

# Cloud Container Engine

## User Guide

**Issue** 01  
**Date** 2024-11-11



**Copyright © Huawei Cloud Computing Technologies Co., Ltd. 2024. 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**



HUAWEI 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.

## **Huawei Cloud Computing Technologies Co., Ltd.**

Address: Huawei Cloud Data Center Jiaoxinggong Road  
Qianzhong Avenue  
Gui'an New District  
Gui Zhou 550029  
People's Republic of China

Website: <https://www.huaweicloud.com/intl/en-us/>

---

# Contents

---

<b>1 High-Risk Operations.....</b>	<b>1</b>
<b>2 Clusters.....</b>	<b>10</b>
2.1 Cluster Overview.....	10
2.1.1 Basic Cluster Information.....	10
2.1.2 Kubernetes Version Release Notes.....	11
2.1.2.1 Kubernetes 1.29 Release Notes.....	11
2.1.2.2 Kubernetes 1.28 Release Notes.....	16
2.1.2.3 Kubernetes 1.27 Release Notes.....	21
2.1.2.4 Kubernetes 1.25 Release Notes.....	27
2.1.2.5 Kubernetes 1.23 Release Notes.....	32
2.1.2.6 Kubernetes 1.21 (EOM) Release Notes.....	33
2.1.2.7 Kubernetes 1.19 (EOM) Release Notes.....	34
2.1.2.8 Kubernetes 1.17 (EOM) Release Notes.....	36
2.1.2.9 Kubernetes 1.15 (EOM) Release Notes.....	37
2.1.2.10 Kubernetes 1.13 (EOM) Release Notes.....	38
2.1.2.11 Kubernetes 1.11 (EOM) Release Notes.....	39
2.1.2.12 Release Notes for Kubernetes 1.9 (EOM) and Earlier Versions.....	41
2.1.3 Patch Version Release Notes.....	46
2.2 Buying a Cluster.....	63
2.2.1 Comparison Between Cluster Types.....	63
2.2.2 Buying a CCE Standard/Turbo Cluster.....	65
2.2.3 Comparing iptables and IPVS.....	72
2.3 Connecting to a Cluster.....	73
2.3.1 Connecting to a Cluster Using kubectl.....	73
2.3.2 Accessing a Cluster Using an X.509 Certificate.....	76
2.3.3 Accessing a Cluster Using a Custom Domain Name.....	77
2.3.4 Revoking a Credential to Access the Corresponding Cluster.....	79
2.4 Managing a Cluster.....	81
2.4.1 Modifying Cluster Configurations.....	81
2.4.2 Enabling Overload Control for a Cluster.....	95
2.4.3 Changing Cluster Scale.....	96
2.4.4 Changing the Default Security Group of a Node.....	97
2.4.5 Deleting a Cluster.....	99

2.4.6 Hibernating or Waking Up a Cluster.....	103
2.4.7 Renewing a Yearly/Monthly Cluster.....	103
2.4.8 Changing the Billing Mode of a Cluster from Pay-per-Use to Yearly/Monthly.....	104
2.5 Upgrading a Cluster.....	105
2.5.1 Process and Method of Upgrading a Cluster.....	105
2.5.2 Before You Start.....	109
2.5.3 Performing Post-Upgrade Verification.....	124
2.5.3.1 Cluster Status Check.....	124
2.5.3.2 Node Status Check.....	125
2.5.3.3 Node Skipping Check.....	125
2.5.3.4 Service Check.....	125
2.5.3.5 New Node Check.....	126
2.5.3.6 New Pod Check.....	126
2.5.4 Migrating Services Across Clusters of Different Versions.....	128
2.5.5 Troubleshooting for Pre-upgrade Check Exceptions.....	129
2.5.5.1 Pre-upgrade Check.....	129
2.5.5.2 Node Restrictions.....	133
2.5.5.3 Upgrade Management.....	134
2.5.5.4 Add-ons.....	135
2.5.5.5 Helm Charts.....	136
2.5.5.6 SSH Connectivity of Master Nodes.....	136
2.5.5.7 Node Pools.....	136
2.5.5.8 Security Groups.....	137
2.5.5.9 Arm Node Restrictions.....	138
2.5.5.10 Residual Nodes.....	139
2.5.5.11 Discarded Kubernetes Resources.....	139
2.5.5.12 Compatibility Risks.....	140
2.5.5.13 CCE Agent Versions.....	143
2.5.5.14 Node CPU Usage.....	144
2.5.5.15 CRDs.....	144
2.5.5.16 Node Disks.....	144
2.5.5.17 Node DNS.....	145
2.5.5.18 Node Key Directory File Permissions.....	145
2.5.5.19 kubelet.....	146
2.5.5.20 Node Memory.....	146
2.5.5.21 Node Clock Synchronization Server.....	147
2.5.5.22 Node OS.....	147
2.5.5.23 Node CPU Cores.....	148
2.5.5.24 Node Python Commands.....	148
2.5.5.25 ASM Version.....	149
2.5.5.26 Node Readiness.....	149
2.5.5.27 Node journald.....	150

