence binary file. The installation procedure varies by OS. AMD64 is used as an example. For details about more installation methods, see Client Installation. sudo curl -fL https://github.com/telepresenceio/telepresence/releases/latest/download/telepresencelinux-amd64 -o /usr/local/bin/telepresence
- 2. Add execute permissions to the file. sudo chmod a+x /usr/local/bin/telepresence

Step 2 Install Telepresence Traffic Manager in the CCE cluster. For details, see Install/ Uninstall the Traffic Manager.

Run the following command in your local environment where the Telepresence client was installed: telepresence helm install

----End

Configuring an Interception Rule

- **Step 1** Access kube-apiserver of the CCE cluster in your local development environment. telepresence connect
- **Step 2** Create a Deployment as an example, which is deployed using an Nginx image. kubectl create deployment echo-server --image=nginx --port=80 --replicas=1

Information similar to the following is displayed:

deployment.apps/echo-server

Step 3 Create a Service and associate it with the Deployment.

The following is an example YAML file: apiVersion: v1 kind: Service metadata: labels: app: echo-server name: echo-server namespace: default spec: internalTrafficPolicy: Cluster ipFamilies: - IPv4 ipFamilyPolicy: SingleStack ports: name: http port: 80 protocol: TCP targetPort: 80 selector: app: echo-server sessionAffinity: None type: ClusterIP

In the cluster, use cURL to test the connectivity of the ClusterIP Service. If the access is successful, information similar to the following will be displayed.

```
[root@dev-cluster-nodepool-90885-mtoz6 ~]# curl 10.247.120.106
<!DOCTYPE html>
<html>
<head>
<title>Welcome to nginx!</title>
<style>
    body {
        width: 35em;
        margin: 0 auto;
        font-family: Tahoma, Verdana, Arial, sans-serif;
    }
</style>
</head>
<body>
<h1>Welcome to nginx!</h1>
If you see this page, the nginx web server is successfully installed and
working. Further configuration is required.
For online documentation and support please refer to
<a href="http://nginx.org/">nginx.org</a>.<br/>br/>Commercial support is available at
<a href="http://nginx.com/">nginx.com</a>.
<em>Thank you for using nginx.</em>
</body>
</html>
```

Step 4 Specify the traffic to be intercepted. The following command shows an example. This command establishes a rule to redirect traffic sent to the **echo-server** Service to **localhost:80**. By doing so, you can debug the functionality of the **echo-server** Service in your local environment by listening on **localhost:80**. telepresence intercept echo-server --port 80:80

Information similar to the following is displayed:

Using Deployment echo-server Intercept name : echo-server State : ACTIVE Workload kind : Deployment Intercepting : 192.168.1.215 -> 127.0.0.1 80 -> 80 TCP

Step 5 Write an HTTP server in the local IDE to listen on port 80.



Step 6 Access the **echo-server** Service in the cluster again. If information similar to the following is displayed, the traffic has been sent to the local server.

[root@dev-cluster-nodepool-90885-mtoz6 ~]# curl 10.247.120.106 telepresence test, i'am local[root@dev-cluster-nodepool-90885-mtoz6 ~]#

The logs printed by the local server indicate that the requests forwarded from the cluster have been received.

----End

12.16 Containerizing and Using QingTian Enclave on CCE

QingTian Enclave is a next-generation security structure introduced to QingTianbacked ECSs. QingTian Enclave instances are completely independent VMs without persistent storage, interactive access, or external networking. They ensure the temporary, secure storage of data, provide a highly secure, isolated execution environment, and can run sensitive tasks and applications. For details, see What Is QingTian Enclave?

QingTian Enclave has the following features:

- **Completely independent VMs**: A QingTian Enclave instance is an independent VM built within an ECS. It has its own OS and resources and is logically isolated from the host ECS to ensure high security.
- **No persistent storage**: QingTian Enclave instances have no persistent storage. All data is lost after the instances are shut down. This ensures the temporary, secure storage of data, preventing data leakage.
- No interactive access or external networking: QingTian Enclave instances do not support interactive access, such as SSH connections. They cannot access external networks. They can only communicate with the host ECSs through internal secure channels.
- **High isolation and access control**: The **root** user on the host ECSs cannot directly access the QingTian Enclave instances. Any user or process on a host ECS must communicate with the QingTian Enclave instances through a secure local channel (vsock). This channel ensures the security and integrity of data transmission and prevents man-in-the-middle attacks and data leakage.

QingTian Enclave instances are applied to:

- **Sensitive data processing**: They can process sensitive data, such as encryption keys and personal identity information. They ensure that the data is not disclosed during the processing.
- **Secure computing**: They provide an isolated environment for highly secure computing tasks to prevent interference from malicious software or attackers.
- **Privacy protection**: They can protect your private data and ensure the security of sensitive information during transmission and processing.

Typically, a QingTian Enclave instance can be directly created on a QingTianbacked ECS. However, this setup does not support dynamic instance scaling and cannot automatically adjust based on fluctuations in service volume. This section describes how to containerize and use a QingTian Enclave instance in a CCE cluster and use Kubernetes scheduling and scaling capabilities for service scalability.

Solution Architecture

To deploy QingTian Enclave in a CCE cluster, install the add-on for containerizing QingTian Enclave and package a QingTian Enclave image file (.eif) into a container image. After completing these steps, you can create QingTian Enclave containers within the CCE cluster. kubelet allocates and schedules such containers through the **/dev/qtbox_service0** API for managing QingTian Enclave instance lifecycle.

Compared with running QingTian Enclave instances on VMs, running QingTian Enclave containers splits services into untrusted and trusted components. The untrusted components include external service components of an application and components related to QingTian enclave instance management (such as the qt-enclave tool). The trusted components include services that involve encryption and decryption or sensitive information protection and run in a QingTian Enclave instance.

When running QingTian Enclave containers, qt-enclave is used to create a QingTian Enclave instance through a secure local channel (vsock) to run services that require encryption, decryption, or sensitive information protection.



Figure 12-14 Using QingTian Enclave in a CCE cluster

Prerequisites

A CCE standard or Turbo cluster is available.

Managing a QingTian Enclave Instance in a CCE Cluster

Step 1 Log in to the ECS console, create an ECS that supports the QingTian Enclave feature, for example, a C7t ECS with 4 vCPUs and 8 GiB of memory or a higher flavor, and use Linux as the system image. (Huawei Cloud EulerOS 2.0 is recommended). In the Trusted Settings area, select Enclave.

Selected specifications	General computing	g-plus c7t.xlarge.2 4 vC	PUs 8 GiB						
os									
Image ⑦									
Public image	Private image	Shared image K	poGallery image						
0	85		6	0	3	200	os		0
Huawei Cloud	selles		•	C	e	-9 ¹ 1-	•	•	
EulerOS	CentOS	SUSE	Ubuntu	Debian	OpenSUSE	AlmaLinux	Rocky Linux	CentOS Stream	CoreOS
	00								
•									
openEuler	SUSESAP	Windows							
Huawei Cloud EulerO	S 2.0 Standard 64 bit(1	0 GiB)	~ Q						
Trusted Settings									
S Enclave									
Host protection (HS	iS) ⑦								
HSS provides you with a	a range of security funct	ions, such as risk preventio	n, intrusion detection, adva	nced protection, security	operations, and web page	tamper protection.			

- **Step 2** Configure the ECS to be managed by CCE and pay attention to the following settings:
 - Storage: Add a data disk of at least 20 GiB of storage space to the ECS for the container runtime and kubelet components.

Storage & Backup	
System Disk 🕜	
Disk Type	System Disk (GiB)
General Purpose SSD V	− 40 +
IOPS limit: 2,280, IOPS <u>burst limit</u> : 8,0	00 Encryption: Disabled Advanced Options
Data Disk	
Disk Type	Data Disk (GiB) Quantity
General Purpose SSD V	− 100 + − 1 + Delete
IOPS limit: 3,000, IOPS burst limit: 8,0	00 Encryption: Disabled Disk type: VBD Shared disk: Disabled Data disk image: Not used Advanced Options

• Network: Select the VPC where the cluster is located.

Network				
VPC 🕥				
vpc-cce(192.168.0.0/16)	~	Q Create VPC		
Primary NIC				
subnet-cce(192.168.0.0/24)	~	Automatically assign IP address	~	Q Available private IP addresses: 249
Add Extension NIC				
NICs you can still add: 2				

- Step 3 Log in to the CCE console and click the cluster name to access the cluster console. In the navigation pane, choose Nodes. In the right pane, click the Nodes tab and click Accept Node in the upper right corner.
- Step 4 Click Select Cloud Server and select the created ECS.

	~	Select Cloud Server						
Node Configuration Node Pool Specifications OS	Configure the specifications and OS of a cloud server, on which your containential applications run. Performance Performance P	 The selected cloud server The node is inclusted in The node is inclusted in The node is site that mutual To separately store systems To separately store systems Only cloud servers will The partitioned disk or If a node server base second 	s (nodes) must the same VPC ning or stopped stem compone han 10 GiB. have at least 3 h the same dat h the same dat the cloud sen ted for manage	meet the following requirem as the current cluster at an of lagged with CC inst, use a local disk or attach 2 vCPUs and 4 GiB memory, a disk configuration can be a ver will not be managed as a ement, CCE will automatically	ents: View Cloud Ser E-Dynamic-Provision at least one data dis and can be bound to dded in batches. data disk, please do stop it. The added s	vers C ning-Node. k of 20 GiB or larger to you only one network interface data backup and disk clear ecurity group, agency, and	ur node. Make sure that any 2. rup in advance resource tags may persist.	attached data
	Username root	Q. Enter a name , ID or single	IP		1			
	Password Enter a password.	ecs-enclave-test2 C	AZ AZ1	c7Llarge.4 Huawei Cloud Eul	192.168.0	High I/O 40 GiB	High I/O 100GiB	Pay-per-us
Storage Settings Co	nfigure storage resources for containers and applications on the node.	$(1, \dots, n) \in \mathbb{R}^{n}$	<i>8</i> .	$\mathbb{C}_{n+1}^{\infty}$	$\mathcal{T}_{\mathcal{T}}(\mathcal{T}) = \mathcal{T}$	$(1,1,1,\dots,1,n)$	100-10-10-1	0.04
System Disk	Directly use the system disk of the cloud server.	0.000	-	Million.	$\{(a,b),(a)\}$	$(a_i(a_i)) \in (a_i(a_i))$	$= \{ e_i \mid i \in [i], i \in [i], \dots, i \in [i], $	-
System Component Storage	Data Disk System Disk	(1,1,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2	1	dittan.	$(a_{ij},a_{j})_{i \in \mathbb{N}}$	10.00	10758	10.00

Step 5 Install the components that support the QingTian Enclave feature on the node by running the script and enter the following parameters in the **Post-installation Command** area in **Advanced Settings**:

curl -k https:/ addons/addon cd /tmp unzip addons cd addons bash post_ins	/cce-statics.ap-southeast-3.obs.ap-southeast-3.myhuaweicloud.com/job-platform-package/cce- ns-qt-enclave-20250417103811268.zip -o /tmp/addons-qt-enclave.zip -qt-enclave.zip tall.sh
Post-installation Command	curl + https://coe-statics.ap-southeast-3.dos.ap- southeast-3.minuaweicloud.com/job-platform- package/coe-addrons/addons-qt-enclave- 202050417103811268 zp-o.https/addons-qt-enclave-zp crl htm Comments: that affect the status of the node functioning that in the node in fail in enter the usable state, resulting in the node herein rescript affect "10 minutes. Please use it with reaction Red Practines

Step 6 Configure other parameters as required to complete node management. For details about the parameters, see **Accepting Nodes for Management**.

After the node has been managed, locate the row containing this node and choose **More** > **View YAML** in the **Operation** column. You can see that there are schedulable QingTian Enclave resources in the YAML file.



----End

Building and Using a QingTian Enclave Container Image

- Step 1 Prepare an ECS that runs Huawei Cloud EulerOS 2.0 and supports the QingTian Enclave feature and build a QingTian Enclave image file named hello-enclave.eif. For details, see Getting Started with QingTian Enclave.
 - 1. Log in to the ECS and install Docker for image builds.
 - a. Confirm that the repository is configured properly.

```
    Check whether the parameters in the /etc/yum.repos.d/hce.repo file are configured properly. The proper configuration is as follows:
    [base]

            name=HCE $releasever base
            baseurl=https://repo.huaweicloud.com/hce/$releasever/os/$basearch/
            enabled=1
            gpgcheck=1
            gpgkey=https://repo.huaweicloud.com/hce/$releasever/os/RPM-GPG-KEY-HCE-2
            [updates]
            name=HCE $releasever updates
            baseurl=https://repo.huaweicloud.com/hce/$releasever/updates/$basearch/
```

- b. Install the Docker software package. yum install docker -y
- 2. Install qt CLI, configure parameters for resource isolation in the configuration file as required, and enable resource isolation. To use qt CLI, install the necessary Python libraries. For details, see **QingTian CLI (qt CLI)**. yum install qt-enclave-bootstrap yum install virtio-qtbox yum install qingtian-tool

Install the required Python libraries.

pip3 install docker knack

3. Configure parameters for resource isolation in the configuration file and enable resource isolation. In this practice, 1 GiB of memory and 2 vCPUs are used.

systemctl start qt-enclave-env

4. Build a QingTian Enclave image. In this practice, the **hello_enclave.sh** script is used as the QingTian Enclave application.

vi hello_enclave.sh

The content is as follows:

```
#!/bin/bash
while true
do
    echo "hello enclave!"
    sleep 2
done
```

Create a Dockerfile.

vi Dockerfile

The content is as follows:

FROM ubuntu:latest COPY hello_enclave.sh /root/hello_enclave.sh CMD ["/root/hello_enclave.sh"]

- 5. Check that the script has execution permissions. chmod +x hello_enclave.sh
- 6. Build a Docker image named **hello-enclave**. docker build -f Dockerfile -t hello-enclave .
- Run the **qt enclave make-img** command to convert the Docker image to a QingTian Enclave image file named **hello-enclave.eif**. qt enclave make-img --docker-uri hello-enclave --eif hello-enclave.eif

The output is as follows:

- **Step 2** Use **hello-enclave.eif** to build a container image. The built container image has the QingTian Enclave feature.
 - Set up a container build environment. mkdir container cd container cp /usr/local/bin/qt . cp /etc/qingtian/enclave/qt-enclave-env.conf . cp /root/hello-enclave.eif .

2. Create a Dockerfile in the **container** directory. vi Dockerfile

```
The content is as follows:

FROM ubuntu

RUN apt update && \

apt install -y python3 && \

apt install -y python3-pip && \

apt install -y python3-docker && \

apt install -y python3-docker && \

apt install -y python3-knack && \

apt install -y libcjson-dev && \

apt install -y libcjb2.0-dev && \

In -s /usr/bin/python3 /usr/bin/python && \

mkdir -p /etc/qingtian/enclave && \

mkdir -p /var/log/qlog

COPY qt-enclave-env.conf /etc/qingtian/enclave/
```

COPY qt-enclave-env.conf /etc/qingtian/enclave/ COPY qt /usr/bin COPY hello-enclave.eif / COPY entrypoint.sh / ENTRYPOINT /entrypoint.sh

3. Create **entrypoint.sh** in the **container** directory. vi entrypoint.sh

The content is as follows: #!/bin/bash qt enclave start --mem 4096 --cpus 2 --eif hello-enclave.eif --cid 4--debug-mode qt enclave console --enclave-id 0

- 4. Build a container image with the QingTian Enclave feature. docker build -f Dockerfile -t enclave-demo.
- 5. Push the container image to SWR and use it in a Kubernetes cluster. For details, see **Pushing an Image**.

----End

Deploying a QingTian Enclave Container on CCE

Step 1 Create an enclave-deployment.yaml file and deploy a workload.

```
The following shows an example: (You need to replace the image address with the
one pushed to SWR during the image build.)
kind: Deployment
apiVersion: apps/v1
metadata:
 name: hello-enclave
 namespace: default
spec:
 replicas: 1
 selector:
  matchLabels:
   app: hello-enclave
   version: v1
 template:
  metadata:
   labels:
    app: hello-enclave
    version: v1
  spec:
   containers:
     - name: container-1
      image: swr.ap-southeast-3.myhuaweicloud.com/container/enclave-demo:latest
      resources:
       limits:
        cpu: 250m
        huawei.com/qt_enclaves: '1'
```

hugepages-TGI: 4GI
memory: 512Mi
requests:
cpu: 250m
huawei.com/qt_enclaves: '1'
hugepages-1Gi: 4Gi
memory: 512Mi
imagePullPolicy: IfNotPresent
securityContext:
seccompProfile:
type: Unconfined

Step 2 Create the workload.

kubectl create -f enclave-deployment.yaml

Information similar to the following is displayed:

deployment.apps/hello-enclave created

Step 3 Check the pod status.

kubectl get pod | grep hello-enclave

Information similar to the following is displayed:

hello-enclave-587b9f6d49-bvq22 1/1 Running 0 38s

Step 4 View the pod logs.

kubectl logs hello-enclave-587b9f6d49-bvq22 --tail=10

If information similar to the following is displayed, the QingTian Enclave container is functioning properly:

hello enclave! hello enclave! hello enclave! hello enclave!

----End

13 Permission

13.1 Configuring kubeconfig for Fine-Grained Management on Cluster Resources

Application Scenarios

By default, the kubeconfig file provided by CCE for users has permissions bound to the **cluster-admin** role, which are equivalent to the permissions of user **root**. It is difficult to implement refined management on users with such permissions.

Purpose

Cluster resources are managed in a refined manner so that specific users have only certain permissions (such as adding, querying, and modifying resources).

Precautions

Ensure that kubectl is available on your host. If not, download it from **here** (corresponding to the cluster version or the latest version).

Configuration Method

D NOTE

In the following example, only pods and Deployments in the **test** space can be viewed and added, and they cannot be deleted.

Step 1 Set the service account name to **my-sa** and namespace to **test**.

kubectl create sa **my-sa** -n **test**

[root@test-arm-54016 ~]# [root@test-arm-54016 ~]# kubectl create sa my-sa -n test serviceaccount/my-sa created [root@test-arm-54016 ~]#

Step 2 Configure the role table and assign operation permissions to different resources. vi role-test.yaml

The content is as follows:

NOTE

In this example, the permission rules include the read-only permission (get/list/watch) of pods in the **test** namespace, and the read (get/list/watch) and create permissions of deployments.

apiVersion: rbac.authorization.k8s.io/v1 kind: Role metadata: annotations: rbac.authorization.kubernetes.io/autoupdate: "true" labels: kubernetes.io/bootstrapping: rbac-defaults name: myrole namespace: test rules: - apiGroups: - "" resources: - pods verbs: - get - list - watch apiGroups: - apps resources: - pods - deployments verbs: - get - list - watch - create

Create a Role.

kubectl create -f role-test.yaml

```
[root@test-arm-54016 ~]# kubectl create -f role-test.yaml
role.rbac.authorization.k8s.io/myrole created
[root@test-arm-54016 ~]#
```

Step 3 Create a RoleBinding and bind the service account to the role so that the user can obtain the corresponding permissions.

vi myrolebinding.yaml

The content is as follows: apiVersion: rbac.authorization.k8s.io/v1 kind: RoleBinding metadata: name: myrolebinding namespace: test roleRef: apiGroup: rbac.authorization.k8s.io kind: Role name: myrole subjects: - kind: ServiceAccount name: my-sa namespace: test

Create a RoleBinding.

kubectl create -f myrolebinding.yaml

[root@test-arm-54016 ~]# kubectl create -f myrolebinding.yaml rolebinding.rbac.authorization.k8s.io/myrolebinding created [root@test-arm-54016 ~]#

The user information is configured. Now perform **Step 5** to **Step 7** to write the user information to the configuration file.

Step 4 Manually create a token that is valid for a long time for ServiceAccount.

vi **my-sa-token.yaml**

The content is as follows: apiVersion: v1 kind: Secret metadata: name: my-sa-token-secret namespace: **test** annotations: kubernetes.io/service-account.name: my-sa type: kubernetes.io/service-account-token

Create a token:

kubectl create -f **my-sa-token.yaml**

Step 5 Configure the cluster information.

1. Decrypt the ca.crt file in the secret and export it.

kubectl get secret **my-sa-token-secret** -n **test** -oyaml |grep ca.crt: | awk '{print \$2}' |base64 -d > /home/ ca.crt

- Set a cluster access mode. test-arm specifies the cluster to be accessed. https://192.168.0.110:5443 specifies the apiserver IP address of the cluster. For details about how to obtain the IP address, see Figure 13-1. /home/ test.config specifies the path for storing the configuration file.
 - If the internal API server address is used, run the following command: kubectl config set-cluster test-arm --server=https://192.168.0.110:5443 --certificate-authority=/ home/ca.crt --embed-certs=true --kubeconfig=/home/test.config
 - If the public API server address is used, run the following command: kubectl config set-cluster test-arm --server=https://192.168.0.110:5443 --kubeconfig=/home/ test.config --insecure-skip-tls-verify=true

froot@test-arm-54016 home]# kubectl config set-cluster test-arm --server=https://10.0.1.100:5443 --certificate-authority=/hom ca.crt --embed-certs=true --kubeconfig=/home/test.config Cluster "test-arm" set. froot@test-arm-54016 home]# _

NOTE

If you **perform operations on a node in the cluster** or **the node that uses the configuration is a cluster node**, do not set the path of kubeconfig to **/root/.kube/config**.

By default, the apiserver IP address of the cluster is a private IP address. After an EIP is bound, you can use the public network IP address to access the apiserver.

Figure 13-1 Obtaining the internal or public API server address

Connection Info

Intranet URL	https://192.167.0.108:5443
EIP	Bind
Custom SAN	02
kubectl	Configure
Certificate Authentication	X.509 certificate Download

Step 6 Configure the cluster authentication information.

1. Obtain the cluster token. (If the token is obtained in GET mode, run **based64** -**d** to decode the token.)

token=\$(kubectl describe secret my-sa-token-secret -n test | awk '/token:/{print \$2}')

2. Set the cluster user **ui-admin**.

kubectl config set-credentials ui-admin --token=\$token --kubeconfig=/home/test.config

ootQtest-arm-54016 home]# kubectl config set-credentials ui-admin --token=\$token --kubeconfig=/home/test.config er "ui-admin" set. ootQtest-arm-54016 home]#

Step 7 Configure the context information for cluster authentication access. uiadmin@test specifies the context name.

> kubectl config set-context **ui-admin@test** --cluster=**test-arm** --user=**ui-admin** --kubeconfig=/home/ test.config

root@test-arm-54016 home]# kubectl config set-context ui-admin@test --cluster=test-arm --user=ui-admin --kubeconfig=/home/test onfig ontext "ui-admin@test" created. root@test-arm-54016 home]#

Step 8 Configure the context. For details about how to use the context, see Verification. kubectl config use-context ui-admin@test --kubeconfig=/home/test.config

[paas@test-arm-54016 home]\$ kubectl config use-context ui-admin@test --kubeconfig=/home/test.config Gwitched to context "ui-admin@test". [paas@test-arm-54016 home]\$

NOTE

If you want to assign other users the above permissions to perform operations on the cluster, provide the generated configuration file **/home/test.config** to the user after performing step **Step 7**. The user must ensure that the host can access the API server address of the cluster. When performing step **Step 8** on the host and using kubectl, the user must set the kubeconfig parameter to the path of the configuration file.

----End

Verification

1. Pods in the **test** namespace cannot access pods in other namespaces. kubectl get pod -n **test** --kubeconfig=/home/test.config

[paas@test-arm-54016 home]\$	kubectl	get pod -n test ·	kubeconf ig=	/home/test.config					
NAME	READY	STATUS	RESTARTS	AGE					
test-pod-56cfcbf45b-12q92	0/1	CrashLoopBackOff	27	91m					
[paas@test-arm-54016 home]\$									
[paas@test-arm-54016 home]\$	kubectl	get podkubecom	nfig=/home/te	st.config					
Error from server (Forbidde)	i): pods	is forbidden: Use	er "system∶se	rviceaccount:test:my-sa"	cannot lis	t resource	"pods"	in AF	'I gro
up "" in the namespace "defa	ult"								
[paas@test-arm-54016 home]\$									

2. Pods in the **test** namespace cannot be deleted.

paas@test-arm-54016 home1\$ kubect1 delete pod -n test test-pod-56cfcbf45b-12q92 --kubeconfig=/home/test.config rror from server (Porbidden): pods "test-pod-56cfcbf45b-12q92" is forbidden: User "system:serviceaccount:test:my-sa" cannot del te resource "pods" in API group "" in the namespace "test" uass@test-arm-54016 home1\$

Further Readings

For more information about users and identity authentication in Kubernetes, see **Authenticating**.

13.2 Configuring Namespace-level Permissions for an IAM User

Application Scenarios

In a containerized environment, various teams and departments have different resource access needs. If the permissions are set too broadly, it can lead to issues like cross-environment occupation, misoperations, and resource competition. To address these problems, it is crucial to have precise control over permissions.

CCE permissions management offers fine-grained control over permissions using IAM and Kubernetes RBAC. It supports IAM-based fine-grained permissions control and IAM token authentication. With cluster-level and namespace-level permissions control, users' access to specific resources can be effectively restricted, ensuring resource isolation and security.

Aut hori zati on	Description
IAM auth oriza tion	IAM authorization for user groups is primarily concerned with managing access to cloud platform resources. Policies are used to control the permissions of each user group on specific resources. IAM emphasizes precise control over cloud resources.
RBA C auth oriza tion	RBAC authorization for user groups is role-based. Permissions are linked to roles, which are then assigned to user groups. RBAC authorization is typically employed for internal access control within applications. RBAC places greater emphasis on aligning roles with tasks.

Table 13-1 IAM and RBAC authorization

This example describes how to manage member account permissions at the namespace level. For more information about CCE permissions management, see **Permissions**.

Solution

Assume that there are an R&D and test team and an O&M team. The two teams need to access cluster A created by member account A and have different resource access requirements, which are listed in Table 13-2.

Table 13-2 Resource access requirement
--

Team	Permission	Policy Content
R&D and test team	Namespace A of cluster A	Access and operations on resources in Namespace A for software development and testing
O&M team	Namespace B of cluster A	Access resources in Namespace B for software development and testing

The details are as follows:

- 1. Create different user groups for different teams.
- 2. Assign different permissions to these user groups, which means, perform IAM and RBAC authorization.

Figure 13-2 Solution details



Notes and Constraints

- Before granting permissions to user groups, you need to get familiar with the system policies listed in **Permissions** for CCE. To grant permissions for other services, you need to learn about all **system-defined permissions** supported by IAM.
- Users with the **Security Administrator** permission, which includes all permissions except IAM role switching, can manage authorization settings on the namespace permissions page on the CCE console. They can also view the current user group and its permissions. The **admin** user group, for example, has this permission by default.

Step 1: Create Users and User Groups

Member account A creates users and user groups for the R&D and test team and the O&M team, to make user and resource management easier. In this example,

three users are created: development, test, and O&M users. You can create additional users as needed.

- **Step 1** Log in to the management console.
- **Step 2** Hover the cursor on the username in the upper right corner and choose **Identity and Access Management** from the drop-down list.
- **Step 3** In the navigation pane, choose **User Groups**. In the upper right corner on the displayed page, click **Create User Group**.

On the displayed page, enter a user group name and click OK.

In this example, you need to create two user groups, for example, **group1** (for the R&D and test team) and **group2** (for the O&M team).

The two new user groups are displayed in the user group list.

Figure 13-3 Viewing the new user groups

ser Groups 🕤			Create Oster Create
Detelo User groups available for creation: 1			
O, Enter a group name.			
□ Name ⊕	Users Description 0	Created 🖨	Operation
group2	Operations	Sep 13, 2024 15:21 21 GMT+60:00	Authorize Modify ManageUser Delete
C onet	Development and Testing	8ep 13, 2024 15 15 07 GM7+05:00	Authorize Modily Manage User Delete

Step 4 In the navigation pane, choose **Users**. In the upper right corner on the displayed page, click **Create User**.

Configure parameters in **Set User Details** as required and click **Next**. For details, see **Figure 13-4**.

Select the user group to which the user is to be added and click **Create**.

In this example, three users are created for the R&D, test, and O&M personnel. The R&D and test users are added to **group1**, and the O&M user is added to **group2**.

The three new users are displayed in the user list.

Set User Details	2 (Optional) Add User to Group	a 3) Finish				
User Details	The username, email address, and mobile num	iber can be used as login credentials.				
	* Username	Email Address	Mobile Number	Description	External Identity ID	Operatio
	develop	Enter an email address.	+86 (Chines V Enter a mobile number.	Enter a brief description.	Enter an external identity ID.	Delete
	Add User Users available for addition: 1					
Access Type	Programmatic access					
	Allows access to Huawei Cloud services o	nly by using development tools, such as APIs, C	CLI, and SDKs, and requires an access key or password. 👩			
	Management console access Allows access to Huawei Cloud services o	nly by using the management console and requ	uires a password.			
Conducted Torus	Management console access Allows access to Huawei Cloud services o Arceas Key	nly by using the management console and requ	uires a password.			
Credential Type	Management console access Allows access to Huawei Cloud services o Access key You can download the access key after yo	nly by using the management console and requ u create the user.	ures a password.			
Credential Type	Management console access Allows access to Huawei Cloud services o Access key You can download the access key after yo Password	nly by using the management console and requ u create the user.	uires a password.			
Credential Type	Anagement console access Alarous access to Huawe Coud services o Access tey Vou can download the access key after yo Password	nly by using the management console and require to the user.	ures a password			
Credential Type	Allows access to Huavei Cloud services o Allows access to Huavei Cloud services o Access key You can download the access key after yo Passend Set now Friter a bassaind	nly by using the management console and required to react the user.	ures a password			
Credential Type	Shangement console access Advess access to Husenic Cloud services o Access key You can download the access key after yo Passound Set now Enter a password exet Set pair Revier password exet af the to	nly by using the management console and requ u create the user.	ures a passoord			
Credential Type	Management console access Alows access to Hause Cloud services o Access key Wo can deventual the access key after yo Passavid Set now End y passavid Set now Access key after yo	ny ty using the management console and require or oracle the user.	ures a passoord			
Credenial Type	Management console access Alona access to Hause Cloud services o Access key You can deverted the access key after you Passand Enter a password Enter a password Charging password Apassword will be automatically generated Apassword will be automatically generated	rly by using the management console and regular or and regular or and the user.	ures a password			
Credential Type	Management console access Mona access to Hausen Cloud services o Access key You can download the access key after you Panacod Enter a password Enter a password Accession et al. If the access here Accession et al. If the accession of the accession Accession of the accession of the accession of the accession Accession of the accession of the accessio	rhy by using the management console and regular or state the user.	ures a password			
Credential Type	Management console access Mona access to Husen Cloud services of Access tery Tora can develop the access key after you Person of Set now Enter a passion of Activity access and services of the Approximated the automaticality gene Activity access and the analyticality gene Activity access and the analyticality gene Activity access access access access access access access and Access access a	nly by using the management console and regular or state the user.	utes a password d provide it to the user.			

Figure 13-4 Creating a user

Figure 13-5 Viewing the new users

Users 💮				Q Getting to Know Identity Policies and Trast Agencie	a on the New Console > ()+ Go to New Cons	ole @ Feedback Create L	нт
UMI User Legin Link: https://www.icoud.com/authol/	lagin Ya-hundad, pag, cikilania, pagi 🗇						
Database Monthly Expert Users	available for creation: 44						
Username v 🔍 🔍 Enter a user	NAPA.						8
Username 0	Description 0	Status 0	Last.Activity (i)	Created @	Operation		
Orvesse		C Exalled		5ep 13, 2024 15.46.17 GMT+05.00	Authorize Modify Security Settings C	elete	
operation		Enabled		Sep 13, 2024 15:36 89 GMT+60:00	Authorize Modify Security Settings E	leista	
		Constant of Constant		Ferr 43, 2024 (F-20, 20, 20, 20, 20, 20, 20, 20, 20, 20,	And and a start for the former of		

----End

Step 2: Perform IAM Authorization for the User Groups

Member account A grants IAM permissions to user groups **group1** and **group2** and manages user group permissions based on cloud services.

- **Step 1** In the navigation pane, choose **User Groups**, locate the user group to be authorized, and click **Authorize**.
- **Step 2** On the displayed page, select **Cloud Container Engine (CCE)** in the upper right corner.

Select a policy as required. For more information about **CCE FullAccess** and **CCE ReadOnlyAccess** policies, see **System-defined Policies**.

In this example, select the CCE FullAccess and CCE ReadOnlyAccess policies for group1, and select the CCE ReadOnlyAccess policy for group2.

Figure 13-6 IAM authorization

1 Select	PhilipBate 2 Indext Score 3 Presid								
Assign sele	cled permissions to group1.								Create Policy
Vent	Keelel (2) Cay Personans Ivan Joulier Project		Al policesitoles	× (Cloud Container Engine	Pully M.	v) Enter a policy in	arre, role name, or descr	anter Q
•	Poly/Fole Kane	Ture							
•	COI Administrator Cood Container Engine Administrator	System-defined role							
•	COE Factorem Commer operation permissions on CCE cluster resources, excluding the namespeca-level permissions for the clusters (with Nationnelse REAC enables) and the phyloged administrator spendion, such as appeny; configuration and cluster coefficient pr.	System-defined policy							
•	COE ReadOny-Access Premission to view Claud Container Expine cluster resources, accluding the namespace-level permission of the clusters (with Kaberneles FERA evabled)	System-defined policy							

Step 3 Click **Next**, select a more refined scope as required (for example, **All resources**), and click **OK**.

Table 13-3 Aut	horization scopes

Solution	Description
All resources	IAM users will be able to use all resources, including those in enterprise projects, region-specific projects, and global services under your account based on assigned permissions.
Enterprise projects	IAM users will be able to use resources in the selected enterprise projects based on assigned permissions. For example, an enterprise project may contain resources that are deployed in different regions. After you associate the enterprise project with the IAM users, they can access the resources in this enterprise project based on the assigned permissions. IAM users will be able to use resources in the selected enterprise projects based on assigned permissions.
Region- specific projects	IAM users will be able to use resources in the selected region- specific projects based on assigned permissions. IAM users will be able to use resources in the selected region- specific projects based on assigned permissions.

----End

Step 3: Perform RBAC Authorization for the User Groups

Member account A performs RBAC authorization on the two user groups for namespace-level permissions management. This ensures that user groups with different roles have the minimum permissions, improving system security and management efficiency.

- **Step 1** Click in the upper left corner and choose **Cloud Container Engine** to access the CCE console.
- **Step 2** In the navigation pane, choose **Permissions**. In the right pane, select a **cluster** to be authorized and a **namespace** and click **Add Permission** in the upper right corner.

i igui e i e	, enorming .	operations of		page	
CCE	Permissions 🥱				Quick Links Add Perm
Clusters					
Permissions Cloud Native Cost Governance OBT Add-ons SoftWare Repository for	Administrator Read and write permissions on all resources in all namespaces.	O&M Read and write permissions for most resources in all namespaces, and read-only permissions for nodes, storage volumes, namespaces, and quotas.	Developer Read and write permissions for most resources in all or selected namespaces. When configured for all namespaces, they are equal to the O&M permissions.	 Viewer Read-only permissions for most resources in all or selected namespaces. 	Custom Permissions available are determined by the ClusterRole or Role you select.
	Delete	•	ce-example CCE Standard	V Namespace V	Search keyword Q
	User/User Group \ominus	Permission Type	Namespace	e 😔 🔰 🖉	Operation
	User	🐼 Administrator	All namesp	eaces E	dit Delete

Figure 13-7 Performing operations on the Permissions page

Step 3 On the displayed page, confirm the cluster name, select the user or user group to be authorized, and select a namespace to be used for cluster authorization. In this example, select the default namespace for group1 and the test namespace for group2.

You can select a permission type based on your requirements. For details about permission types, see **Namespace Permissions (Kubernetes RBAC-based)**. In this example, you can select **Custom**.

- 1. Click Add Custom Role.
- Configure the custom role. Table 13-4 lists the custom role parameters of group1 and group2.

Name	example1		
Туре	ClusterRole	Role	
Namespac e	default		
Rule	All operations: * Read-only: get + list + w Read-write: get + list + w	ratch vatch + create + update + patch + delete	
	$\fbox{g} \times \fbox{I} \times \fbox{wat} \times$	persistentvolumeclai ×	Delete
	* ×	endpoi × po × servic × eve × namespac ×	Delete
	+ Add		

Figure 13-8 Adding a custom role

 Table 13-4 Description

Parame ter	Example Value	Description
Name	group1: example1 group2: example2	Name of a custom role

Parame ter	Example Value	Description
Туре	group1: Role group2: Role	Custom roles are classified into ClusterRole and Role. Each ClusterRole or Role contains a group of rules that represent related permissions. For details, see Using RBAC Authorization.
		 ClusterRole: a cluster-level resource that can be used to configure cluster access permissions.
		 Role: used to configure access permissions in a namespace. When creating a Role, specify the namespace to which the Role belongs.
		In this example, you only need to configure namespace permissions. Therefore, you can choose Role .
Rule	group1: – get, list, watch: persistentvolume- claims – *: resources except persistentvolume-	The information on the left indicates the permissions to be granted. The right part indicates the resources to which the permissions are granted. You can configure this parameter based on the actual requirements.
	claims group2: – get, list, watch: *	

3. Click OK.

Step 4 On the **Add Permission** page, select the newly created role for **Custom** and click **OK**.

In this example, two permissions, **example1** and **example2**, need to be created and assigned to **group1** and **group2**, respectively.

The new permissions are displayed in the permissions list.

Figure 13-9 Viewing the permissions list

CCE	Permissions ⊘				Childrichina Add Parminian
Clusters					
Permissions Claud Native Cost Governance	Administrator Read and write permissions on all resources in all namespaces.	OBM Read and write pervisoises for most resources in at namespaces, and read only permissions for nodes, storage volumes, namespaces, and quotes.	Developer Read and write provisions for most ensurines in all or selected namespaces. When configured for all namespaces, they are equal to the OAM permissions.	Vewer Read.only permissions for most resources in all or selected namespaces.	Coston Permissions available are determined by the Christeffair or Pairs you select.
Add-one					
Soft/Vane Repository for Cardaleer 13	Debte			the manyle CCR Standa	al v Ramesane v South Report Q Q
	User/User Group (-)	Permission Type (i)	Namespace 0	Op	infon
	User Group group1	ecanple1	default.	Ea	Conte
	User Group group2	Signuo	(default)	ter	Coate
	. un -uliu	😏 Administrator	All namespaces	Ed	Deate

----End

Step 4: Verify Permissions

Log in to the management console as the user created using **Step 1: Create Users and User Groups** and check whether the user has the required permissions. This section uses the **develop** user created previously as an example to verify whether the permissions have been configured.

Step 1 Log in to the management console as the newly created user.

Bind a mobile number following instructions and verify the login. Whether or not a mobile number needs to be linked to an account depends on the choice made during the initial setup of basic user information.

Reset the password following instructions (required for the first login).

Figure 13-10 Logging in to the management console

en de sérvicientes	
develop	
	Ø
Log	In
Forgot Password	Remember me

Use Another Account: HUAWEI ID | Federated User

IAM User Login

- **Step 2** Click in the upper left corner and choose **Cloud Container Engine** to access the CCE console.
- **Step 3** Check whether the **develop** user can access other clusters except **cce-example**.

Click the name of another cluster. If you see a message stating that the user does not have the necessary permissions, it means the user cannot access other clusters.

Figure 13-11 No permissions



Step 4 Check whether the **develop** user can access the **cce-example** cluster.

Click the name of the **cce-example** cluster to access the cluster console. In the navigation pane, choose **Namespace**. In the right pane, only the **default** namespace (namespace for which permissions are configured) is displayed.

Figure 13-12 Viewing namespaces

Namespaces							Quick Links Create Namespace
Export	r a keyword.						
Namespace ()	Status ()	Source ()	Label \ominus	Node Affinity	Created ()	Description ()	Operation
default	O Running	Default	kubernetes ioimetadata name detauit node-local-dna-injection enabled	Disable Z	6 hours ago		Edit YAML Manage Quota More 🛩
Total Records: 1							

Step 5 Check whether the **develop** user can create a PVC.

In the navigation pane, choose **Storage**. In the right pane, click the **PVs** tab and click **Create PVC** in the upper right corner.

In the window that slides out from the right, configure related parameters and click **Create**. A message is displayed, indicating that the user does not have the permissions. The **develop** user only has permissions to view the PVCs, but does not have permissions to perform any operations on them because of the permissions configured in **Step 3: Perform RBAC Authorization for the User Groups**.

Figure 13-13 PVC creation failed due to insufficient permissions



Step 6 Check whether the develop user can upgrade a workload.

In the navigation pane, choose **Workloads**. In the workload list, locate the row containing the **wordpress1** workload and click **Upgrade**.

Change the original image tag from **php7.3** to **latest** and click **Upgrade Workload**.

Figure 13-14 Changing the image tag of a workload

en Co	ntainer - 1				+ Add Container
	Basic Info	Container	wordpress	Pull Policy	Awara (2)
	Lifecycle	Name			- · ·
	Health Check	Image Name	wordpress Replace Image	Image Tag	latest V
	Environment Variables	CPU Quota	Request All cores; Limit All cores 🕥	Memory	Request All MB; Limit All MB ()
	Data Storage			Quota	
	Security	GPU Quota	This function is unavailable because add-on GPU is not installed. Install now	NPU Quota	This function is unavailable because add-on NPU is not installed. Install now
	Logging	Privileged Container	0	Init Container	0

In the workload list, check whether the image tag of **wordpress1** has been changed to **latest**. If it is, the workload has been upgraded, and the operation permissions of the **develop** user have been configured successfully.

Figure 13-15 wordpress1 after upgrade

Deployments StatefulSets DaernonSets Jobs	Cron Jobs Pods					Over Lines Create Non 12ML Create Workload
Delete Bala Redeplay Expert -						
O. Select a property or order a keyword.						0.00
■ Worklaad Name ⊕	Status	Pada (Normal/MI) ()	Namespace ()	Created ()	Image Name ()	Operation
·	O Running	111.	(setual)	.15. dam. aux	🧼 wordpress labert	Monitor: View Log. Upgrade: More \sim
vordpress 1	O Running	111.	default	35. days. aga	🖝 mondpress Sales	Monitor: Weiv Log: Upgrade: Moni \sim
 and a set 	O Running	111.	(select)	.15. dam. aux	🔹 nyaçû l	Monitor: View Log. Upgrade: More \sim
· · • • •	O Running	212.	(setual)	35. dam. aga	🧼 wonlpress php? 3	Monitor: New Log. Upgrade: Now \sim
Total Records: 4 10 V C 1 >						

----End

Step 5: Clear Resources

If a user group no longer requires permissions, you can delete the permissions of the user group using member account A or even delete the created user and user group.

Step 1 Delete RBAC authorization.

Log in to the management console as member account A.

Click in the upper left corner and choose **Cloud Container Engine** to access the CCE console.

In the navigation pane, choose **Permissions**. On the page displayed, select a **cluster** and a **namespace**.

In the permissions list, locate the row containing the target permission and click **Delete**. In the dialog box displayed, click **Yes**.

Step 2 Delete a user group.

Hover the cursor on the username in the upper right corner and choose **Identity and Access Management** from the drop-down list.

In the navigation pane, choose **User Groups**. In the right pane, locate the row containing the target user group and click **Delete**.

In the displayed dialog box, enter **DELETE** and click **OK**.

Step 3 Delete a user.

In the navigation pane, choose **Users**. In the right pane, locate the row containing the target user and click **Delete**. You can also click **Edit** to disable the user and enable it again as required.

In the displayed dialog box, enter **DELETE** and click **OK**.

----End

Background

CCE permissions are classified into cluster permissions and namespace permissions. Namespace permissions are based on Kubernetes RBAC and can be used to grant permissions on resources in clusters and namespaces.

Currently, the CCE console provides four types of namespace-level ClusterRole permissions by default: cluster-admin, admin, edit, and view. However, these permissions apply to resources in the namespace regardless of resource types (pods, Deployments, and Services).

Solution

Kubernetes RBAC enables you to easily control permissions on namespace resources.

- Roles: define a set of rules for accessing Kubernetes resources in a namespace.
- RoleBindings: define the relationship between users and roles.
- ClusterRoles: define a set of rules for accessing Kubernetes resources in a cluster (including all namespaces).
- ClusterRoleBindings: define the relationship between users and cluster roles.

Roles and ClusterRoles specify actions that can be performed on specific resources. RoleBindings and ClusterRoleBindings bind Roles to specific users, user groups, or service accounts.



Figure 13-16 Role binding

The user in the preceding figure can be an IAM user or user group in CCE. You can efficiently control permissions on namespace resources through RoleBindings.

The section describes how to use Kubernetes RBAC to grant user **user-example** with the permission for viewing pods. (This is the only permission the user has.)
Prerequisites

RBAC is supported only on clusters of v1.11.7-r2 or later.

Creating an IAM User and User Group

Log in to the IAM console and create an IAM user named **user-example** and a user group named **cce-role-group**. For details about how to create an IAM user and user group, see **Creating an IAM User and Logging In** and **Creating a User Group and Assigning Permissions**.

User Groups / cce-role-group	
Name cce-role-group	Group ID 0c96fad22880f32a3f84c009862af6f7
Description 🖉	Created Jun 05, 2021 16:20:56 GMT+08:00
Permissions Manage permissions for the user group. Add	Users Manage users in the user group.
Username J≡	Description JΞ
user-example	-

Grant the **CCE FullAccess** permission to the **cce-role-group** user group. For details about how to grant permissions to a user group, see **Creating a User Group and Assigning Permissions**.

User Groups / cce-role-group		
Name cce-role-group	Group ID (Dc96fad22880f32a3f84c009862af6f7 □
Description 🖉	Created J	lun 05, 2021 16:20:56 GMT+08:00
Permissions Manage permissions for the user group.	Users Manage users in the	user group.
Assign Permissions		All policies/roles
Policy/Role Name	Type JΞ	Description ↓=
✓ CCE FullAccess	System-defined policy	Common operation permissions on CCE cluster resources, excludin

CCE FullAccess has the permissions for cluster operations (such as cluster creation), but does not have the permissions to operate Kubernetes resources (such as viewing pods).

Creating a Cluster

Log in to CCE and create a cluster.

NOTICE

Do not use IAM user **user-example** to create a cluster because CCE automatically assigns the cluster-admin permissions of all namespaces in the cluster to the user who creates the cluster. That is, the user can fully control the resources in the cluster and all its namespaces.

Log in to the CCE console as IAM user **user-example**, **download the kubectl configuration file in the cluster and access the cluster**, and run the following command to obtain the pod information. (The output shows that **user-example** does not have the permission to view the pods or other resources.) This indicates that **user-example** does not have the permissions to operate Kubernetes resources.

kubectl get pod

Error from server (Forbidden): pods is forbidden: User "0c97ac3cb280f4d91fa7c0096739e1f8" cannot list resource "pods" in API group "" in the namespace "default"

kubectl get deploy

Error from server (Forbidden): deployments.apps is forbidden: User "0c97ac3cb280f4d91fa7c0096739e1f8" cannot list resource "deployments" in API group "apps" in the namespace "default"

Creating a Role and RoleBinding

Log in to the CCE console, **download the kubectl configuration file in the cluster and access the cluster**, and create a Role and RoleBinding.

NOTE

Log in as the account used to create the cluster because CCE automatically assigns the cluster-admin permissions to the account, which means that the account has the permissions to create Roles and RoleBindings. Alternatively, you can use IAM users who have the permissions to create Roles and RoleBindings.

The definition of a Role is simple. You just specify a namespace and some rules. For example, the following rules allow you to perform GET and LIST operations on pods in the **default** namespace.

- apiGroups indicates the API group to which the resource belongs.
- **resources** indicates the resources that can be operated. Pods, Deployments, ConfigMaps, and other Kubernetes resources are supported.
- **verbs** indicates the operations that can be performed. **get** indicates querying a specific object, and **list** indicates listing all objects of a certain type. Other value options include **create**, **update**, and **delete**.

For details, see Using RBAC Authorization.

After creating a Role, you can bind the Role to a specific user, which is called RoleBinding. The following shows an example:

The **subjects** section binds a Role with an IAM user so that the IAM user can obtain the permissions defined in the Role, as shown in the following figure.





You can also specify a user group in the **subjects** section. In this case, all users in the user group obtain the permissions defined in the Role.

subjects: - kind: Group name: 0c96fad22880f32a3f84c009862af6f7 # User group ID apiGroup: rbac.authorization.k8s.io

Verification

Use IAM user **user-example** to connect to the cluster and view the pods. The pods can be viewed.

kubectl get pod READY STATUS RESTARTS AGE NAMF nginx-658dff48ff-7rkph 1/1 Running 0 4d9h nginx-658dff48ff-nidhi Running 0 4d9h 1/1 # kubectl get pod nginx-658dff48ff-7rkph NAME READY STATUS RESTARTS AGE nginx-658dff48ff-7rkph 1/1 Running 0 4d9h

Try querying Deployments and Services in the namespace. The output shows **user-example** does not have the corresponding permissions. Try querying the pods in namespace kube-system. The output shows **user-example** does not have the corresponding permission, either. This indicates that the IAM user **user-example** has only the GET and LIST Pod permissions in the **default** namespace, which is the same as expected.

kubectl get deploy
Error from server (Forbidden): deployments.apps is forbidden: User "0c97ac3cb280f4d91fa7c0096739e1f8"

cannot list resource "deployments" in API group "apps" in the namespace "default" # kubectl get svc

Error from server (Forbidden): services is forbidden: User "0c97ac3cb280f4d91fa7c0096739e1f8" cannot list resource "services" in API group "" in the namespace "default" # kubectl get pod --namespace=kube-system

- Error from server (Forbidden): pods is forbidden: User "0c97ac3cb280f4d91fa7c0096739e1f8" cannot list resource "pods" in API group "" in the namespace "kube-system"

14_{Release}

14.1 Overview

Background

When switching between old and new services, you may be challenged in ensuring the system service continuity. If a new service version is directly released to all users at a time, it can be risky because once an online accident or bug occurs, the impact on users is great. It could take a long time to fix the issue. Sometimes, the version has to be rolled back, which severely affects user experience.

Solution

Several release policies are developed for service upgrade: grayscale release, bluegreen deployment, A/B testing, rolling upgrade, and batch suspension of release. Traffic loss or service unavailability caused by releases can be avoided as much as possible.

This document describes the principles and practices of grayscale release and blue-green deployment.

 Grayscale release, also called canary release, is a smooth iteration mode for version upgrade. During the upgrade, some users use the new version, while other users continue to use the old version. After the new version is stable and ready, it gradually takes over all the live traffic. In this way, service risks brought by the release of the new version can be minimized, the impact of faults can be reduced, and quick rollback is supported.

The following figure shows the general process of grayscale release. First, divide 20% of all service traffic to the new version. If the service version runs normally, gradually increase the traffic proportion and continue to test the performance of the new version. If the new version is stable, switch all traffic to it and bring the old version offline.



L

If an exception occurs in the new version when 20% of the traffic goes to the new version, you can quickly switch back to the old version.



 Blue-green deployment provides a zero-downtime, predictable manner for releasing applications to reduce service interruption during the release. A new version is deployed while the old version is retained. The two versions are online at the same time. The new and old versions work in hot backup mode. The route weight is switched (0 or 100) to enable different versions to go online or offline. If a problem occurs, the version can be quickly rolled back.



Realizing Grayscale Release or Blue-Green Deployment

Kubernetes-native features can be used to implement simple grayscale release or blue-green deployment. For example, you can change the value of the label that determines the service version in the selector of a Service to change the pod backing the Service. In this way, the service can be directly switched from one version to another. If you have complex grayscale release or blue-green deployment requirements, you can deploy service meshes and open-source tools, such as Nginx Ingress and Traefik, to the cluster. You can obtain detailed instructions in the following sections:

- Using Services to Implement Simple Grayscale Release and Blue-Green Deployment
- Using Nginx Ingress to Implement Grayscale Release and Blue-Green Deployment

Impleme ntation	Application Scenario	Feature	Disadvantage
Service	Simple release requirements and small-scale test scenarios	No need to introduce too many plug-ins or complex configurations	Manual operations and poor automation

Table 14-1 Implementation mode comparison

Impleme ntation	Application Scenario	Feature	Disadvantage
Nginx Ingress	No special requirements.	 Only the annotation supported by Nginx Ingress needs to be configured. Header-based, cookie- based, and service weight- based traffic division policies are supported 	The nginx-ingress add-on needs to be installed in the cluster, which consumes resources.

Both Services and Nginx Ingresses use open-source Kubernetes capabilities to implement grayscale release and blue-green deployment. In this process, CCE allows you to easily perform the following operations:

- All resources can be created, viewed, and modified on the console, which is more intuitive than the kubectl command line tool.
- LoadBalancer Services are supported by the ELB service. When creating a Service, you can use an existing ELB instance or create a new one.
- The nginx-ingress add-on can be installed in just a few clicks, and ELB load balancers can be created and interconnected during the installation.

14.2 Using Services to Implement Simple Grayscale Release and Blue-Green Deployment

To implement grayscale release for a CCE cluster, deploy other open source tools, such as Nginx Ingress, to the cluster or deploy services to a service mesh. These solutions are difficult to implement. If your grayscale release requirements are simple and you do not want to introduce too many plug-ins or complex configurations, you can refer to this section to implement simple grayscale release and blue-green deployment based on native Kubernetes features.

Principles

Users usually use Kubernetes objects such as Deployments and StatefulSets to deploy services. Each workload manages a group of pods. The following figure uses Deployment as an example.



Generally, a Service is created for each workload. The Service uses the selector to match the backend pod. Other Services or objects outside the cluster can access the pods backing the Service. If a pod needs to be exposed, set the Service type to LoadBalancer. The ELB load balancer functions as the traffic entrance.

• Grayscale release principles

Take a Deployment as an example. A Service, in most cases, will be created for each Deployment. However, Kubernetes does not require that Services and Deployments correspond to each other. A Service uses a selector to match backend pods. If pods of different Deployments are selected by the same selector, a Service corresponds to multiple versions of Deployments. You can adjust the number of replicas of Deployments of different versions to adjust the weights of services of different versions to achieve grayscale release. The following figure shows the process:



• Blue-green deployment principles

Take a Deployment as an example. Two Deployments of different versions have been deployed in the cluster, and their pods are labeled with the same key but different values to distinguish versions. A Service uses the selector to select the pod of a Deployment of a version. In this case, you can change the value of the label that determines the version in the Service selector to change the pod backing the Service. In this way, you can directly switch the service traffic from one version to another. The following figure shows the process:



Prerequisites

The Nginx image has been uploaded to SWR. The Nginx images have two versions: v1 and v2. The welcome pages are **Nginx-v1** and **Nginx-v2**.

Resource Creation

You can use YAML to deploy Deployments and Services in either of the following ways:

- On the **Create Deployment** page, click **Create YAML** on the right and edit the YAML file in the window.
- Save the sample YAML file in this section as a file and use kubectl to specify the YAML file. For example, run the **kubectl create -f xxx.yaml** command.

Step 1: Deploy Services of Two Versions

Two versions of Nginx services are deployed in the cluster to provide external access through ELB.

Step 1 Create a Deployment of the first version. The following uses nginx-v1 as an example. Example YAML:

apiVersion: apps/v1 kind: Deployment metadata: name: nginx-v1	
spec:	
replicas: 2 selector: matchLabels: app: nginx	# Number of replicas of the Deployment, that is, the number of pods # Label selector
version: vi	
template:	
metadata:	# De d Jahal
app: nginx version: v1	# Pod ladel
spec:	
containers:	
 image: {your_i name: contain 	repository}/nginx:v1 # The image used by the container is nginx:v1. er-0
resources:	
limits:	
cpu: 100m	
memory: 20)0Mi
requests:	
cpu: 100m	
memory: 20	JOMi
imagePullSecret	S:
- name: default-	-secret

Step 2 Create a Deployment of the second version. The following uses nginx-v2 as an example. Example YAML:

apiVersion: apps/v1 kind: Deployment metadata: name: nginx-v2 spec:	
replicas: 2	# Number of replicas of the Deployment, that is, the number of pods
selector:	# Label selector
matchLabels:	
app: nginx	
version: v2	
template:	
metadata:	
labels:	# Pod label
app: nginx	
version: v2	
spec:	
containers:	
- image: {your_i name: contain	repository}/nginx:v2 # The image used by the container is nginx:v2. er-0
resources:	
limits:	
cpu: 100m	
memory: 20	/0Mi
requests:	
cpu: 100m	
memory: 20	/OMi
imagePullSecret	S:
 name: default- 	-secret

You can log in to the CCE console to view the deployment status.

----End

Step 2: Implement Grayscale Release

Step 1 Create a LoadBalancer Service for the Deployment. Do not specify the version in the selector. Enable the Service to select the pods of the Deployments of two versions. Example YAML:

```
apiVersion: v1
kind: Service
metadata:
 annotations:
  kubernetes.io/elb.id: 586c97da-a47c-467c-a615-bd25a20de39c # ID of the ELB load balancer. Replace it
with the actual value.
 name: nginx
spec:
 ports:
 - name: service0
  port: 80
  protocol: TCP
  targetPort: 80
                  # The selector does not contain version information.
 selector:
  app: nginx
  type: LoadBalancer # Service type (LoadBalancer)
```

Step 2 Run the following command to test the access:

for i in {1..10}; do curl <EXTERNAL_IP>; done;

<EXTERNAL_IP> indicates the IP address of the ELB load balancer.

The command output is as follows (Half of the responses are from the Deployment of version v1, and the other half are from version v2):

- Nginx-v2 Nginx-v1 Nginx-v1 Nginx-v2 Nginx-v2 Nginx-v2 Nginx-v2 Nginx-v2 Nginx-v2 Nginx-v2
- **Step 3** Use the console or kubectl to adjust the number of replicas of the Deployments. Change the number of replicas to 4 for v1 and 1 for v2.

kubectl scale deployment/nginx-v1 --replicas=4

kubectl scale deployment/nginx-v2 --replicas=1

Step 4 Run the following command to test the access again:

for i in {1..10}; do curl <EXTERNAL_IP>; done;

<EXTERNAL_IP> indicates the IP address of the ELB load balancer.

In the command output, among the 10 access requests, only two responses are from the v2 version. The response ratio of the v1 and v2 versions is the same as the ratio of the number of replicas of the v1 and v2 versions, that is, 4:1. Grayscale release is implemented by controlling the number of replicas of services of different versions.

Nginx-v1 Nginx-v1 Nginx-v1 Nginx-v2 Nginx-v2 Nginx-v2 Nginx-v1 Nginx-v1 Nginx-v1

NOTE

If the ratio of v1 to v2 is not 4:1, you can set the number of access times to a larger value, for example, 20. Theoretically, the more the times, the closer the response ratio between v1 and v2 is to 4:1.

----End

Step 3: Implement Blue-Green Deployment

Step 1 Create a LoadBalancer Service for a deployed Deployment and specify that the v1 version is used. Example YAML:

apiVersion: v1 kind: Service metadata:	
kubernetes.io/elb.id: 586c97da-a47c-467c-a615-bd25a20de39c	# ID of the ELB load balancer. Replace it
with the actual value.	·
name: nginx	
spec:	
ports:	
- name: service0	
port: 80	
protocol: TCP	
targetPort: 80	
selector: # Set the version to v1 in the selector. app: nginx version: v1	
type: LoadBalancer # Service type (LoadBalancer)	

Step 2 Run the following command to test the access:

for i in {1..10}; do curl <EXTERNAL_IP>; done;

<EXTERNAL_IP> indicates the IP address of the ELB load balancer.

The command output is as follows (all responses are from the v1 version):

Nginx-v1 Nginx-v1 Nginx-v1 Nginx-v1 Nginx-v1 Nginx-v1 Nginx-v1 Nginx-v1 Nginx-v1 Nginx-v1

Step 3 Use the console or kubectl to modify the selector of the Service so that the v2 version is selected.

kubectl patch service nginx -p '{"spec":{"selector":{"version":"v2"}}}'

Step 4 Run the following command to test the access again:

for i in {1..10}; do curl <EXTERNAL_IP>; done;

<EXTERNAL_IP> indicates the IP address of the ELB load balancer.

The returned results show that are all responses are from the v2 version. The bluegreen deployment is successfully implemented.

Nginx-v2 Nginx-v2 Nginx-v2 Nginx-v2 Nginx-v2 Nginx-v2 Nginx-v2 Nginx-v2 Nginx-v2 Nginx-v2

----End

14.3 Using Nginx Ingress to Implement Grayscale Release and Blue-Green Deployment

This section describes the scenarios and practices of using Nginx Ingress to implement grayscale release and blue-green deployment.

Application Scenarios

Nginx Ingress supports three traffic division policies based on the header, cookie, and service weight. Based on these policies, the following two release scenarios can be implemented:

• Scenario 1: Split some user traffic to the new version.

Assume that Service A that provides layer-7 networking is running. A new version is ready to go online, but you do not want to replace the original Service A. You want to forward the user requests whose header or cookie contains **foo=bar** to the new version of Service A. After the new version runs stably for a period of time, you can gradually bring the new version online and smoothly bring the old version offline. The following figure shows the process:



Service A

Service A'

• Scenario 2: Split a certain proportion of traffic to the new version.

Assume that Service B that provides layer-7 services is running. After some problems are resolved, a new version of Service B needs to be released. However, you do not want to replace the original Service B. Instead, you want to switch 20% traffic to the new version of Service B. After the new version runs stably for a period of time, you can switch all traffic from the old version to the new version and smoothly bring the old version offline.



Annotations

Nginx Ingress supports release and testing in different scenarios by configuring annotations for grayscale release, blue-green deployment, and A/B testing. The implementation process is as follows: Create two ingresses for the service. One is a common ingress, and the other is an ingress with the annotation **nginx.ingress.kubernetes.io/canary: "true"**, which is called a canary ingress. Configure a traffic division policy for the canary ingress. The two ingresses cooperate with each other to implement release and testing in multiple scenarios. The annotation of Nginx Ingress supports the following rules:

nginx.ingress.kubernetes.io/canary-by-header

Header-based traffic division, which is applicable to grayscale release. If the request header contains the specified header name and the value is **always**, the request is forwarded to the backend service defined by the canary ingress. If the value is **never**, the request is not forwarded and a rollback to the source version can be performed. If other values are used, the annotation is ignored and the request traffic is allocated according to other rules based on the priority.

• nginx.ingress.kubernetes.io/canary-by-header-value

This rule must be used together with canary-by-header. You can customize the value of the request header, including but not limited to **always** or **never**. If the value of the request header matches the specified custom value, the request is forwarded to the corresponding backend service defined by the canary ingress. If the values do not match, the annotation is ignored and the request traffic is allocated according to other rules based on the priority.

• nginx.ingress.kubernetes.io/canary-by-header-pattern

This rule is similar to canary-by-header-value. The only difference is that this annotation uses a regular expression, not a fixed value, to match the value of the request header. If this annotation and canary-by-header-value exist at the same time, this one will be ignored.

• nginx.ingress.kubernetes.io/canary-by-cookie

Cookie-based traffic division, which is applicable to grayscale release. Similar to canary-by-header, this annotation is used for cookies. Only **always** and **never** are supported, and the value cannot be customized.

• nginx.ingress.kubernetes.io/canary-weight

Traffic is divided based on service weights, which is applicable to blue-green deployment. This annotation indicates the percentage of traffic allocated by the canary ingress. The value ranges from 0 to 100. For example, if the value is set to **100**, all traffic is forwarded to the backend service backing the canary ingress.

D NOTE

- The preceding annotation rules are evaluated based on the priority. The priority is as follows: canary-by-header -> canary-by-cookie -> canary-weight.
- When an ingress is marked as a canary ingress, all non-canary annotations except nginx.ingress.kubernetes.io/load-balance and nginx.ingress.kubernetes.io/upstream-hash-by are ignored.
- For more information, see Annotations.

Prerequisites

- To use Nginx Ingress to implement grayscale release of a cluster, install the nginx-ingress add-on as the Ingress Controller and expose a unified traffic entrance externally. For details, see Installing the Add-on.
- The Nginx image has been uploaded to SWR. The Nginx images have two versions. The welcome pages are **Old Nginx** and **New Nginx**.

Resource Creation

You can use YAML to deploy Deployments and Services in either of the following ways:

- On the **Create Deployment** page, click **Create YAML** on the right and edit the YAML file in the window.
- Save the sample YAML file in this section as a file and use kubectl to specify the YAML file. For example, run the **kubectl create -f xxx.yaml** command.

Step 1: Deploy Services of Two Versions

Two versions of Nginx are deployed in the cluster, and Nginx Ingress is used to provide layer-7 domain name access for external systems.

Step 1 Create a Deployment and Service for the first version. This section uses old-nginx as an example. Example YAML:

apiVersion: apps/v1 kind: Deployment metadata:

```
name: old-nginx
spec:
 replicas: 2
 selector:
  matchLabels:
   app: old-nginx
 template:
  metadata:
   labels:
     app: old-nginx
  spec:
    containers:
    - image: {your_repository}/nginx:old # The image used by the container is nginx:old.
     name: container-0
     resources:
      limits:
        cpu: 100m
        memory: 200Mi
      requests:
        cpu: 100m
        memory: 200Mi
    imagePullSecrets:
    - name: default-secret
---
apiVersion: v1
kind: Service
metadata:
name: old-nginx
spec:
 selector:
  app: old-nginx
 ports:

    name: service0

  targetPort: 80
  port: 8080
  protocol: TCP
 type: NodePort
```

Step 2 Create a Deployment and Service for the second version. This section uses newnginx as an example. Example YAML:

```
apiVersion: apps/v1
kind: Deployment
metadata:
name: new-nginx
spec:
 replicas: 2
 selector:
  matchLabels:
   app: new-nginx
 template:
  metadata:
   labels:
    app: new-nginx
  spec:
   containers:
   - image: {your_repository}/nginx:new # The image used by the container is nginx:new.
     name: container-0
     resources:
      limits:
       cpu: 100m
       memory: 200Mi
      requests:
       cpu: 100m
       memory: 200Mi
   imagePullSecrets:
   - name: default-secret
```

apiVersion: v1 kind: Service metadata: name: new-nginx spec: selector: app: new-nginx ports: - name: service0 targetPort: 80 port: 8080 protocol: TCP type: NodePort

You can log in to the CCE console to view the deployment status.

Step 3 Create an ingress to expose the service and point to the service of the old version. Example YAML:

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
 name: gray-release
 namespace: default
 annotations:
  kubernetes.io/elb.port: '80'
spec:
 rules:
  - host: www.example.com
   http:
     paths:
       - path: /
       backend:
         service:
          name: old-nginx
                              # Set the back-end service to old-nginx.
          port:
           number: 80
        property:
         ingress.beta.kubernetes.io/url-match-mode: STARTS_WITH
        pathType: ImplementationSpecific
 ingressClassName: nginx # Nginx ingress is used.
```

Step 4 Run the following command to verify the access:

```
curl -H "Host: www.example.com" http://<EXTERNAL_IP>
```

In the preceding command, <EXTERNAL_IP> indicates the external IP address of the Nginx ingress.

Expected outputs:

Old Nginx

----End

Step 2: Launch the New Version of the Service in Grayscale Release Mode

Set the traffic division policy for the service of the new version. CCE supports the following policies for grayscale release and blue-green deployment:

Header-based, cookie-based, and weight-based traffic division rules

Grayscale release can be implemented based on all these policies. Blue-green deployment can be implemented by adjusting the new service weight to 100%. For details, see the following examples.

Pay attention to the following:

- Only one canary ingress can be defined for the same service so that the backend service supports a maximum of two versions.
- Even if the traffic is completely switched to the canary ingress, the old version service must still exist. Otherwise, an error is reported.

• Header-based rules

In the following example, only the request whose header contains **Region** set to **bj** or **gz** can be forwarded to the service of the new version.

Create a canary ingress, set the backend service to the one of the new а. versions, and add annotations. apiVersion: networking.k8s.io/v1 kind: Inaress metadata: name: canary-ingress namespace: default annotations: nginx.ingress.kubernetes.io/canary: "true" # Enable canary. nginx.ingress.kubernetes.io/canary-by-header: "Region" nginx.ingress.kubernetes.io/canary-by-header-pattern: "bj|gz" # Requests whose header contains **Region** with the value **bj** or **gz** are forwarded to the canary ingress. kubernetes.io/elb.port: '80' spec: rules: - host: www.example.com http: paths: - path: / backend: service: # Set the back-end service to **new-nginx**. name: new-nginx port: number: 80 property: ingress.beta.kubernetes.io/url-match-mode: STARTS_WITH pathType: ImplementationSpecific ingressClassName: nginx # Nginx ingress is used. b. Run the following command to test the access: \$ curl -H "Host: www.example.com" -H "Region: bj" http://<EXTERNAL_IP> New Nginx \$ curl -H "Host: www.example.com" -H "Region: sh" http://<EXTERNAL_IP> Old Nginx

\$ curl -H "Host: www.example.com" -H "Region: gz" http://<EXTERNAL_IP>
New Nginx
\$ curl -H "Host: www.example.com" http://<EXTERNAL_IP>

Old Nginx

In the preceding command, <EXTERNAL_IP> indicates the external IP address of the Nginx ingress.

Only requests whose header contains **Region** with the value **bj** or **gz** are responded by the service of the new version.

• Cookie-based rules

In the following example, only the request whose cookie contains **user_from_bj** can be forwarded to the service of the new version.

a. Create a canary ingress, set the backend service to the one of the new versions, and add annotations.

D NOTE

	If you have created a canary ingress in the preceding steps, delete it and then perform this step to create a canary ingress.
i	apiVersion: networking.k8s.io/v1 kind: Ingress metadata: name: canary-ingress namespace: default annotations: nginx.ingress.kubernetes.io/canary: "true"
0	contains user_from_bj are forwarded to the canary ingress.
	spec
	rules:
	- host: www.example.com
	http:
	paths:
	- path: /
	backend:
	service:
	name: new-nginx # Set the back-end service to new-nginx .
	port:
	number: 80
	property:
	ingress.beta.kubernetes.io/url-match-mode: STARTS_WITH
	path lype: implementationSpecific
	ingressciassivame: nginx # inginx ingress is used.

b. Run the following command to test the access:

```
$ curl -s -H "Host: www.example.com" --cookie "user_from_bj=always" http://
<EXTERNAL_IP>
New Nginx
$ curl -s -H "Host: www.example.com" --cookie "user_from_gz=always" http://
<EXTERNAL_IP>
Old Nginx
$ curl -s -H "Host: www.example.com" http://<EXTERNAL_IP>
Old Nginx
```

In the preceding command, <EXTERNAL_IP> indicates the external IP address of the Nginx ingress.

Only requests whose cookie contains **user_from_bj** with the value **always** are responded by the service of the new version.

• Service weight-based rules

Example 1: Only 20% of the traffic is allowed to be forwarded to the service of the new version to implement grayscale release.

a. Create a canary ingress and add annotations to import 20% of the traffic to the backend service of the new version.

NOTE

If you have created a canary ingress in the preceding steps, delete it and then perform this step to create a canary ingress.

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
name: canary-ingress
namespace: default
annotations:
nginx.ingress.kubernetes.io/canary: "true"  # Enable canary.
nginx.ingress.kubernetes.io/canary-weight: "20"  # Forward 20% of the traffic to the canary
ingress.
kubernetes.io/elb.port: '80'
```

spec:

b.

```
rules:
  - host: www.example.com
   http:
    paths:
      path: /
       backend:
        service:
         name: new-nginx
                            # Set the back-end service to new-nginx.
         port:
          number: 80
       property:
        ingress.beta.kubernetes.io/url-match-mode: STARTS WITH
       pathType: ImplementationSpecific
 ingressClassName: nginx # Nginx ingress is used.
Run the following command to test the access:
$ for i in {1..20}; do curl -H "Host: www.example.com" http://<EXTERNAL_IP>; done;
Old Nginx
Old Nainx
Old Nginx
New Nginx
Old Nginx
New Nginx
Old Nginx
New Nginx
Old Nginx
Old Nginx
Old Nginx
Old Nginx
Old Nginx
New Nginx
Old Nginx
Old Nginx
Old Nginx
Old Nginx
Old Nginx
Old Nginx
In the preceding command, <EXTERNAL_IP> indicates the external IP
address of the Nginx ingress.
```

It can be seen that there is a 4/20 probability that the service of the new version responds, which complies with the setting of the service weight of 20%.

D NOTE

After traffic is divided based on the weight (20%), the probability of accessing the new version is close to 20%. The traffic ratio may fluctuate within a small range, which is normal.

Example 2: Allow all traffic to be forwarded to the service of the new version to implement blue-green deployment.

a. Create a canary ingress and add annotations to import 100% of the traffic to the backend service of the new version.

NOTE

If you have created a canary ingress in the preceding steps, delete it and then perform this step to create a canary ingress.

apiVersion: networking.k8s.io/v1 kind: Ingress metadata: name: canary-ingress namespace: default annotations:

```
nginx.ingress.kubernetes.io/canary: "true" # Enable canary.
  nginx.ingress.kubernetes.io/canary-weight: "100" # All traffic is forwarded to the canary
ingress.
  kubernetes.io/elb.port: '80'
spec:
 rules:
  - host: www.example.com
   http:
     paths:
       - path: /
       backend:
         service:
          name: new-nginx
                               # Set the back-end service to new-nginx.
          port:
           number: 80
       property:
        ingress.beta.kubernetes.io/url-match-mode: STARTS_WITH
       pathType: ImplementationSpecific
 ingressClassName: nginx # Nginx ingress is used.
```

b. Run the following command to test the access:

\$ for i in {1..10}; do curl -H "Host: www.example.com" http://<EXTERNAL_IP>; done; New Nginx New Nginx

In the preceding command, <EXTERNAL_IP> indicates the external IP address of the Nginx ingress.

All access requests are responded by the service of the new version, and the blue-green deployment is successfully implemented.

15 Batch Computing

15.1 Deploying and Using Kubeflow in a CCE Cluster

15.1.1 Deploying Kubeflow

Background

Building an end-to-end AI computing platform based on Kubernetes is complex. More than a dozen of phases is required. Apart from the familiar model training phase, the process also includes data collection, preprocessing, resource management, feature extraction, data verification, model management, model release, and monitoring. If AI algorithm engineers want to run a model training task, they have to build an entire AI computing platform first. Imagine how timeand labor-consuming that is and how much knowledge and experience it requires.



Figure 15-1 Phrases for training a model

Kubeflow was released in 2017, which is built on containers and Kubernetes. It aims to provide data scientists, machine learning engineers, and system O&M personnel with a platform for agile deployment, development, training, release, and management of machine learning services. It leverages the advantages of cloud native technologies to enable users to quickly and easily deploy, use, and manage the most popular machine learning software.

Kubeflow 1.0 is now available, providing capabilities in development, building, training, and deployment that cover the entire process of machine learning and deep learning for enterprise users.



The following shows an example.

With Kubeflow 1.0, you first develop a model using Jupyter, and then set up containers using tools such as Fairing (SDK). Next, you create Kubernetes resources to train the model. After the training is complete, you create and deploy servers for inference using KFServing. This is how you use Kubeflow to establish an end-to-end agile process of a machine learning task. This process can be fully automated using pipelines, which help achieve DevOps in the AI field.

Prerequisites

- A cluster named clusterA has been created on CCE. The cluster has an available GPU node that has two or more GPUs.
- EIPs have been bound to the nodes, and the kubectl command line tool has been configured. For details, see **Connecting to a Cluster Using kubectl**.

Installing Kustomize

Kustomize is an open-source tool used to manage the configuration of applications running in Kubernetes clusters. It allows you to modify application configuration. Starting with Kubeflow 1.3, all components should be deployed only using Kustomize.

Step 1 Use the official script to install Kustomize. Kubeflow is incompatible with earlier versions of Kustomize. Therefore, only Kustomize 5 and later versions are supported. In this example, Kubeflow 5.1.0 is used.

curl -o install_kustomize.sh "https://raw.githubusercontent.com/kubernetes-sigs/kustomize/master/hack/ install_kustomize.sh" sh install kustomize.sh 5.1.0 . The installation may take 3 to 5 minutes, and the information similar to the following will be displayed: v5.1.0

kustomize installed to /root/kubeflow/./kustomize

Step 2 Move kustomize to the **/bin** directory so that the **kustomize** command can be used globally.

cp kustomize /bin/

----End

Installing Kubeflow

Perform the steps in this section to install all official Kubeflow components. After the installation, you can access the Kubeflow central dashboard. For details, see **Connecting to Kubeflow**.

- Step 1 Install Kubeflow 1.7.0. wget https://github.com/kubeflow/manifests/archive/refs/tags/v1.7.0.zip unzip v1.7.0.zip
- **Step 2** Use Kustomize to create a YAML file for deploying Kubeflow. cd ./manifests-1.7.0/ kustomize build example -o example.yaml
- Step 3 Configure storage resources required by Kubeflow.
 - katib-mysql
 - mysql-pv-claim
 - minio-pv-claim
 - authservice-pvc

Some storage resources need to be configured during the installation. The storage configuration in the official example cannot take effect in CCE. This may result in the preceding PVC fail to be created. Therefore, create a PVC with the same name in the cluster in advance. In this example, the EVS disk is used. You can change the storage type as required.

Create the **pvc.yaml** file. The following is an example:

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
name: katib-mysql
 namespace: kubeflow
annotations
 everest.io/disk-volume-type: SAS # EVS disk type
labels:
  to be deployed
  failure-domain.beta.kubernetes.io/zone: <your_zone>
                                              # AZ of the node where the application is to be
deployed
spec:
 accessModes:
 - ReadWriteOnce
 resources:
  requests:
   storage: 10Gi
storageClassName: csi-disk
apiVersion: v1
kind: PersistentVolumeClaim
```

metadata: name: mysql-pv-claim namespace: kubeflow annotations: everest.io/disk-volume-type: SAS # EVS disk type labels: failure-domain.beta.kubernetes.io/region: <your_region> # Region of the node where the application is to be deployed failure-domain.beta.kubernetes.io/zone: <your_zone> # AZ of the node where the application is to be deployed spec: accessModes: - ReadWriteOnce resources: requests: storage: 20Gi storageClassName: csi-disk apiVersion: v1 kind: PersistentVolumeClaim metadata: name: minio-pvc namespace: kubeflow annotations: everest.io/disk-volume-type: SAS # EVS disk type labels: failure-domain.beta.kubernetes.io/region:
Region of the node where the application is to be deployed failure-domain.beta.kubernetes.io/zone: <your_zone> # AZ of the node where the application is to be deployed spec: accessModes: - ReadWriteOnce resources: requests: storage: 20Gi storageClassName: csi-disk apiVersion: v1 kind: PersistentVolumeClaim metadata: name: authservice-pvc namespace: istio-system annotations: everest.io/disk-volume-type: SAS # EVS disk type labels: failure-domain.beta.kubernetes.io/region: //region # Region of the node where the application is to be deployed failure-domain.beta.kubernetes.io/zone: <your_zone> # AZ of the node where the application is to be deployed spec: accessModes: - ReadWriteOnce resources: requests: storage: 10Gi storageClassName: csi-disk

Create a PVC.

kubectl apply -f pvc.yaml

Step 4 Create related resources.

kubectl apply -f example.yaml

NOTE

Official images may fail to be pulled due to network problems, and the ImagePullBackOff or FailedPullImage error may occur in the workload. In this case, add a proper image proxy.

Step 5 Check whether pods in all namespaces are running.

kubectl get pod -A

If an unexpected problem occurs during resource creation, rectify it by referring to **Common Issues**.

----End

Common Issues

 In some scenarios where CRD resources do not exist, the following information is displayed: error: resource mapping not found for name: "<RESOURCE_NAME>" namespace: "<SOME_NAMESPACE>" from "STDIN": no matches for kind "<CRD_NAME>" in version "<CRD_FULL_NAME>" ensure CRDs are installed first

Solution:

This is because kustomization creates CRs ahead of CRDs. If you encounter this error message, create the resource again.

• When a workload is created, an error message is displayed, indicating that there are too many pods on the node. The error message is displayed as follows:

0/x nodes are available: x Too many pods.

Solution:

This message indicates that the number of schedulable pods on the node exceeds the node's upper limit. To solve this problem, increase the number of nodes.

• The **training-operator** workload cannot run properly. The error message in the log is displayed as follows:

Waited for 1.039518449s due to client-side throttling, not priority and fairness, request: GET:https:// 10.247.0.1:443/apis/xxx/xx?timeout=32s

Solution:

Run the following command to check the statuses of the unavailable APIServices in the cluster:

kubectl get apiservice

If there is no APIService in the **FALSE** state, the **training-operator** workload will run 1 to 2 minutes later.

15.1.2 Training a TensorFlow Model

After Kubeflow is deployed, it is easy to use the ps-worker mode to train TensorFlow models. This section describes an official TensorFlow training example provided by Kubeflow. For details, see **TensorFlow Training (TFJob)**.

Running the Mnist Example

Step 1 Deploy the TFJob resource to start training.

Create the **tf-mnist.yaml** file. The following is an example: apiVersion: "kubeflow.org/v1" kind: TFJob metadata: name: tfjob-simple namespace: kubeflow spec: tfReplicaSpecs: Worker: replicas: 2 restartPolicy: OnFailure template: spec: containers: - name: tensorflow image: kubeflow/tf-mnist-with-summaries:latest command: - "python" - "/var/tf_mnist/mnist_with_summaries.py"

Step 2 Create the TFJob.

kubectl apply -f tf-mnist.yaml

Step 3 View the logs after the worker running is complete. kubectl -n kubeflow logs tfjob-simple-worker-0

Information similar to the following is displayed:

... Accuracy at step 900: 0.964 Accuracy at step 910: 0.9653 Accuracy at step 920: 0.9665 Accuracy at step 930: 0.9681 Accuracy at step 940: 0.9664 Accuracy at step 950: 0.9667 Accuracy at step 960: 0.9694 Accuracy at step 970: 0.9683 Accuracy at step 980: 0.9687 Accuracy at step 990: 0.966 Adding run metadata for 999

Step 4 Delete the TFJob.

kubectl delete -f tf-mnist.yaml

----End

Using a GPU

The training can be performed in the GPU scenario. In this scenario, the cluster must contain GPU nodes and proper drivers must be installed.

Step 1 Specify the GPU resources in the TFJob.

Create the **tf-gpu.yaml** file. The following is an example:

This example runs in the TensorFlow distributed architecture. The ResNet50 model in the convolutional neural network (CNN) is used to train randomly generated images. A total of **32** (**batch_size**) images are trained each time, and the images are trained **100** times in total. Additionally, the performance (**image/sec**) of each training is recorded.

```
apiVersion: "kubeflow.org/v1"
kind: "TFJob"
metadata:
name: "tf-smoke-gpu"
spec:
tfReplicaSpecs:
PS:
replicas: 1
template:
metadata:
creationTimestamp: null
```

spec: containers: - args: - python - tf_cnn_benchmarks.py - --batch_size=32 - --model=resnet50 - --variable_update=parameter_server - --flush_stdout=true - --num_gpus=1 - --local_parameter_device=cpu - --device=cpu - --data format=NHWC image: docker.io/kubeflow/tf-benchmarks-cpu:v20171202-bdab599-dirty-284af3 name: tensorflow ports: - containerPort: 2222 name: tfjob-port resources: limits: cpu: "1" workingDir: /opt/tf-benchmarks/scripts/tf_cnn_benchmarks restartPolicy: OnFailure Worker: replicas: 1 template: metadata: creationTimestamp: null spec: containers: - args: - python - tf_cnn_benchmarks.py - --batch_size=32 - --model=resnet50 - --variable_update=parameter_server - --flush_stdout=true - --num_gpus=1 - --local_parameter_device=cpu - --device=qpu - --data_format=NHWC image: docker.io/kubeflow/tf-benchmarks-gpu:v20171202-bdab599-dirty-284af3 name: tensorflow ports: containerPort: 2222 name: tfjob-port resources: limits: nvidia.com/gpu: 1 # Number of GPUs workingDir: /opt/tf-benchmarks/scripts/tf_cnn_benchmarks restartPolicy: OnFailure

Step 2 Create the TFJob.

kubectl apply -f tf-gpu.yaml

Step 3 After the worker runs the job (about 5 minutes if a GPU is used), run the following command to view the result. kubectl logs tf-smoke-gpu-worker-0

Information similar to the following is displayed:

```
...

INFO|2023-09-02T12:04:25|/opt/launcher.py|27| Running warm up

INFO|2023-09-02T12:08:55|/opt/launcher.py|27| Done warm up

INFO|2023-09-02T12:08:55|/opt/launcher.py|27| Step Img/sec loss

INFO|2023-09-02T12:08:56|/opt/launcher.py|27| 1 images/sec: 68.8 +/- 0.0 (jitter = 0.0) 8.777

INFO|2023-09-02T12:09:00|/opt/launcher.py|27| 10 images/sec: 70.4 +/- 0.4 (jitter = 1.8) 8.557

INFO|2023-09-02T12:09:04|/opt/launcher.py|27| 20 images/sec: 70.5 +/- 0.3 (jitter = 1.5) 8.090

INFO|2023-09-02T12:09:09|/opt/launcher.py|27| 30 images/sec: 70.3 +/- 0.3 (jitter = 1.6) 8.041
```

INFO 2023-09-02T12:09:13 /opt/launcher.py 27 40	images/sec: 70.1 +/- 0.2 (jitter = 1.7) 9.464
INFO 2023-09-02T12:09:18 /opt/launcher.py 27 50	images/sec: 70.1 +/- 0.2 (jitter = 1.6) 7.797
INFO 2023-09-02T12:09:23 /opt/launcher.py 27 60	images/sec: 70.1 +/- 0.2 (jitter = 1.6) 8.595
INFO 2023-09-02T12:09:27 /opt/launcher.py 27 70	images/sec: 70.0 +/- 0.2 (jitter = 1.7) 7.853
INFO 2023-09-02T12:09:32 /opt/launcher.py 27 80	images/sec: 69.9 +/- 0.2 (jitter = 1.7) 7.849
INFO 2023-09-02T12:09:36 /opt/launcher.py 27 90	images/sec: 69.8 +/- 0.2 (jitter = 1.7) 7.911
INFO 2023-09-02T12:09:41 /opt/launcher.py 27 100	images/sec: 69.7 +/- 0.1 (jitter = 1.7) 7.853
INFO 2023-09-02T12:09:41 /opt/launcher.py 27	
INFO 2023-09-02T12:09:41 /opt/launcher.py 27 total in	nages/sec: 69.68
INFO 2023-09-02T12:09:41 /opt/launcher.py 27	
INFO 2023-09-02T12:09:42 /opt/launcher.py 80 Finished	d: python tf_cnn_benchmarks.pybatch_size=32
model=resnet50variable_update=parameter_server	-flush_stdout=truenum_gpus=1
local_parameter_device=cpudevice=gpudata_forma	at=NHWCjob_name=workerps_hosts=tf-smoke-
gpu-ps-0.default.svc:2222worker_hosts=tf-smoke-gpu	u-worker-0.default.svc:2222task_index=0
INFO 2023-09-02T12:09:42 /opt/launcher.py 84 Comma	and ran successfully sleep for ever.

The training performance of a single GPU is 69.68 images per second.

----End

15.1.3 Using Kubeflow and Volcano to Train an AI Model

Kubernetes has become the de facto standard for cloud native application orchestration and management. An increasing number of applications are migrated to Kubernetes. AI and machine learning inherently involve a large number of computing-intensive tasks. Kubernetes is a preferential tool for developers building AI platforms because of its excellent capabilities in resource management, application orchestration, and O&M monitoring.

Problems of the Default Kubernetes Scheduler in Batch Computing

Kubeflow uses the default scheduler of Kubernetes, which was initially designed for long-term running services. Its scheduling capability is inadequate for tasks that involve batch computing and elastic scheduling in AI and big data scenarios. The main constraints are as follows:

Resource preemption

A TensorFlow job consists of parameter servers (PS) and workers that need to collaborate to complete the training task. If only one pod is running, the job cannot execute properly. The default Kubernetes scheduler assigns pods individually and does not recognize the PS-worker dependency in a TFJob of a Kubeflow job. This scheduling approach can lead to resource allocation issues, especially when a cluster is under heavy load. In the scenario illustrated in the figure, there are four GPUs available in the cluster. TFJob1 and TFJob2 are configured with four workers each, but they are only allocated two GPUs each. However, both TFJob1 and TFJob2 actually need four GPUs each. This can result in a deadlock situation where both jobs are waiting for resources to be released, leading to inefficient GPU resource utilization.



Lack of affinity-based scheduling

In distributed training, data exchange between PS and workers is common and crucial for training efficiency. The bandwidth between PS and workers directly impacts training performance. However, the default Kubernetes scheduler does not account for the logical relationship between PS and workers, leading to random scheduling. For instance, when running two TFJobs, each with one PS and two workers, different scheduling outcomes can occur. The best scenario is depicted in case (c), where the PS and workers can leverage the local network for faster data transmission, enhancing training speed.



Volcano, a Perfect Batch Scheduling System for Accelerating AI Computing

Volcano is an enhanced batch scheduling system for high-performance computing workloads running on Kubernetes. It complements Kubernetes in machine learning, deep learning, HPC, and big data computing scenarios, providing capabilities such as gang scheduling, computing task queue management, tasktopology, and GPU affinity scheduling. In addition, Volcano enhances batch task creation and lifecycle management, fair-share, binpack, and other Kubernetesnative capabilities. It fully addresses the constraints of Kubeflow in distributed training mentioned above.



For more information about Volcano, visit https://github.com/volcano-sh/volcano.

Using Volcano in Huawei Cloud

The convergence of Kubeflow and Volcano, two open-source projects, greatly simplifies and accelerates AI computing workloads running on Kubernetes. The two projects have been recognized by an increasing number of players in the field and applied in production environments. Volcano is used in Huawei Cloud CCE, CCI, and Kubernetes-Native Batch Computing Solution. Volcano will continue to iterate with optimized algorithms, enhanced capabilities such as intelligent scheduling, and new inference features such as GPU Share, to further improve the efficiency of Kubeflow batch training and inference.

Implementing Typical Distributed AI Training Jobs

This section describes how to perform distributed training of a digital image classification model using the MNIST dataset based on Kubeflow and Volcano.

- **Step 1** Log in to the CCE console and click the cluster name to access the cluster console.
- Step 2 Deploy volcano on the cluster.

In the navigation pane, choose **Add-ons**. In the right pane, find **Volcano Scheduler** and click **Install**. In the window that slides out from the right, configure the specifications and click **Install**.

- Step 3 Deploy the MNIST dataset.
 - Download kubeflow/examples to the local host and select a guide based on the environment. The command is as follows: yum install git git clone https://github.com/kubeflow/examples.git

 Install Python3. For details, see Getting and installing the latest version of Python. wget https://www.python.org/ftp/python/3.6.8/Python-3.6.8.tgz tar -zxvf Python-3.6.8.tgz cd Python-3.6.8 ./configure make make install After the installation, run the following commands to check whether the installation is successful:

python3 -V pip3 -V

- 3. Install and start Jupyter Notebook. pip3 install jupyter notebook jupyter notebook --allow-root
- 4. Configure an SSH tunnel on PuTTY and remotely connect to the notebook.
- 5. After the connection is successful, enter **localhost:8000** in the address box of a browser to log in to the notebook.

Ç jupyter		Quit	Logout
Files Running Clusters			
Select items to perform actions on them.			
□ 0 👻 🖿 / go / src / kubeflow / examples / mnist	Name 🔶	Last Modified	File size
C data			
C web-ui			
E mnist_aws.ipynb			31.8 kB
mist_azure.ipynb			30.7 kB
🔲 🖉 mnist_gcp.ipynb			84 kB
🔲 🖉 mnist_ibm.ipynb			29.5 kB
mist_vanilla_k8s.ipynb			74 kB
Dockerfile.model			323 B
k8s_util.py			6.09 kB
Makefile			2.17 kB
C model.py			8.68 kB
C notebook_setup.py			2.29 kB
README.md			5.63 kB
requirements.txt			184 B

6. Create a distributed training job as prompted by Jupyter. Set the value of **schedulerName** to **volcano** to enable volcano.

kind: TFJob
metadata:
name: {train_name}
spec:
schedulerName: volcano
tfReplicaSpecs:
Ps:
replicas: {num_ps}
template:
metadata:
annotations:
sidecar.istio.io/inject: "false"
spec:
serviceAccount: default-editor
containers:
- name: tensorflow
command:
env:
image: {image}
workingDir: /opt
restartPolicy: OnFailure
Worker:
replicas: 1
template:
metadata:
annotations:
sidecar.istio.io/inject: "false"
spec:
serviceAccount: default-editor
containers:

- name: tensorflow
command:
env:
image: {image}
workingDir: /opt
restartPolicy: OnFailure

Step 4 Submit the job and start the training.

kubectl apply -f mnist.yaml

root@ecs-1953:~/tmp# kubectl get po						
NAME	RĒADY	STATUS	RESTARTS	AGE		
tensorflow-mnist-ps-0	1/1	Running	0	5s		
tensorflow-mnist-worker-0	1/1	Running	0	5s		

After the training job is complete, you can query the training results on the Kubeflow UI. This is how you run a simple distributed training job using Kubeflow and Volcano. Kubeflow simplifies TensorFlow job configuration. Volcano, with simply one more line of configuration, saves you significant time and effort in large-scale distributed training by providing capabilities such as gang scheduling and task topology to eliminate deadlocks and achieve affinity scheduling.

----End

15.2 Deploying and Using Caffe in a CCE Cluster

15.2.1 Prerequisites

This section provides an example for running Caffe on CCE to classify an image. For more information, see https://github.com/BVLC/caffe/blob/master/ examples/00-classification.ipynb.

Pre-configuring OBS Storage Data

Create an OBS bucket and ensure that the following folders have been created and required files have been uploaded to the specified paths using the OBS Browser.

The folder name can be in the format of *File path in the bucket/File name*. You can search for the file download addresses in the specified paths of the specified project in GitHub, as shown in **1** and **2**.

1. models/bvlc_reference_caffenet/bvlc_reference_caffenet.caffemodel

https://github.com/BVLC/caffe/tree/master/models/ bvlc_reference_caffenet

		canemodel_un	license
BAIR/BVLC CaffeNet by Model	vvlc_reference_caffenet.caffemodel	http://dl.caffe.berkeleyvision.org/bvlc_reference_caffenet.caffemodel	unrestricted

 models/bvlc_reference_caffenet/deploy.prototxt
 https://github.com/BVLC/caffe/tree/master/models/ bvlc_reference_caffenet

ξ ⁹ master → caffe / models / bvlc_reference_caffene	t/	Go to file
shelhamer BVLC -> BAIR 📖	44da39f on 15 Apr 2017	🕚 History
deploy.prototxt	[examples] switch examples + models to Input layers	5 years ago
🗅 readme.md	BVLC -> BAIR	4 years ago
solver.prototxt	Renaming CaffeNet model prototxts and unignoring models/*	7 years ago
🗋 train_val.prototxt	Set CaffeNet train_val test mirroring to false	6 years ago

python/caffe/imagenet/ilsvrc_2012_mean.npy https://github.com/BVLC/caffe/tree/master/python/caffe/imagenet

ł	master 👻	caffe / pytho	on / caffe / imagenet /		Go to file
۲	shelhamer	r image classifica	tion in python	2fc32d5 on 20 May 2014	🕑 History
			_		
D	ilsvrc_2012	2_mean.npy		make python wrapper mean match binaryproto dimensions	7 years ago

4. outputimg/

An empty folder **outputimg** is created to store output files.

5. examples/images/cat.jpg

https://github.com/BVLC/caffe/blob/master/examples/00classification.ipynb

Save the picture of the cat in the link.

6. data/ilsvrc12/*

https://github.com/BVLC/caffe/tree/master/data/ilsvrc12

Obtain and execute the **get_ilsvrc_aux.sh** script. The script downloads a compressed package and decompresses it. After the script is executed, upload all decompressed files to the directory.

7. caffeEx00.py

set up Python environment: numpy for numerical routines, and matplotlib for plotting
import numpy as np
import matplotlib as mpl
mpl.use('Agg')
import matplotlib.pyplot as plt
display plots in this notebook
#%matplotlib inline
set display defaults
plt rcParams['figure figsize'] = (10, 10)
large images

plt.rcParams['figure.figsize'] = (10, 10) # large images plt.rcParams['image.interpolation'] = 'nearest' # don't interpolate: show square pixels plt.rcParams['image.cmap'] = 'gray' # use grayscale output rather than a (potentially misleading) color heatmap

The caffe module needs to be on the Python path; # we'll add it here explicitly. import sys caffe_root = '/home/' # this file should be run from {caffe_root}/examples (otherwise change this line) sys.path.insert(0, caffe_root + 'python')

import caffe

If you get "No module named _caffe", either you have not built pycaffe or you have the wrong path.

import os

#else:

- # print 'Downloading pre-trained CaffeNet model...'
- # !../scripts/download_model_binary.py ../models/bvlc_reference_caffenet
```
caffe.set_mode_cpu()
model_def = caffe_root + 'models/bvlc_reference_caffenet/deploy.prototxt'
model_weights = caffe_root + 'models/bvlc_reference_caffenet/bvlc_reference_caffenet.caffemodel'
net = caffe.Net(model_def, # defines the structure of the model
          model_weights, # contains the trained weights
          caffe.TEST) # use test mode (e.g., don't perform dropout)
# load the mean ImageNet image (as distributed with Caffe) for subtraction
mu = np.load(caffe_root + 'python/caffe/imagenet/ilsvrc_2012_mean.npy')
mu = mu.mean(1).mean(1) # average over pixels to obtain the mean (BGR) pixel values
print 'mean-subtracted values:', zip('BGR', mu)
# create transformer for the input called 'data'
transformer = caffe.io.Transformer({'data': net.blobs['data'].data.shape})
transformer.set_transpose('data', (2,0,1)) # move image channels to outermost dimension
transformer.set mean('data', mu)
                                         # subtract the dataset-mean value in each channel
transformer.set_raw_scale('data', 255)
                                         # rescale from [0, 1] to [0, 255]
transformer.set_channel_swap('data', (2,1,0)) # swap channels from RGB to BGR
# set the size of the input (we can skip this if we're happy
# with the default; we can also change it later, e.g., for different batch sizes)
net.blobs['data'].reshape(50,
                                 # batch size
                 3, # 3-channel (BGR) images
                 227, 227) # image size is 227x227
image = caffe.io.load_image(caffe_root + 'examples/images/cat.jpg')
transformed_image = transformer.preprocess('data', image)
plt.imshow(image)
plt.savefig(caffe_root + 'outputimg/img1.png')
# copy the image data into the memory allocated for the net
net.blobs['data'].data[...] = transformed_image
### perform classification
output = net.forward()
output_prob = output['prob'][0] # the output probability vector for the first image in the batch
print 'predicted class is:', output_prob.argmax()
# load ImageNet labels
labels_file = caffe_root + 'data/ilsvrc12/synset_words.txt'
#if not os.path.exists(labels_file):
# !../data/ilsvrc12/get_ilsvrc_aux.sh
labels = np.loadtxt(labels_file, str, delimiter='\t')
print 'output label:', labels[output_prob.argmax()]
# sort top five predictions from softmax output
top_inds = output_prob.argsort()[::-1][:5] # reverse sort and take five largest items
print 'probabilities and labels:'
zip(output_prob[top_inds], labels[top_inds])
```

15.2.2 Preparing Resources

Adding a GPU Node to a Cluster

- **Step 1** Log in to the CCE console and click the name of the target cluster to access the cluster console.
- **Step 2** Install the GPU add-on.

- 1. In the navigation pane, choose **Add-ons**. Locate **gpu-beta** (or **gpu-device-plugin**) and click **Install**.
- 2. In the window that slides out from the right, configure the key parameters.
 - **NVIDIA Driver**: Enter the download link of the NVIDIA driver. Select a driver based on the graphics card model of the GPU node.

Retain the default values for other parameters. For details, see **gpu-beta**.

3. Click Install.

Step 3 Create a GPU node.

- 1. In the navigation pane, choose **Nodes**. Click **Create Node** in the upper right corner and configure the parameters.
- 2. Select **GPU-accelerated** for node specifications and configure other parameters as required. For details, see **Creating a Node**.
- 3. After the configuration, click **Next: Confirm**. On the page displayed, confirm the configuration and click **Submit**.

Step 4 View its status in the node list.

----End

Importing an OBS Volume

Go to the storage management page and import the OBS volume created in **Pre-configuring OBS Storage Data**.

15.2.3 Caffe Classification Example

This section uses the official Caffe classification example at https://github.com/ BVLC/caffe/blob/master/examples/00-classification.ipynb to illustrate how to run Caffe jobs on CCE.

Using CPUs

Create a job using the third-party **image bvlc/caffe:cpu**. Set the container specifications.

Image caffe	Change Image		
* Image Version	сри		
* Container Name	container-0		
Container Resources	CPU	Request 2 cores but is not allowed to use more CPU/n Limit 2 cores	A container is guaranteed to have as much CPU/memory as it requests, nemory than its limit.
	Memory	Request 5 GIB Limit 5 GIB	If the memory limit is exceeded, the container will be terminated.
	GPU GPU/Graphics Card	100 % GPU graphics card type of the node to Any GPU type nvidia-14	Percentage of GPU resources reserved for the container. o which jobs will be scheduled.
	Ascend 310 Quota	Use 1	The maximum number of Ascend 310 processors that can be used by a container

Add the startup command **python /home/caffeEx00.py**.

Lifecycle	Environment Variables Data Storage Log Policies						
Set commands for starting and running containers. <i>Set container lifecycle parameters Set Learn how</i> to set container start command							
Start Command							
Command	python /home/caffeEx00.py		Example	Binary Bash			
Args	→ Multi-Args per Input Box		Command	python /var/tf_mnist/mnist_with_summaries.py			
	Enter the arguments for the command.	Ū	Args	-log_dir=/trainlearning_rate=0.01batch_size=1			
	⊕ Add			50			

Mount the imported OBS volume.

Add Cloud Vo	lume				2
Туре	⊖ evs	⊖ sfs	OBS	SFS Turbo	
Allocation Mode	Manual	 Automatic 			
* Name	cce-obs-	▼ C Create	an OBS bucket and clic	k refresh.	
Sub-Type	Standard				
* Container Path	0			Permission	Operation
/home				Read/Write 💌	Delete
Add Container P	ath				
		ОК	Cancel		

Click **Create**. After the job execution is complete, go to the **outputimg** directory of the OBS volume to view the image used for inference.

Log in to the node added in Adding a GPU Node to a Cluster and run the docker logs {*Container ID*} command to view the classification result. The result is displayed as tabby cat.

I1119 09:42:08.196944 1 upgrade_proto.cpp:44] Attempting to upgrade input file specified using deprecated trans
formation parameters: /home/models/bvlc_reference_caffenet/bvlc_reference_caffenet.caffemodel
I1119 09:42:08.197005 1 upgrade_proto.cpp:47] Successfully upgraded file specified using deprecated data transf
ormation parameters.
W1119 09:42:08.197010 1 upgrade_proto.cpp:49] Note that future Caffe releases will only support transform_param
messages for transformation fields.
I1119 09:42:08.197012 1 upgrade_proto.cpp:53] Attempting to upgrade input file specified using deprecated V1Lay
erParameter: /home/models/bvlc_reference_caffenet/bvlc_reference_caffenet.caffemodel
I1119 09:42:08.432265 1 upgrade_proto.cpp:61] Successfully upgraded file specified using deprecated V1LayerPara
meter
I1119 09:42:08.494707 1 net.cpp:744] Ignoring source layer loss
/usr/local/lib/python2.7/dist-packages/skimage/transform/_warps.py:84: UserWarning: The default mode, 'constant', w
ill be changed to 'reflect' in skimage 0.15.
warn("The default mode, 'constant', will be changed to 'reflect' in "
mean-subtracted values: [('B', 104.0069879317889), ('G', 116.66876761696767), ('R', 122.6789143406786)]
predicted class is: 281
output label: n02123045 tabby, tabby cat
probabilities and labels:

Using GPUs

Create a job using the third-party **bvlc/caffe:gpu**. Set the container specifications.

Image caffe	Change Image		
* Image Version	gpu	•	
* Container Name	container-0		
Container Resources	CPU	Request 2 cores but is not allowed to use more CPU/m Limit 2 cores	A container is guaranteed to have as much CPU/memory as it requests, nemory than its limit.
	Memory	Request 5 GiB Imit 5 GiB	If the memory limit is exceeded, the container will be terminated.
	GPU GPU/Graphics Card	100 % GPU graphics card type of the node to Any GPU type nvidia-t4	Percentage of GPU resources reserved for the container.
	Ascend 310 Quota	Use 1	The maximum number of Ascend 310 processors that can be used by a container

Add the startup command **python /home/caffeEx00_GPU.py**.

Lifecycle	Environment Variables Data Storage Log Policies				
Set commands f	or starting and running containers. \mathscr{O} Learn how to set container lifecycle paran	neters 🔗 Lean	m how to set containe	er start com	mand
Start Command					
Command	python /home/caffeEx00_GPU.py		Example Command	Binary	Bash
1095	- Mutu-Args per Input Box				· · · = · · = = · · · · ·
	Enter the arguments for the command.	ច	Args	🗖 log	g_dir=/trainlearning_rate=0.01batch_size=1
	(➔ Add			50	

Mount the imported OBS volume.

Add Cloud Vol	lume					×
Туре	○ EVS	⊖ SFS	OBS	O SFS Turbo		
Allocation Mode	 Manual 	 Automatic 				
* Name	cce-obs-	C Create a	n OBS bucket and click	refresh.		
Sub-Type	Standard					
* Container Path	0			Permission	Operation	
/home				Read/Write 🔻	Delete	
+ Add Container Pa	ath					
		ОК	Cancel			

Click **Create**. After the job execution is complete, go to the **outputimg** directory of the OBS volume to view the image used for inference.

Log in to the node added in Adding a GPU Node to a Cluster and run the docker logs {*Container ID*} command to view the classification result. The result is displayed as tabby cat.

1 upgrade_proto.cpp:44] Attempting to upgrade input file specified using deprecated trans	
ormation parameters: /home/models/bvlc_reference_caffenet/bvlc_reference_caffenet.caffemodel	
1119 09:42:08.197005 1 upgrade_proto.cpp:47] Successfully upgraded file specified using deprecated data transf	
rmation parameters.	
1119 09:42:08.197010 1 upgrade_proto.cpp:49] Note that future Caffe releases will only support transform_param	
nessages for transformation fields.	
1119 09:42:08.197012 1 upgrade_proto.cpp:53] Attempting to upgrade input file specified using deprecated V1Lay	
Parameter: /home/models/bvlc_reference_caffenet/bvlc_reference_caffenet.caffemodel	
1119 09:42:08.432265 1 upgrade_proto.cpp:61] Successfully upgraded file specified using deprecated V1LayerPara	
eter	
1119 09:42:08.494707 1 net.cpp:744] Ignoring source layer loss	
ısr/local/lib/python2.7/dist-packages/skimage/transform/_warps.py:84: UserWarning: The default mode, 'constant', س	
ll be changed to 'reflect' in skimage 0.15.	
warn("The default mode, 'constant', will be changed to 'reflect' in "	
ean-subtracted values: [('B', 104.0069879317889), ('G', 116.66876761696767), ('R', 122.6789143406786)]	
redicted class is: 281	
utput label: n02123045 tabby, tabby cat	Γ.
cobabilities and labels:	1

15.3 Deploying and Using TensorFlow in a CCE Cluster

Preparing Resources

- Create a CCE cluster and GPU nodes, and use the gpu-beta add-on to install the graphics card driver.
- Add an object storage volume to the cluster.

Pre-configuring Data

Download data from https://github.com/zalandoresearch/fashion-mnist.

Get the Data

Many ML libraries already include Fashion-MNIST data/API, give it a try!

You can use direct links to download the dataset. The data is stored in the same format as the original MNIST data.

Name	Content	Examples	Size	Link	MD5 Checksum
train-images-idx3- ubyte.gz	training set images	60,000	26 MBytes	Download	8d4fb7e6c68d591d4c3dfef9ec88bf0d
train-labels-idx1- ubyte.gz	training set labels	60,000	29 KBytes	Download	25c81989df183df01b3e8a0aad5dffbe
t10k-images-idx3- ubyte.gz	test set images	10,000	4.3 MBytes	Download	bef4ecab320f06d8554ea6380940ec79
t10k-labels-idx1- ubyte.gz	test set labels	10,000	5.1 KBytes	Download	bb300cfdad3c16e7a12a480ee83cd310

Obtain the TensorFlow machine learning (ML) example and modify it based on your requirements.

basicClass.py

TensorFlow and tf.keras import tensorflow as tf from tensorflow import keras

Helper libraries import numpy as np import gzip from tensorflow.python.keras.utils import get_file import matplotlib as mpl mpl.use('Agg') import matplotlib.pyplot as plt

print(tf.__version__)

#fashion_mnist = keras.datasets.fashion_mnist
#(train_images, train_labels), (test_images, test_labels) = fashion_mnist.load_data()

```
def load_data():
  base = "file:///home/data/"
  files = [
      'train-labels-idx1-ubyte.gz', 'train-images-idx3-ubyte.gz',
't10k-labels-idx1-ubyte.gz', 't10k-images-idx3-ubyte.gz'
  ]
  paths = []
   for fname in files:
     paths.append(get_file(fname, origin=base + fname))
  with gzip.open(paths[0], 'rb') as lbpath:
     y_train = np.frombuffer(lbpath.read(), np.uint8, offset=8)
  with gzip.open(paths[1], 'rb') as imgpath:
     x_train = np.frombuffer(
        imgpath.read(), np.uint8, offset=16).reshape(len(y_train), 28, 28)
  with gzip.open(paths[2], 'rb') as lbpath:
     y_test = np.frombuffer(lbpath.read(), np.uint8, offset=8)
  with gzip.open(paths[3], 'rb') as imgpath:
     x_test = np.frombuffer(
        imgpath.read(), np.uint8, offset=16).reshape(len(y_test), 28, 28)
  return (x_train, y_train), (x_test, y_test)
```

(train_images, train_labels), (test_images, test_labels) = load_data()

```
class_names = ['T-shirt/top', 'Trouser', 'Pullover', 'Dress', 'Coat',
          'Sandal', 'Shirt', 'Sneaker', 'Bag', 'Ankle boot']
plt.figure()
plt.imshow(train_images[0])
plt.colorbar()
plt.grid(False)
plt.savefig('/home/img/basicimg1.png')
train_images = train_images / 255.0
test_images = test_images / 255.0
plt.figure(figsize=(10,10))
for i in range(25):
  plt.subplot(5,5,i+1)
  plt.xticks([])
  plt.yticks([])
  plt.grid(False)
   plt.imshow(train_images[i], cmap=plt.cm.binary)
  plt.xlabel(class names[train labels[i]])
plt.savefig('/home/img/basicimg2.png')
model = keras.Sequential([
   keras.layers.Flatten(input_shape=(28, 28)),
  keras.layers.Dense(128, activation=tf.nn.relu),
   keras.layers.Dense(10, activation=tf.nn.softmax)
])
model.compile(optimizer=tf.train.AdamOptimizer(),
          loss='sparse_categorical_crossentropy',
          metrics=['accuracy'])
model.fit(train_images, train_labels, epochs=5)
test_loss, test_acc = model.evaluate(test_images, test_labels)
print('Test accuracy:', test_acc)
predictions = model.predict(test_images)
def plot_image(i, predictions_array, true_label, img):
 predictions_array, true_label, img = predictions_array[i], true_label[i], img[i]
 plt.grid(False)
 plt.xticks([])
 plt.yticks([])
 plt.imshow(img, cmap=plt.cm.binary)
 predicted_label = np.argmax(predictions_array)
 if predicted_label == true_label:
  color = 'blue'
 else:
  color = 'red'
 plt.xlabel("{} {:2.0f}% ({})".format(class_names[predicted_label],
                      100*np.max(predictions_array),
                      class_names[true_label]),
                      color=color)
def plot_value_array(i, predictions_array, true_label):
 predictions_array, true_label = predictions_array[i], true_label[i]
 plt.grid(False)
 plt.xticks([])
 plt.yticks([])
 thisplot = plt.bar(range(10), predictions_array, color="#777777")
 plt.ylim([0, 1])
 predicted_label = np.argmax(predictions_array)
```

```
thisplot[predicted_label].set_color('red')
 thisplot[true_label].set_color('blue')
i = 0
plt.figure(figsize=(6,3))
plt.subplot(1,2,1)
plot_image(i, predictions, test_labels, test_images)
plt.subplot(1,2,2)
plot_value_array(i, predictions, test_labels)
plt.savefig('/home/img/basicimg3.png')
i = 12
plt.figure(figsize=(6,3))
plt.subplot(1,2,1)
plot_image(i, predictions, test_labels, test_images)
plt.subplot(1,2,2)
plot_value_array(i, predictions, test_labels)
plt.savefig('/home/img/basicimg4.png')
# Plot the first X test images, their predicted label, and the true label
# Color correct predictions in blue, incorrect predictions in red
num_rows = 5
num_cols = 3
num_images = num_rows*num_cols
plt.figure(figsize=(2*2*num_cols, 2*num_rows))
for i in range(num_images):
 plt.subplot(num_rows, 2*num_cols, 2*i+1)
 plot_image(i, predictions, test_labels, test_images)
 plt.subplot(num_rows, 2*num_cols, 2*i+2)
 plot_value_array(i, predictions, test_labels)
plt.savefig('/home/img/basicimg5.png')
```

Go to the OBS bucket page, create the **data** and **img** folders, and upload **basicClass.py**.

Objects Deleted Objects Fragments							
Objects are basic units of data storage. In OBS, files and folders are treated as objects. Any file type can be uploaded and managed in a bucket. Learn more							
Upload Object Create Folder Restore	Delete Change Storage C	Class					
Object Name ↓Ξ	Storage Class ↓=	Size ↓≡	Encrypted				
📄 🖻 img							
🗌 🖻 data							
basicClass.py	Standard	1.000	No				

Go to the **data** folder and upload the four .gz files downloaded from GitHub.

ML Example

In this section, the ML example from the TensorFlow official website is used. For details, see https://www.tensorflow.org/tutorials/keras/classification?hl=en-us.

Create a job using the third-party **tensorflow/tensorflow:1.15.5-gpu**. Set the container specifications.

Image Name	tensorflow	Change Image
* Image Version	1.15.5-gpu	•
* Container Name	container-0	
Privileged Container	The privileged cont	ainer is given access to all devices on the host node.
Container Resources	CPU	Request 2 cores
		A container is guaranteed to have as much CPU/memory as it requests,
		but is not allowed to use more CPU/memory than its limit.
		Limit 2 cores
	Memory	Request MIB
		Limit 4 MiB
		If the memory limit is exceeded, the container will be terminated.
	GPU	☑ Use 100 %
		Percentage of GPU resources reserved for the container.
	GPU/Graphics Card	Workload instances will be scheduled to the nodes that have the GPU
		type you specify.
		Any GPU type
		nvidia-p4

Add **pip install matplotlib;python /home/basicClass.py** in the **Start Command** area.

	^	Lifecycle	Set commands for starting and running containers. \mathscr{P} Learn how to set contain	ier lifecycle p	oarameters 🤗 Learn
:	Start	Command			
	C	Command	/bin/bash		Example
	A	Args	⇔ Multi-Args per Input Box		Command
			pip install matplotlib; python /home/basicClass.py	Ū	Args
			(+) Add		

Mount the created OBS volume.

Add Cloud Vol	lume					×
Туре	EVS	⊖ sfs	OBS	O SFS Turbo		
Allocation Mode	 Manual 	 Automatic 				
* Name	cce-obs-tensorflow	▼ C Create a	n OBS bucket and click	refresh.		
Sub-Type	Standard					
* Container Path	0			Permission	Operation	
/home				Read/Write 🔻	Delete	
Add Container Pa	ath					
		ОК	Cancel			

Click **Create**. Wait until the job execution is complete. On the OBS page, you can view the execution results that are shown as images.

Objects D	eleted Objects	Fragments		
Objects are basic You can use OBS Upload Objec	units of data storage. Ir Browser+ to move an o t Create Folder	n OBS, files and folders are treated as obj bject to any other folder in this bucket.	ects. Any file type can be uploaded and n	nanaged in a bucket. Learn more
Name	≡	Storage Class JΞ	Size J≡	Encrypted J≡
← Back				
basicim	g4.png	Standard	296.18 KB	No
basicim	g2.png	Standard	275.08 KB	No
basicim	g1.png	Standard	191.60 KB	No
basicim	g3.png	Standard	113.71 KB	No
basicim	g5.png	Standard	11.23 KB	No

If you want to use kubectl, you can use the following example YAML:

kind: Job
apiVersion: batch/v1
metadata:
name: testjob
namespace: default
spec:
parallelism: 1
completions: 1
backoffLimit: 6
template:
metadata:
name: testjob
spec:
volumes:
 name: cce-obs-tensorflow
persistentVolumeClaim:

```
claimName: cce-obs-tensorflow
containers:

    name: container-0

  image: 'tensorflow/tensorflow:1.15.5-gpu'
  restartPolicy: OnFailure
  command:
    - /bin/bash
  args:
   - '-c'
   - pip install matplotlib;python /home/basicClass.py
  resources:
   limits:
     cpu: '2'
     memory: 4Gi
     nvidia.com/gpu: '1'
   reauests:
     cpu: '2'
     memory: 4Gi
     nvidia.com/gpu: '1'
  volumeMounts:
    - name: cce-obs-tensorflow
     mountPath: /home
  imagePullPolicy: IfNotPresent
imagePullSecrets:
 - name: default-secret
```

15.4 Deploying and Using Flink in a CCE Cluster

Apache Flink is a framework and distributed processing engine for stateful computations over unbounded (streams) and bounded (batches) data streams. It can process data streams in real time with low latency and high throughput and process complex events. Deploying Flink in CCE clusters enables you to build a high-performance, reliable, and flexible data processing system for a wide range of applications in big data environments while ensuring optimal resource usage. This section describes how to deploy Flink in a CCE cluster and how to run a Flink WordCount job in the CCE cluster. In this example, the Flink cluster is deployed standalone. For details about the deployment process, see the **Kubernetes** | **Apache Flink**.

Prerequisites

- There is a cluster with certain nodes available for use. For details, see **Buying** a CCE Standard/Turbo Cluster.
- An EIP has been assigned to nodes in the cluster, and the kubectl has been configured. For details, see Binding an EIP to an Instance and Connecting to a Cluster Using kubectl.

Step 1: Deploy a Flink Cluster

Three key components are required for deploying a Flink cluster. The Flink official website provides a resource definition file for each component. For details, see **Table 15-1**. In addition, you need to use the **flink-configuration-configmap.yaml** configuration file on the Flink official website to configure the Flink cluster.

Key Component	Resource Definition File	Description
Deployment for running JobManager	jobmanager-session- deployment-non- ha.yaml	JobManager serves as the central coordinator in a Flink cluster. It coordinates Flink jobs, including task distribution, job scheduling, resource allocation, and fault tolerance.
Deployment for running TaskManager	taskmanager- session- deployment.yaml	TaskManager is a worker node in a Flink cluster and is responsible for executing data processing tasks. Each TaskManager runs one or more task slots, which are isolated units of execution.
Service exposing the JobManager's REST and UI ports	jobmanager- service.yaml	The REST and Web UI ports of Flink JobManager are exposed so that users can access the REST API and Web UI of JobManager through the Service.

Table 15-1 Key components of the Flink cluster

Step 1 Configure basic information about the Flink cluster.

1. Create a YAML file named **flink-configuration-configmap.yaml**. vim flink-configuration-configmap.yaml

Check that the file contains comments and the content is as follows:	
apiVersion: v1	
kind: ConfigMap	
metadata:	
name: flink-config	
labels:	
app: flink	
# data defines the data stored in ConfigMap. In the example, data contains two configurat	tion files:
config.yaml and log4j-console.properties.	
data:	
config.yaml: +	
# RPC address of Flink JobManager. It is usually the JobManager name. In this example, the	e RPC
address is flink-jobmanager .	
jobmanager.rpc.address: flink-jobmanager	
# Number of task slots in each TaskManager. Set the value to 2 , indicating that each TaskN	/lanager
can process two tasks concurrently.	
taskmanager.numberOfTaskSlots: 2	
# Port of the Flink BLOB service. It is used to transfer large objects, such as job code or larg	je files.
blob.server.port: 6124	
jobmanager.rpc.port: 6123 # RPC port of JobManager	
taskmanager.rpc.port: 6122 # RPC port of TaskManager	
jobmanager.memory.process.size: 1600m # Total memory of JobManager	
taskmanager.memory.process.size: 1/28m # Total memory of TaskManager	
parallelism.default: 2 # The default degree of parallelism is 2 for Flink jobs.	
tog4j-console.properties: +	
# The following comiguration affects the logging for user code and Flink logs.	
rootLogger appender Def concele ref = Concele Appender # Conditional above.	10
rootLogger.appenderRef.colling.ref = CollingEileAppender # Send the logs to the consol	ile.
# If you only what to change the longing in Elipty delate the company in the following	linos:
# in you only want to change the togging in runk, detete the comments in the following #loagerflink name – organache flink	unes.
#togget.turk.harrie = org.apache.turk	

2.

3.

#logger.flink.level = INFO

```
# The following eight lines keep the log level of the public libraries or connectors at INFO.
                  # The configuration of the root logger does not overwrite the configuration here.
                  # You need to manually change the log level.
                  logger.pekko.name = org.apache.pekko
                  logger.pekko.level = INFO
                  logger.kafka.name= org.apache.kafka
                  logger.kafka.level = INFO
                  logger.hadoop.name = org.apache.hadoop
                  logger.hadoop.level = INFO
                  logger.zookeeper.name = org.apache.zookeeper
                  logger.zookeeper.level = INFO
                  # Send all logs of the INFO level to the console.
                  appender.console.name = ConsoleAppender
                  appender.console.type = CONSOLE
                  appender.console.layout.type = PatternLayout
                  appender.console.layout.pattern = %d{yyyy-MM-dd HH:mm:ss,SSS} %-5p %-60c %x - %m%n
                  # Export all logs of the INFO level to a specified scrolling file.
                  appender.rolling.name = RollingFileAppender
                  appender.rolling.type = RollingFile
                  appender.rolling.append = false
                  appender.rolling.fileName = ${sys:log.file}
                  appender.rolling.filePattern = ${sys:log.file}.%i
                  appender.rolling.layout.type = PatternLayout
                  appender.rolling.layout.pattern = %d{yyyy-MM-dd HH:mm:ss,SSS} %-5p %-60c %x - %m%n
                  appender.rolling.policies.type = Policies
                  appender.rolling.policies.size.type = SizeBasedTriggeringPolicy
                  appender.rolling.policies.size.size=100MB
                  appender.rolling.strategy.type = DefaultRolloverStrategy
                  appender.rolling.strategy.max = 10
                  # Disable false alarms in the Netty channel handler.
                  logger.netty.name = org.jboss.netty.channel.DefaultChannelPipeline
                  logger.netty.level = OFF
               Use flink-configuration-configmap.yaml to configure basic information
               about the Flink cluster.
               kubectl create -f flink-configuration-configmap.yaml
               Check whether the ConfigMap named flink-config is successfully created.
               kubectl get configmap
               If the following information is displayed, the ConfigMap was created
               successfully.
               NAME
                               DATA AGE
               flink-config
                                    595
                               2
               kube-root-ca.crt 1
                                   16d
Step 2 Create a Service that exposes the REST and UI ports of JobManager.
```

Create a YAML file named jobmanager-service.yaml. 1. vim jobmanager-service.yaml

```
Check that the file contains comments and the content is as follows:
```

```
apiVersion: v1
kind: Service
metadata:
name: flink-jobmanager
spec:
 type: ClusterIP # The Service is used for internal communication within the cluster.
               # Define the list of ports to be exposed by Service.
 ports:
 - name: rpc
  port: 6123
 - name: blob-server
  port: 6124
 - name: webui
  port: 8081
```

selector: # Define the Service's label selector, which is used to determine the pods to which the Service routes traffic. app: flink

component: jobmanager

- 2. Use **jobmanager-service.yaml** to create a Service named **flink-jobmanager**. kubectl create -f jobmanager-service.yaml
- 3. Check whether the Service is created successfully. kubectl get service flink-jobmanager

If the following information is displayed, the Service was successfully created. NAME TYPE CLUSTER-IP EXTERNAL-IP PORT(S) AGE flink-jobmanager ClusterIP 10.247.199.212 <none> 6123/TCP,6124/TCP,8081/TCP 115s

Step 3 Create a Deployment for running JobManager.

1. Create a YAML file named **jobmanager-session-deployment-non-ha.yaml**. vim jobmanager-session-deployment-non-ha.yaml

Check that the file contains comments and the content is as follows:

```
apiVersion: apps/v1
kind: Deployment
metadata:
name: flink-jobmanager
spec:
                       # Set the number of JobManager replicas to 1.
 replicas: 1
 selector:
  matchLabels:
                         # Define labels.
    app: flink
   component: jobmanager
 template:
  metadata:
   labels:
     app: flink
     component: jobmanager
  spec:
   containers:
    - name: jobmanager
                            # Set the container name to jobmanager.
     image: apache/flink:1.20.0-scala_2.12 # Use the Flink image of v1.20.0 and the Scala of v2.12.
     args: ["jobmanager"] # Designate the container to run as JobManager.
     ports:
                      # Expose ports in the container.
     - containerPort: 6123 # Used for communication between TaskManager and JobManager.
      name: rpc
     - containerPort: 6124 # Used to transfer binary objects.
      name: blob-server
     - containerPort: 8081 # Used to access the Flink web management page.
      name: webui
     livenessProbe:
      tcpSocket:
       port: 6123
                        # Use TCP to check the health status of RPC port 6123.
      initialDelaySeconds: 30
      periodSeconds: 60
     volumeMounts:
                           # Mount a storage volume.
     - name: flink-config-volume
      mountPath: /opt/flink/conf
     securityContext:
                          # For details, see the _flink_User in the official Flink image. You can
      runAsUser: 9999
change the username if necessary.
   volumes:
                        # Define storage volumes to store configuration files.
    - name: flink-config-volume
     configMap:
      name: flink-config
      items:
      - key: config.yaml # Mount the config.yaml file in the ConfigMap to the specified path of the
container.
        path: config.yaml # The path in the container is /opt/flink/conf/config.yaml.
       - key: log4j-console.properties # Mount the log4j-console.properties file in the ConfigMap
to the specified path of the container.
        path: log4j-console.properties # The path in the container is /opt/flink/conf/log4j-
console.properties.
```

2. Use **jobmanager-session-deployment-non-ha.yaml** to create a Deployment named **flink-jobmanager**.

kubectl create -f jobmanager-session-deployment-non-ha.yaml

3. Check whether the Deployment **flink-jobmanager** is successfully created. kubectl get pod

The Deployment was successfully created if the following information is displayed:

NAMEREADYSTATUSRESTARTSAGEflink-jobmanager-789c8777-vhqbv1/1Running097s

Step 4 Create a Deployment for running TaskManager.

1. Create a YAML file named **taskmanager-session-deployment.yaml**. vim taskmanager-session-deployment.yaml

Check that the file contains comments and the content is as follows:

```
apiVersion: apps/v1
kind: Deployment
metadata:
 name: flink-taskmanager
spec:
                       # Set the number of TaskManager replicas to 2.
 replicas: 2
 selector:
                          # Define labels.
  matchLabels:
   app: flink
   component: taskmanager
 template:
  metadata:
   labels:
     app: flink
     component: taskmanager
  spec:
   containers:
    - name: taskmanager
                             # Set the container name to taskmanager.
     image: apache/flink:1.20.0-scala_2.12 # Use the Flink image of v1.20.0 and the Scala of v2.12.
     args: ["taskmanager"] # Designate the container to run as TaskManager.
     ports:
                      # Expose ports in the container.
     - containerPort: 6122 # Used for communication between TaskManager and JobManager.
      name: rpc
     livenessProbe:
      tcpSocket:
        port: 6122
                        # Use TCP to check the health status of RPC port 6122.
      initialDelaySeconds: 30
      periodSeconds: 60
     volumeMounts:
                           # Mount a storage volume.
     - name: flink-config-volume
      mountPath: /opt/flink/conf/
     securityContext:
      runAsUser: 9999
                          # For details, see the _flink_User in the official Flink image. You can
change the username if necessary.
   volumes:
                        # Define storage volumes to store configuration files.
    - name: flink-config-volume
     configMap:
      name: flink-config
      items:
      - key: config.yaml
        path: config.yaml
      - key: log4j-console.properties
        path: log4j-console.properties
Use taskmanager-session-deployment.yaml to create a Deployment named
flink-taskmanager.
kubectl create -f taskmanager-session-deployment.yaml
```

3. Check whether the Deployment **flink-taskmanager** is successfully created. kubectl get pod

2.

The Deployment was successfully created if the following information is displayed:

NAME READY	STATU	IS	RESTARTS	AGE
flink-jobmanager-789c8777-vhqbv	1/1	Running	0	13m
flink-taskmanager-579f47cf9f-prrff	1/1	Running	0	23s
flink-taskmanager-579f47cf9f-wgt6	6 1/1	Runnin	g 0	23s

----End

Step 2: Publish the Service

Create a NodePort Service for **flink-jobmanager** to allow external networks to access **flink-jobmanager** through the public IP address and automatically allocated external port number. The Service will forward external requests to the corresponding container.

- Step 1 Log in to the CCE console. Choose Workloads > Deployments, click flinkjobmanager, the Access Mode tab, and then Create Service.
- Step 2 On the Create Service page, set Service Type to NodePort. In the Ports area, set both Container Port and Service Port to 8081, and click OK. The Service automatically generates a port for accessing the node. The port is displayed in Figure 15-3. In this example, the port is 30327. You can access the workload using the EIP and the port of any node in the cluster.

Figure 15-2 Creating a NodePort Service

Create Service	Create from YAML)
Service Name	flink-jobmanager-06422				
Service Type	ClusterIP Expose services through the internal IP of the cluster, which can only be accessed within the cluster	NodePort Expose services via IP and static port (NodePort) on each node	LoadBalancer Provide external services through ELB load balancing, high availability, ultra-high performance, stability and security DNAT Expose cluster node access type services through NAT gateway, support multiple nodes to share and use elastic IP		
Service Affinity	Cluster-level Node-level	0			
Ports	Protocol Container	Port ? Service Port ? No	ode Port		0
	TCP v 8081	8081	Auto 🗸		Delete
			+		

Figure 15-3 NodePort

Pods Access Mode Containers Auto Sca	ing Scheduling Policies Change Hi	story Events Security	Droups APM Settings							
Services										
Create Service Doints Dipot ~	Create Services Expert v 🗸 Outer (semanteptitel), Land Salamouri 1980, Outer v 2084 2000, Salamotel ECC 30271 (200)									
 Select a property or enter a keyword. 							Q 0			
Service 8	Selector 0	Namespace ()	Service Type (8)	Access Address 0	Access Port Container Port/Protocol 0	Created B	Operation			
O fink-jobmanager	app fink component jobmanager	default	ClusterP	10.247.190.151 (Cluster IP)	6123 → 6123 / TCP 6124 → 6124 / TCP 8081 → 8081 / TCP	1 minutes ago	Manage Pod. View Events. More \sim			
O fink-jobmanager-96422	app fink component jobmanager	default	NodePort	10.247.235.193 (Charles IP)	8081 -> 8081 / TCP 31823 / TCP	2 minutes ago	Managa Pod. View Events. More \sim			
Total Records: 2							10 < (1) >			

Step 3 Check whether the Service can be accessed. Choose **Nodes**, click the **Nodes** tab, select a node, and copy its EIP.

In the address box of a browser, enter *EIP of the node.Port for accessing the node.* If the Flink dashboard page is displayed, the access is successful. If the access failed, check whether the source IP address for the node port is set to **0.0.0.0/0** or **All** in the inbound rule of the cluster security group. For details, see **Configuring Security Group Rules**.

Fiaure	15-4	Flink	Dashboard

🌰 Apache Flink Dashboard					Version: 120.0 Commit: b1fe7b4	0 2034-07-2510422:22+02:00	Message: Ø
Demmon Josts Orrented lobs Direct Managers	Australiate trait States 4 Total has State 4 — base Melengenes 2 Recenting Job List		Ruming John O Frankel 9 - Connel 8 - Factor 0				
aP Job Manager ∴ Submit New Job	Jakhane	C Start Time	Duration	2 End Time	t Tasks	Status	
	Completed Job List	0 Start Time	0 Duration	0 End Time	0 Tasks	Suna	0

----End

Step 3: Run the Flink Job

Use the official **WordCount.jar** file to demonstrate how to execute Flink jobs in a CCE cluster. The WordCount task is to calculate the number of occurrences of each word in the text.

- Step 1 Download and decompress the flink-1.20.0-bin-scala_2.12.tgz file. The file can be obtained at https://archive.apache.org/dist/flink/flink-1.20.0/flink-1.20.0-bin-scala_2.12.tgz. Check whether the WordCount.jar package exists in the flink-1.20.0-bin-scala_2.12\flink-1.20.0\examples\streamin directory.
- Step 2 Add a .jar package on the Dashboard page. Open the Apache Flink Dashboard page, choose Submit New Job from the navigation tree, click Add New in the upper right corner, and select WordCount.jar in the flink-1.20.0-bin-scala_2.12\flink-1.20.0\examples\streamin directory. Click the WordCount.jar file and specify the output file path, for example, --output /opt/flink/output in the Program Arguments text box.

Figure 15-5 Uploading a WordCount job

📥 Apache Flink Dashboard	21			Vensieer 1.20.0 Cererrit: 015/704 @ 2024-07-2570422.22+02:00 Message: 🏮
© Overview	Uploaded Jars			T Ant New.
≡ Jobs ~				
63 Task Managers	Name	Upload Time	Entry Class	
	WordCount.jar	2024-09-02, 09:35:30	orgapache finkstreaming examples wordcount WordCount	Delete
dr 166 Manager	A og apache/firk.streaming.exam	ples.wordcount.WordCount	@Parallelism	
🙏 Subrat New Job	cutput: /opt/Tink/output		In Sampoint Path	
	Allow Non Restored State		Show Plan Subert	

Step 3 Click the blue box on the lower right of **Overview** and click **Taskmanager** to check the endpoint of the job.

Figure 15-6 Checking an endpoint

🗼 Apache Fink Dashboard	8										Version	120.0 Commit: 01fe764 @ 20	24-07-25704:22:3	M 00:50+5	essage: 🜖
Overview	WordCount														
≡ Jots ^	Job ID	3/5/25975917653803eef5	c\$\$dc8b3cd		Job	State	TINISHED	2		Ac	tions	Job Manager Log			
② Running Jobs	Job Type	STREAMING			24	t Time	2024-09-0	2 02:47:55.047		En	d Time	2024-09-02 09:47:55.194			
Correlated Inte	Duration	147mi													
C . Data Manager	Overview Exceptions Dat	ta Slowy TimeLine C	heckpoints Configuration												
2) task homogens						Detail SubTanka	TaskManagera	Watermarka Accurre	latora Backi	Pressure Metrica	FlameScaph				
d" Job Managar					6	Endpoint 0	Bytes received	0 Records received 0	Dytes cent.	0 Records sent	0 Start Time	0 End Time	0 Duration	Status	More
 Submit New Job 						192,165-0.92,37259	6.21 XB	287	0.5		2024-09-02 09:47:55.128	2024-09-02 09:47:55.194	60ms	INSIED	
	Source Researcy appl (1) Internet Reactions (1) Pedagement (1) Responsed (1) Source Reaction Source Reaction Source Reaction	- 59 (excenter -+ IFs white Utilitar In-white Committee Paralitations 1 Readyments of local Model Starty Image No.A Out Show No.A	n	> <										
	Name		Status 0 Bytes Received	0 Records Received	0 Bytes Sent	0 Records Se	et 0 Paralelos	0 Start Time	0 Duration	0 End Time	0 Tacks				
	Source in-memory-input -> tokeniter counter -> file-sink Writer -> file-sink C	i annite	FINSIED 0.8 FINSIED 6,2140	207	6.19 KD	207		2024-09-02 09-47-55.12	6 33ms	2024-09-0	2 09.47.55.159 1				
	counter -> file-sink Writer -> file-sink C	onniter	TINSIED 621/0	267	0.0			2024-09-02 0947-55.12	6 60ms	2034-89-0	2 09:47:55:194				

Step 4 Use the endpoint to obtain the TaskManager pod. Run the following command to query the IP address of the Flink pod: kubectl get pod -o wide | grep flink

Rubecti get pou -o wide | grep fillik

flink-taskmanager-579f47cf9f-prrff is the TaskManager pod if the following information is displayed:

flink-jobmanag	ger-789c8777	-vhqbv	1/1	Running	1 (28m ago)	40h	192.168.0.139	
192.168.0.53	<none></none>	<none></none>		-				
flink-taskmana	ger-579f47cf	9f-prrff	1/1	Running	1 (28m ago)	40h	192.168.0.92	
192.168.0.53	<none></none>	<none></none>						
flink-taskmana	ger-579f47cf	9f-wgt66	1/1	Running	1 (28m ago)	40h	192.168.0.194	
192.168.0.212	<none></none>	<none></none>						

Step 5 After the job is complete, go to **flink-taskmanager-579f47cf9f-prrff** to check whether the number of occurrences of each word is correctly displayed. kubectl exec -it *flink-taskmanager-579f47cf9f-prrff* -- bash

Run the **ls** command to query the output path.

ls /opt/flink/output/

Information similar to the following is displayed:

2024-09-02--01

Check the content of the 2024-09-02--01 folder.

ls /opt/flink/output/2024-09-02--01

Information similar to the following is displayed:

part-bd89ad8b-a0dd-4b4d-b771-4c88eaed61e4-0

Check the number of occurrences of each word.

cat /opt/flink/output/2024-09-02--01/part-bd89ad8b-a0dd-4b4d-b771-4c88eaed61e4-0

Information similar to the following is displayed:

(to,1) (be,1) (or,1) (not,1) (to,2) (be,2) (that,1)

----End

Step 4: Clear the Cluster

Step 1	Delete the Deployment that runs the JobManager.
	kubectl delete -f jobmanager-session-deployment-non-ha.yaml
	Information similar to the following is displayed:
	deployment.apps "flink-jobmanager" deleted
Step 2	Delete the Deployment that runs the TaskManager.
	kubectl delete -f taskmanager-session-deployment.yaml
	Information similar to the following is displayed:
	deployment.apps "flink-taskmanager" deleted
Step 3	Delete the ConfigMap.
	kubectl delete -f flink-configuration-configmap.yaml
	Information similar to the following is displayed:
	configmap "flink-config" deleted
Step 4	Delete the Service.
	kubectl delete -f jobmanager-service.yaml
	Information similar to the following is displayed:
	service "flink-jobmanager" delete

----End

15.5 Deploying and Using ClickHouse in a CCE Cluster

ClickHouse is a columnar database management system for online analytical processing (OLAP). It is suitable for real-time query and analysis of large-scale datasets. There are four ways to deploy ClickHouse on containers. For details, see **Table 15-2**. ClickHouse Operator is a tool for deploying and managing ClickHouse in Kubernetes clusters. It can replicate clusters and manage users, configuration files, and persistent volumes. These functions simplify application configuration, management, and monitoring.

Deployment Method	Difficulty in Deployment	Difficulty in Management	
Native kubectl	Difficult	Difficult	
kubectl and Operator	Medium	Medium	
Helm	Easy	Difficult	
Helm and Operator	Easy	Easy	

Table 15-2 ClickHouse deployment on contain	ners
---	------

The following describes how to deploy ClickHouse in a CCE cluster using kubectl and Operator. For details, see https://github.com/Altinity/clickhouse-operator.

Prerequisites

- There is a cluster with certain nodes available for use. For details, see **Buying** a CCE Standard/Turbo Cluster.
- An EIP has been assigned to nodes in the cluster, and the kubectl has been configured. For details, see **Binding an EIP to an Instance** and **Connecting to a Cluster Using kubectl**.

Procedure for Deploying ClickHouse

The following describes how to deploy ClickHouse in a CCE Turbo cluster of v1.29. For details about the cluster parameters, see **Table 15-3**.

Parameter	Value
Туре	CCE Turbo Cluster
Cluster Version	1.29
Region	AP-Singapore
Container Engine	containerd
Network Model	Cloud Native Network 2.0
Request Forwarding	iptables

 Table 15-3 Cluster parameters

Step 1 Create a ClickHouse Operator.

1. Download the YAML file clickhouse-operator-install-bundle.yaml from https://github.com/Altinity/clickhouse-operator/blob/master/deploy/ operator/clickhouse-operator-install-bundle.yaml.

NOTE

clickhouse-operator-install-bundle.yaml is used to deploy the ClickHouse Operator in the **kube-system** namespace to monitor the resources in all Kubernetes namespaces. If the ClickHouse Operator is deployed in another namespace, only resources in that namespace are monitored.

kubectl apply -f clickhouse-operator-install-bundle.yaml

Information similar to the following is displayed:

custom resource definition. a piextensions. k8s. io/clickhouse installations. clickhouse. altinity.com created custom resource definition. a piextensions. k8s. io/clickhouse installation templates. clickhouse. altinity.com created

custom resource definition. a piextensions. k \$s. io/clickhouse operator configurations. clickhouse. altinity. com created

2. Check whether the ClickHouse Operator is successfully created. kubectl get pod -n kube-system | grep clickhouse

If the pod status is **Running**, the ClickHouse Operator was successfully created.

clickhouse-operator-656d67bd4d-k64gm 2/2 Running 4 (15m ago) 3d23h

3. Check all CRD resources related to ClickHouse in the cluster. kubectl get crd | grep clickhouse

Information similar to the following is displayed:

clickhouseinstallations.clickhouse.altinity.com2024-08-20T09:30:30Zclickhouseinstallationtemplates.clickhouse.altinity.com2024-08-20T09:30:30Zclickhousekeeperinstallations.clickhouse-keeper.altinity.com2024-08-20T09:30:30Zclickhouseoperatorconfigurations.clickhouse.altinity.com2024-08-20T09:30:30Z

- **Step 2** Create namespace **test-clickhouse-operator**. To facilitate the verification, the subsequent operations are all performed in **test-clickhouse-operator**. kubectl create namespace *test-clickhouse-operator*
- **Step 3** Create a ClickHouse cluster.
 - Create a YAML file named simple-01.yaml. You can obtain simple-01.yaml from https://raw.githubusercontent.com/Altinity/clickhouse-operator/ master/docs/chi-examples/01-simple-layout-01-1shard-1repl.yaml. vim simple-01.yaml

NOTE

ClickHouseInstallation is a custom resource object (CR) defined when a ClickHouse Operator is used in a Kubernetes cluster. After **ClickHouseInstallation** resources are created or updated, the ClickHouse Operator automatically creates and manages Kubernetes resources, such as StatefulSets, Services, and PersistentVolumeClaims, to ensure that the ClickHouse cluster runs as expected.

The file content is as follows:

```
apiVersion: "clickhouse.altinity.com/v1"
kind: "ClickHouseInstallation"
metadata:
 name: "simple-01"
spec:
 configuration:
  users:
    # printf 'test_password' | sha256sum
   test_user/password_sha256_hex:
10a6e6cc8311a3e2bcc09bf6c199adecd5dd59408c343e926b129c4914f3cb01
   test_user/password: test_password
    # to allow access outside from kubernetes
   test_user/networks/ip:
   - 0.0.0/0
  clusters:
  - name: "simple"
```

Use the preceding file to create a ClickHouse cluster.

kubectl apply -n test-clickhouse-operator -f simple-01.yaml

Step 4 Check whether ClickHouse resources are successfully created.

- Check the pods of the test-clickhouse-operator namespace. If all pods are in the Running state, the pods were successfully created. kubectl get pod -n test-clickhouse-operator
 - Information similar to the following is displayed:

NAME	READY	STA	TUS	RES	TARTS	AGE
chi-simple-01-simple-0-0	-0 2	2/2	Runr	ning	0	3d7h

2. Check the other service resources. kubectl get service -n *test-clickhouse-operator*

Information similar to the following is displayed:

NAME	TYPE CL	JSTER-IP	EXTERNAL-IP	PORT(S)	AGE	
chi-simple-01-simple-0-	0 ClusterII	P None	<none></none>	9000/TCP,812	23/TCP,9009/TCP	3d7h
clickhouse-simple-01	ClusterIP	None	<none></none>	9000/TCP,8123	B/TCP 3d8	h

Step 5 Connect to the ClickHouse database.

kubectl -n test-clickhouse-operator exec -ti chi-simple-01-simple-0-0-0 -- clickhouse-client

If the following information is displayed, the connection is successful: Enter **exit** and press **Enter** to exit the ClickHouse database.

ClickHouse client version 24.8.2.3 (official build). Connecting to localhost:9000 as user default. Connected to ClickHouse server version 24.8.2.

Warnings:

* Linux transparent hugepages are set to "always". Check /sys/kernel/mm/transparent_hugepage/enabled

chi-simple-01-simple-0-0.chi-simple-01-simple-0-0.test-clickhouse-operator.svc.cluster.local :)

Step 6 Clear ClickHouse cluster resources.

Run the following command to delete the ClickHouse cluster:

kubectl delete -f simple-01.yaml -n test-clickhouse-operator

Information similar to the following is displayed:

clickhouseinstallation.clickhouse.altinity.com "simple-01" deleted

----End

Example 1: Creating a ClickHouse Cluster with a PV Provisioned Dynamically

The following example describes how to create a ClickHouse cluster with a PV dynamically provisioned. The EVS is used as an example to describe how to dynamically provision a PV for a ClickHouse cluster.

NOTICE

The VolumeClaimTemplate can only be used to provision EVS disks and local PVs to StatefulSets.

Step 1 Create a StorageClass.

1. Create a YAML file named csi-disk-ssd.yaml. vim csi-disk-ssd.yaml

By default, CCE supports SAS disks. If you want to use another type of disk, you need to create the corresponding StorageClass. For details about StorageClass parameters, see **Table 15-4**.

allowVolumeExpansion: true apiVersion: storage.k8s.io/v1 kind: StorageClass metadata: name: *csi-disk-ssd* provisioner: everest-csi-provisioner parameters: csi.storage.k8s.io/csi-driver-name: disk.csi.everest.io csi.storage.k8s.io/fstype: ext4 everest.io/disk-volume-type: *SSD* everest.io/passthrough: "true" reclaimPolicy: Delete volumeBindingMode: Immediate

provisioner Specifies the storage resource provide add-on for CCE. Set this parameter to provisioner.	er, which is the Everest e everest-csi- ich vary with storage
noremotors Consisting the store as noremotors white	ich vary with storage
parameters Specifies the storage parameters, while types. NOTICE everest.io/disk-volume-type indicates the can be any of the following: - SAS: high I/O - SSD: ultra-high I/O - GPSSD: general purpose SSD - ESSD: extreme SSD - GPSSD2: general purpose SSD v2, white Everest version is 2.4.4 or later and the everest.io/disk-throughput annotation - ESSD2: extreme SSD v2, which is suppresent is 2.4.4 or later and the everest is configured.	ich is supported when the e everest.io/disk-iops and ons are configured. ported when the Everest st.io/disk-iops annotation
Default: SAS	
reclaimPolicySpecifies the value of persistentVoluce creating a PV. The value can be Delete reclaimPolicy is not specified when a created, the value defaults to DeleteDelete: indicates that a dynamical be automatically deleted when the e retained when the PVC is deleted	meReclaimPolicy for te or Retain. If a StorageClass object is Ily provisioned PV will e PVC is deleted. Ily provisioned PV will ed.
volumeBindi ngModeSpecifies when a PV is dynamically pr can be Immediate or WaitForFirstCo - Immediate: The PV is dynamically PVC is createdImmediate: The PV is dynamically PVC is createdWaitForFirstConsumer: The PV is	rovisioned. The value onsumer. v provisioned when a dynamically

Table 15-4 StorageClass parameters

- 2. Use **csi-disk-ssd.yaml** to create a StorageClass named **csi-disk-ssd**. kubectl create -f *csi-disk-ssd.yaml*
- **Step 2** Create a ClickHouse cluster with a PV dynamically provisioned.
 - Create a YAML file named pv-simple.yaml. vim pv-simple.yaml
 For details about the file content, see https://github.com/Altinity/ clickhouse-operator/blob/master/docs/chi-examples/03-persistentvolume-01-default-volume.yaml.

Step 3

NOTICE

EVS disks can be mounted as read-write by a single node, so **accessModes** must be set to **ReadWriteOnce**.

	apiVersion: "clickhouse.altinity.com/v1" kind: "ClickHouseInstallation"
	metadata:
	name: " <i>pv-simple</i> "
	namespace: test-clickhouse-operator
	spec:
	defaults:
	templates:
	dataVolumeClaimTemplate: data-volume-template
	logvolumeClaimTemplate: log-volume-template
	- name: "simple"
	lavout:
	shardsCount: 1
	replicasCount: 1
	templates:
	volumeClaimTemplates: # Dynamic provisioning
	- name: <i>data-volume-template</i> # Template for defining a data storage volume
	spec:
	accessModes:
	- ReadwriteOnce # EVS disks can be mounted as read-write by a single node, so
	resources
	requests
	storage: 10Gi
	storageClassName: csi-disk-ssd # Specify the newly created csi-disk-ssd as the StorageClass.
	- name: <i>log-volume-template</i> # Template for defining the log storage volume
	spec:
	accessModes:
	- ReadWriteOnce # EVS disks can be mounted as read-write by a single node, so
	resources
	requests:
	storage: 10Gi
	storageClassName: csi-disk-ssd # Specify the newly created csi-disk-ssd as the StorageClass.
2	Use ny-simple yami to create a ClickHouse cluster
<u>∠</u> .	ubach, a tast dickause anartasta f nu simila uan
	Rubecu - II test-cucknouse-operator create - I pv-simple.yami
Cheo	ck whether the ClickHouse cluster is successfully created and whether the PV is
	essfully provisioned
succ	essially provisioned.
1.	Check the pods of the test-clickhouse-operator namespace. If all pods are in
	the Running state the pods were successfully created
	kubectl get nod -n <i>test-clickhouse-operator</i>
	If the following information is displayed, the pods are successfully created.
	NAME READY STATUS RESTARTS AGE
	chi-pv-simple-simple-0-0-0 2/2 Running 0 5m2s
	chi-simple-01-simple-0-0-0 1/1 Running 0 3d7h
2	Check whether the PV/Cs named data-volume-template and log-volume-
∠.	template are sussessfully greated
	If STATUS is Bound , the PVC is bound successfully.
	MODES STORAGECLASS VOLUMEATTRIBUTESCLASS AGE
	data-volume-template-chi-pv-simple-simple-0-0-0 Bound pvc-981b1d73-a13e-41d5-aade-
	ea8c6b1199d7 10Gi RWO csi-disk-ssd <unset> 28s</unset>

log-volume-template-chi-pv-simple-simple-0-0-0 Bound pvc-fcf70a2e-131d-4da1-a9c2eddd89887b45 10Gi RWO csi-disk-ssd <unset> 28s

3. Check whether the PV is mounted to the cluster.

Go to the CLI of the **chi-pv-simple-simple-0-0-0** container.

kubectl -n test-clickhouse-operator exec -ti chi-pv-simple-simple-0-0-0 -c clickhouse bash

Check whether the PV is mounted to the container:

df -h

The command output shows that the PV has been mounted to the container. You can press **Ctrl+D** to exit the CLI.

Filesystem	Size	Used Avail Use% Mounted on
overlay	99G	5.1G 89G 6% /
tmpfs	64M	0 64M 0%/dev
tmpfs	3.9G	0 3.9G 0% /sys/fs/cgroup
/dev/mapper/vgpaas-	share	99G 5.1G 89G 6% /etc/hosts
shm	64M	0 64M 0% /dev/shm
/dev/sdb	9.8G	66M 9.8G 1% /var/lib/clickhouse
/dev/sda	9.8G	37M 9.8G 1% /var/log/clickhouse-server
tmpfs	6.3G	12K 6.3G 1% /run/secrets/kubernetes.io/serviceaccount
tmpfs	3.9G	0 3.9G 0% /proc/acpi
tmpfs	3.9G	0 3.9G 0% /proc/scsi
tmpfs	3.9G	0 3.9G 0% /sys/firmware

Step 4 Connect to the ClickHouse database.

kubectl -n *test-clickhouse-operator* exec -ti chi-pv-simple-simple-0-0-0 -- clickhouse-client

If the following information is displayed, you have successfully connected to the ClickHouse database: Enter **exit** and press **Enter** to exit the ClickHouse database.

Defaulted container "clickhouse" out of: clickhouse, clickhouse-log ClickHouse client version 24.8.2.3 (official build). Connecting to localhost:9000 as user default. Connected to ClickHouse server version 24.8.2.

Warnings:

* Linux transparent hugepages are set to "always". Check /sys/kernel/mm/transparent_hugepage/enabled

chi-pv-simple-simple-0-0-0.chi-pv-simple-simple-0-0.test-clickhouse-operator.svc.cluster.local :)

Step 5 Clear ClickHouse cluster resources.

Run the following command to delete the ClickHouse cluster with a PV provisioned dynamically:

kubectl delete -f *pv-simple.yaml* -n *test-clickhouse-operator*

Information similar to the following is displayed:

clickhouseinstallation.clickhouse.altinity.com "pv-simple" deleted

----End

Example 2: Creating a ClickHouse Cluster with a LoadBalancer Service

The following example describes how to create a ClickHouse cluster with a LoadBalancer Service. The LoadBalancer Service allows you to access the ClickHouse cluster from the Internet.

- **Step 1** Create a ClickHouse cluster with a LoadBalancer Service so that you can access the ClickHouse cluster from the Internet.
 - 1. Create a YAML file named **elb.yaml**. vim elb.yaml

```
apiVersion: "clickhouse.altinity.com/v1"
kind: "ClickHouseInstallation"
metadata:
 name: "ck-elb"
namespace: test-clickhouse-operator
spec:
defaults:
  templates:
   dataVolumeClaimTemplate: data-volume-nas
   serviceTemplate: chi-service-elb
 configuration:
  clusters:
    - name: "ck-elb"
     templates:
      podTemplate: pod-template-with-nas
     layout:
      shardsCount: 1
      replicasCount: 1
 templates:
  podTemplates:
    - name: pod-template-with-nas
    spec:
      containers:
       - name: clickhouse
         image: clickhouse/clickhouse-server:23.8
         volumeMounts:
          - name: data-volume-nas
           mountPath: /var/lib/clickhouse
  volumeClaimTemplates:
                                  # Specify the storage access mode, requested storage size, and
StorageClass.
   - name: data-volume-nas
    spec:
      accessModes:
       - ReadWriteOnce
      resources:
       requests:
         storage: 20Gi
      storageClassName: csi-disk-ssd
  serviceTemplates:
                                # Service template
    - name: chi-service-elb
     metadata:
      annotations:
       # Load balancer type. union (default value) indicates shared load balancers, and
performance indicates dedicated load balancers.
       kubernetes.io/elb.class: union
       # Automatically create a load balancer associated with the ingress and define load balancer
parameters.
       kubernetes.io/elb.autocreate: >-
         {"type":"public","bandwidth_name":"cce-bandwidth-
ck","bandwidth_chargemode":"bandwidth","bandwidth_size":5,"bandwidth_sharetype":"PER","ei
p_type":"5_bgp"}
     spec:
      ports:
       - name: http
        port: 8123
       - name: client
         port: 9000
      type: LoadBalancer # Set the Service type to LoadBalancer.
```

For details about parameters in **kubernetes.io/elb.autocreate**, see **Table 15-5**.

Paramete r	Mandat ory	Туре	Description
type	No	Strin g	Network type of the load balancer. – public : public network load balancer – inner : private network load balancer Default: inner
bandwidth _name	Yes for public network load balancer s	Strin g	Bandwidth name. The default value is cce- bandwidth- *****. The value can contain 1 to 64 characters. Only letters, digits, underscores (_), hyphens (-), and periods (.) are allowed.
bandwidth _chargem ode	No	Strin g	Bandwidth billing mode. – bandwidth : billed by bandwidth – traffic : billed by traffic Default: bandwidth
bandwidth _size	Yes for public network load balancer s	Inte ger	 Bandwidth size. The default value is 1 to 2000 Mbit/s. Configure this parameter based on the bandwidth range allowed in your region. The minimum increment for bandwidth adjustment varies depending on the bandwidth range. If the allowed bandwidth does not exceed 300 Mbit/s, the minimum increment is 1 Mbit/s. If the allowed bandwidth is greater than 300 Mbit/s but less than or equal to 1000 Mbit/s, the minimum increment is 50 Mbit/s. If the allowed bandwidth exceeds 1000 Mbit/s, the minimum increment is 500 Mbit/s.
bandwidth _sharetype	Yes for public network load balancer s	Strin g	Specifies the bandwidth sharing mode. PER indicates that the bandwidth is dedicated.

Table 15-5 Parameters in the kubernetes.io/elb.autocreate file

Paramete r	Mandat ory	Туре	Description
eip_type	Yes for public network load balancer s	Strin g	 EIP type. 5_bgp: Dynamic BGP 5_sbgp: Static BGP The types vary by region. For details, see the EIP console.

- 2. Use **elb.yaml** to create a ClickHouse cluster. kubectl create -f *elb.yaml* -n *test-clickhouse-operator*
- **Step 2** Check whether the ClickHouse cluster is successfully created and associated with a LoadBalancer Service.
 - 1. Check the pods of the **test-clickhouse-operator** namespace. If all pods are in the **Running** state, the pods were successfully created. kubectl get pod -n *test-clickhouse-operator*

If the following information is displayed, the ClickHouse cluster is successfully created:

NAME	READY	STATUS	REST	ARTS AGE
chi-ck-elb-ck-elb-0-0-0	1/1	Runnir	ng O	3m4s
chi-pv-simple-simple-0-0	-0 2/	2 Runn	ing 0	33m
chi-simple-01-simple-0-0	-0 1/	1 Runn	ing 0	3d7h

2. Check whether the LoadBalancer Service is successfully created: kubectl get svc -n *test-clickhouse-operator*

If the following information is displayed, the LoadBalancer Service is successfully created:

NAME chi-ck-elb-ck-elb-0-0 chi-pv-simple-simple-0- 35m	TYPE CL ClusterIP 0 ClusterIP	USTER-IP None None	EXTERNAL-IP <none> <none></none></none>	PORT(S) 9000/TCP,8123/TCP,9 9000/TCP,8123/TCP	AGE 9009/TCP 2s 9009/TCP
chi-simple-01-simple-0- 38m	0 ClusterIP	None	<none></none>	9000/TCP,8123/TCP	,9009/TCP
clickhouse-pv-simple clickhouse-simple-01	ClusterIP ClusterIP	None None	<none> <none></none></none>	8123/TCP,9000/TCP 8123/TCP,9000/TCP	35m 3d7h

Step 3 Connect to the ClickHouse database.

kubectl -n *test-clickhouse-operator* exec -ti chi-ck-elb-ck-elb-0-0-0 -- clickhouse-client

If the following information is displayed, you have successfully connected to the ClickHouse database: Enter **exit** and press **Enter** to exit the ClickHouse database.

ClickHouse client version 23.8.16.16 (official build). Connecting to localhost:9000 as user default. Connected to ClickHouse server version 23.8.16 revision 54465.

Warnings:

* Linux transparent hugepages are set to "always". Check /sys/kernel/mm/transparent_hugepage/enabled

chi-ck-elb-ck-elb-0-0-0.chi-ck-elb-ck-elb-0-0.test-clickhouse-operator.svc.cluster.local :)

Step 4 Clear ClickHouse cluster resources.

Delete the ClickHouse cluster (the one that is associated with the LoadBalancer Service).

kubectl delete -f *elb.yaml* -n *test-clickhouse-operator*

Information similar to the following is displayed:

clickhouseinstallation.clickhouse.altinity.com "ck-elb1" deleted

----End

Follow-up Procedure: Clearing Other ClickHouse Resources

1. Delete the **test-clickhouse-operator** namespace. kubectl delete namespace *test-clickhouse-operator*

Information similar to the following is displayed:

namespace "test-clickhouse-operator" deleted

2. Delete the ClickHouse Operator: kubectl delete -f clickhouse-operator-install-bundle.yaml

Information similar to the following is displayed:

customresourcedefinition.apiextensions.k8s.io "clickhouseinstallations.clickhouse.altinity.com" deleted customresourcedefinition.apiextensions.k8s.io "clickhouseinstallationtemplates.clickhouse.altinity.com" deleted

customresourcedefinition.apiextensions.k8s.io

"clickhouseoperatorconfigurations.clickhouse.altinity.com" deleted

customresourcedefinition.apiextensions.k8s.io "clickhousekeeperinstallations.clickhouse-

15.6 Deploying and Using Spark in a CCE Cluster

15.6.1 Installing Spark

Prerequisites

A Linux server that can access the public network is available. The recommended node specifications are 4 vCPUs and 8 GiB memory or higher.

Configuring the JDK

This section uses CentOS as an example to describe how to install JDK 1.8.

- **Step 1** Obtain the available version. yum -y list java*
- Step 2 Install JDK 1.8.

yum install -y java-1.8.0-openjdk java-1.8.0-openjdk-devel

Step 3 Check the version after the installation.

java -version openjdk version "1.8.0_382" OpenJDK Runtime Environment (build 1.8.0_382-b05) OpenJDK 64-Bit Server VM (build 25.382-b05, mixed mode)

- Step 4 Add environment variables.
 - 1. Linux environment variables are configured in the **/etc/profile** file. vim /etc/profile
 - 2. In the editing mode, add the following content to the end of the file: JAVA_HOME=/usr/lib/jvm/java-1.8.0-openjdk-1.8.0.382.b05-1.el7_9.x86_64 PATH=\$PATH:\$JAVA_HOME/bin

keeper.altinity.com" deleted

CLASSPATH=.:\$JAVA_HOME/lib/dt.jar:\$JAVA_HOME/lib/tools.jar export JAVA_HOME PATH CLASSPATH

- 3. Save and close the **profile** file. Run the following command for the modification to take effect: source /etc/profile
- 4. Check the JDK environment variables. echo \$JAVA_HOME echo \$PATH echo \$CLASSPATH

----End

Obtaining the Spark Package

OBS matches Hadoop 2.8.3 and 3.1.1. Hadoop 3.1.1 is used in this example.

- **Step 1** Download **Spark** v3.1.1. If Git is not installed, run **yum install git** to install it. git clone -b v3.1.1 https://github.com/apache/spark.git
- **Step 2** Modify the **/dev/make-distribution.sh** file and specify the Spark version so that the check can be skipped during compilation.
 - Search for the line where VERSION resides and check the number of the line where the version number is located. cat ./spark/dev/make-distribution.sh |grep -n '^VERSION=' -A18
 - Comment out the content displayed from lines 129 to 147 and specify the version.
 sed -i '129,147s/^/#/g' ./spark/dev/make-distribution.sh
 sed -i '148a
 VERSION=3.1.3\nSCALA_VERSION=2.12\nSPARK_HADOOP_VERSION=3.1.1\nSPARK_HIVE=1' ./spark/dev/make-distribution.sh

Step 3 Download dependencies.

wget https://archive.apache.org/dist//maven/maven-3/3.6.3/binaries/apache-maven-3.6.3-bin.tar.gz tar -zxvf apache-maven-3.6.3-bin.tar.gz && mv apache-maven-3.6.3 ./spark/build

- **Step 4** Run the following command to perform compilation: ./spark/dev/make-distribution.sh --name hadoop3.1 --tgz -Pkubernetes -Pyarn -Dhadoop.version=3.1.1
- **Step 5** Wait for the compilation to complete. After the compilation, the software package is named **spark-3.1.3-bin-hadoop3.1**.tgz.

----End

Configuring the Runtime Environment for Spark

To simplify the operation, use the **root** user to place the compiled package **spark-3.1.3-bin-hadoop3.1.tgz** in the **/root** directory on the operation node.

Step 1 Move the software package to the /root directory. mv ./spark/spark-3.1.3-bin-hadoop3.1.tgz /root

Step 2 Run the following command to install Spark:

tar -zxvf spark-3.1.3-bin-hadoop3.1.tgz mv spark-3.1.3-bin-hadoop3.1 spark-obs cat >> ~/.bashrc <<EOF PATH=/root/spark-obs/bin:\\$PATH PATH=/root/spark-obs/sbin:\\$PATH export SPARK_HOME=/root/spark-obs EOF source ~/.bashrc

Step 3 Run the following command where binary spark-submit is used to check the Spark version: spark-submit --version

----End

Interconnecting Spark with OBS

Step 1 Obtain Huawei Cloud OBS JAR. The hadoop-huaweicloud-3.1.1-hw-45.jar package is used, which can be obtained from https://github.com/huaweicloud/ obsa-hdfs/tree/master/release.

wget https://github.com/huaweicloud/obsa-hdfs/releases/download/v45/hadoop-huaweicloud-3.1.1hw-45.jar

- **Step 2** Copy the package to the corresponding directory. cp hadoop-huaweicloud-3.1.1-hw-45.jar /root/spark-obs/jars/
- **Step 3** Modify Spark configuration items. To interconnect Spark with OBS, add ConfigMaps for Spark as follows:
 - 1. Obtain the AK/SK. For details, see Access Keys.
 - 2. Change the values of **AK_OF_YOUR_ACCOUNT**, **SK_OF_YOUR_ACCOUNT**, and **OBS_ENDPOINT** to the actual values.
 - AK_OF_YOUR_ACCOUNT: indicates the AK obtained in the previous step.
 - **SK_OF_YOUR_ACCOUNT**: indicates the SK obtained in the previous step.
 - OBS_ENDPOINT: indicates the OBS endpoint. It can be obtained in Regions and Endpoints.

cp ~/spark-obs/conf/spark-defaults.conf.template ~/spark-obs/conf/spark-defaults.conf

cat >> ~/spark-obs/conf/spark-defaults.conf <<EOF spark.hadoop.fs.obs.readahead.inputstream.enabled=true spark.hadoop.fs.obs.buffer.max.range=6291456 spark.hadoop.fs.obs.buffer.part.size=2097152 spark.hadoop.fs.obs.threads.read.core=500 spark.hadoop.fs.obs.threads.read.max=1000 spark.hadoop.fs.obs.write.buffer.size=8192 spark.hadoop.fs.obs.read.buffer.size=8192 spark.hadoop.fs.obs.connection.maximum=1000 spark.hadoop.fs.obs.access.key=AK_OF_YOUR_ACCOUNT spark.hadoop.fs.obs.secret.key=SK_OF_YOUR_ACCOUNT spark.hadoop.fs.obs.endpoint=OBS ENDPOINT spark.hadoop.fs.obs.buffer.dir=/root/hadoop-obs/obs-cache spark.hadoop.fs.obs.impl=org.apache.hadoop.fs.obs.OBSFileSystem spark.hadoop.fs.obs.connection.ssl.enabled=false spark.hadoop.fs.obs.fast.upload=true spark.hadoop.fs.obs.socket.send.buffer=65536 spark.hadoop.fs.obs.socket.recv.buffer=65536 spark.hadoop.fs.obs.max.total.tasks=20 spark.hadoop.fs.obs.threads.max=20 spark.kubernetes.container.image.pullSecrets=default-secret FOF

----End

Pushing an Image to SWR

To run Spark tasks in Kubernetes, build a Spark container image of the same version and upload it to SWR. A **Dockerfile** file has been generated during compilation. Use this file to create an image and push it to SWR.

Step 1 Create an image. cd ~/spark-obs

docker build -t spark:3.1.3-obs --build-arg spark_uid=0 -f kubernetes/dockerfiles/spark/Dockerfile .

- Step 2 Upload the image.
 - 1. (Optional) Log in to the SWR console, choose **Organizations** in the navigation pane, and click **Create Organization** in the upper right corner. Skip this step if you already have an organization.
 - 2. Choose **My Images** in the navigation pane and click **Upload Through Client**. On the page displayed, click **Generate a temporary login command** and

click \square to copy the command.

- 3. Run the login command copied in the previous step on the node. If the login is successful, the message "Login Succeeded" is displayed.
- 4. Log in to the node where the image is created and run the login command. Docker tag [{ Image name }:{Version name }] swr. ap-southeast-1.myhuaweicloud.com/{ Organization name }/{Image name }:{Version name } docker push swr.ap-southeast-1.myhuaweicloud.com/{Organization name }/{Image name }:{Version name }

Record the image access address for later use.

For example, record the IP address as **swr.apsoutheast-1.myhuaweicloud.com/dev-container/spark:3.1.3-obs**.

----End

Configuring Spark History Server

Step 1 Modify the **~/spark-obs/conf/spark-defaults.conf** file, enable Spark event logging, and configure the OBS bucket name and directory.

cat >> ~/spark-obs/conf/spark-defaults.conf <<EOF spark.eventLog.enabled=true spark.eventLog.dir=*obs://{bucket-name}/{log-dir}/* EOF

- spark.eventLog.enabled: indicates that Spark event logging is enabled if it is set to true.
- spark.eventLog.dir: indicates the OBS bucket name and path. The bucket is named in the format of obs://{bucket-name}/{log-dir}, for example, obs:// spark-sh1/history-obs/. Ensure that the OBS bucket name and directory are correct.
- **Step 2** Modify the **~/spark-obs/conf/spark-env.sh** file. If the file does not exist, run the command to copy the template as a file:

cp ~/spark-obs/conf/spark-env.sh.template ~/spark-obs/conf/spark-env.sh

```
cat >> ~/spark-obs/conf/spark-env.sh <<EOF
SPARK_HISTORY_OPTS="-Dspark.history.fs.logDirectory=obs://{bucket-name}/{log-dir}/"
EOF
```

The OBS address must be the same as that in **spark-default.conf** in the previous step.

Step 3 Start Spark History Server.

start-history-server.sh

Information similar to the following is displayed:

starting org.apache.spark.deploy.history.HistoryServer, logging to /root/spark-obs/logs/spark-root-org.apache.spark.deploy.history.HistoryServer-1-spark-sh1.out

Step 4 Access the server through port 18080 on the node.

Spark 24.5 History Server						
Event log directory: obs://spark-sh1/history-obs/						
Last updated: 2020-09-04 17:22:01						
Client local time zone: Asia/Shanghai						
						Search:
App ID	App Name	Started	Completed	Duration	Spark User	Last Updated
spark-900f4f77737742bc9dd3d348cf91cd45	Spark Pi	2020-09-04 17:17:54	2020-09-04 17:18:17	22 s	root	2020-09-04 17:18:17
Showing 1 to 1 of 1 entries						

To stop the server, run the following command:

stop-history-server.sh

----End

15.6.2 Using Spark

You can use Spark's Kubernetes scheduler spark-submit to submit Spark applications to Kubernetes clusters. For details, see **Running Spark on Kubernetes**. The submission mechanism works as follows:

- Create a pod to run the Spark driver.
- The driver creates pods for executing the programs and establishes a connection with these pods.
- After the application is complete, the pods that execute the programs are terminated and cleaned up, but the driver pod exists and remains in the completed state until the garbage is collected or it is manually cleaned up. In the completed state, the driver pod does not use any computing or memory resources.



Figure 15-7 Submission mechanism

Running SparkPi on CCE

Step 1 Install kubectl on the node where Spark is running. For details, see Connecting to a Cluster Using kubectl.

Step 2 Run the following command to grant the cluster-level permissions:

Create a service account. kubectl create serviceaccount spark # Bind the ClusterRole **spark-role** to the service account created in the previous step, specify the **default** namespace, and grant the ClusterRole permission to edit resources. kubectl create clusterrolebinding spark-role --clusterrole=edit --serviceaccount=default:spark -namespace=default

Step 3 Submit a SparkPi job to CCE. The following shows an example:

spark-submit \

--master k8s://https://**.**.**.5443 \

- --deploy-mode cluster \
- --name spark-pi \

--class org.apache.spark.examples.SparkPi \

--conf spark.executor.instances=2 \

--conf spark.kubernetes.authenticate.driver.serviceAccountName=spark \

--conf spark.kubernetes.container.image=swr.ap-southeast-1.myhuaweicloud.com/dev-container/ spark:3.1.3-obs \

local:///root/spark-obs/examples/jars/spark-examples_2.12-3.1.1.jar

Parameters:

- --master: indicates the API Server of the cluster. https://**.**.5443 is the address of the master node used in ~/.kube/config. It can be obtained from kubectl cluster-info.
- --deploy-mode:
 - **cluster**: a mode in which the driver is deployed on the worker nodes.
 - client: (default value) a mode in which the driver is deployed locally as an external client.
- --name: indicates the name of a job. It is used to name the pods in the cluster.
- --class: indicates the applications, for example, org.apache.spark.examples.SparkPi.
- --conf: indicates the Spark's configuration parameters. It is in the key-value pair format. All parameters that can be specified using --conf are read from the ~/spark-obs/conf/spark-defaults.conf file by default. Therefore, the general configuration can be written to be the default settings, the same way as Interconnecting Spark with OBS.
 - spark.executor.instances: indicates the number of pods for executing programs.
 - spark.kubernetes.authenticate.driver.serviceAccountName: indicates the driver's cluster-level permissions. Select the service account created in Step 2.
 - **spark.kubernetes.container.image**: indicates the image path of the image pushed to SWR in **Pushing an Image to SWR**.
- **local**: indicates the path to the JAR packages stored in the local files. In this example, a local file is used to store the JAR packages. The value of this parameter can be **file**, **http**, or **local**. For details, see the **Official Documentation**.

----End

Accessing OBS

Use **spark-submit** to deliver an HDFS job. Change the value of **obs://bucket-name/filename** at the end of the script to the actual file name of the tenant.

spark-submit \

--master k8s://https://**.**.**.5443 \

--deploy-mode cluster \

--name spark-hdfs-test \

--class org.apache.spark.examples.HdfsTest \

--conf spark.kubernetes.authenticate.driver.serviceAccountName=spark \

--conf spark.kubernetes.container.image=swr.ap-southeast-1.myhuaweicloud.com/dev-container/ spark:3.1.3-obs \

local:///root/spark-obs/examples/jars/spark-examples_2.12-3.1.1.jar obs://bucket-name/filename

Support for Spark Shell Commands to Interact with Spark-Scala

spark-shell \

--conf spark.kubernetes.authenticate.driver.serviceAccountName=spark \ --conf spark.kubernetes.container.image=swr.ap-southeast-1.myhuaweicloud.com/dev-container/ spark:3.1.3-obs \ --master k8s://https://**.**.**:5443

Run the following commands to define the algorithms of Spark computing jobs linecount and wordcount:

def linecount(input:org.apache.spark.sql.Dataset[String]):Long=input.filter(line => line.length()>0).count()
def wordcount(input:org.apache.spark.sql.Dataset[String]):Long=input.flatMap(value => value.split("\\s
+")).groupByKey(value => value).count()

Run the following commands to define data sources:

var alluxio = spark.read.textFile("alluxio://alluxio-master:19998/sample-1g")
var obs = spark.read.textFile("obs://gene-container-gtest/sample-1g")
var hdfs = spark.read.textFile("hdfs://192.168.1.184:9000/user/hadoop/books/sample-1g")

Run the following command to start computing jobs:

spark.time(wordcount(obs))
spark.time(linecount(obs))