2.5.5.28 containerd.sock.....	150
2.5.5.29 Internal Error.....	150
2.5.5.30 Node Mount Points.....	151
2.5.5.31 Kubernetes Node Taints.....	152
2.5.5.32 Everest Restrictions.....	152
2.5.5.33 cce-hpa-controller Limitations.....	153
2.5.5.34 Enhanced CPU Policies.....	153
2.5.5.35 Health of Worker Node Components.....	154
2.5.5.36 Health of Master Node Components.....	154
2.5.5.37 Memory Resource Limit of Kubernetes Components.....	155
2.5.5.38 Discarded Kubernetes APIs.....	155
2.5.5.39 IPv6 Support in CCE Turbo Clusters.....	155
2.5.5.40 NetworkManager.....	156
2.5.5.41 Node ID File.....	156
2.5.5.42 Node Configuration Consistency.....	157
2.5.5.43 Node Configuration File.....	159
2.5.5.44 CoreDNS Configuration Consistency.....	159
2.5.5.45 sudo.....	161
2.5.5.46 Key Node Commands.....	161
2.5.5.47 Mounting of a Sock File on a Node.....	162
2.5.5.48 HTTPS Load Balancer Certificate Consistency.....	163
2.5.5.49 Node Mounting.....	164
2.5.5.50 Login Permissions of User <b>paas</b> on a Node.....	165
2.5.5.51 Private IPv4 Addresses of Load Balancers.....	166
2.5.5.52 Historical Upgrade Records.....	167
2.5.5.53 CIDR Block of the Cluster Management Plane.....	167
2.5.5.54 GPU Add-on.....	167
2.5.5.55 Nodes' System Parameters.....	168
2.5.5.56 Residual Package Version Data.....	168
2.5.5.57 Node Commands.....	169
2.5.5.58 Node Swap.....	169
2.5.5.59 nginx-ingress Upgrade.....	170
2.5.5.60 ELB Listener Access Control.....	172
2.5.5.61 Master Node Flavor.....	172
2.5.5.62 Subnet Quota of Master Nodes.....	172
2.5.5.63 Node Runtime.....	173
2.5.5.64 Node Pool Runtime.....	173
2.5.5.65 Number of Node Images.....	173
2.5.5.66 OpenKruise Compatibility Check.....	174
2.5.5.67 Compatibility Check of Secret Encryption.....	174
2.5.5.68 Compatibility Between the Ubuntu Kernel and GPU Driver.....	174
2.5.5.69 Drainage Tasks.....	175

2.5.5.70 Image Layers on a Node.....	176
2.5.5.71 Cluster Rolling Upgrade.....	176
2.5.5.72 Rotation Certificates.....	176
2.5.5.73 Ingress and ELB Configuration Consistency.....	177
<b>3 Nodes.....</b>	<b>178</b>
3.1 Node Overview.....	178
3.2 Container Engines.....	180
3.3 Node OSs.....	183
3.4 Creating a Node.....	189
3.5 Accepting Nodes for Management.....	195
3.6 Management Nodes.....	199
3.6.1 Managing Node Labels.....	199
3.6.2 Managing Node Taints.....	201
3.6.3 Resetting a Node.....	205
3.6.4 Removing a Node.....	208
3.6.5 Synchronizing the Data of Cloud Servers.....	209
3.6.6 Draining a Node.....	210
3.6.7 Deleting or Unsubscribing from a Node.....	215
3.6.8 Changing the Billing Mode of a Node to Yearly/Monthly.....	216
3.6.9 Modifying the Auto-Renewal Configuration of a Yearly/Monthly Node.....	217
3.6.10 Stopping a Node.....	218
3.7 Node O&M.....	219
3.7.1 Node Resource Reservation Policy.....	219
3.7.2 Space Allocation of a Data Disk.....	222
3.7.3 Maximum Number of Pods That Can Be Created on a Node.....	228
3.7.4 Differences in kubelet and Runtime Component Configurations Between CCE and the Native Community.....	230
3.7.5 Migrating Nodes from Docker to containerd.....	232
3.7.6 Optimizing Node System Parameters.....	234
3.7.6.1 Optimizable Node System Parameters.....	234
3.7.6.2 Changing the RuntimeMaxUse of the Memory Used by the Log Cache on a Node.....	241
3.7.6.3 Changing the Maximum Number of File Handles.....	242
3.7.6.4 Modifying Node Kernel Parameters.....	246
3.7.6.5 Changing Process ID Limits (kernel.pid_max).....	252
3.7.7 Configuring Node Fault Detection Policies.....	255
<b>4 Node Pools.....</b>	<b>268</b>
4.1 Node Pool Overview.....	268
4.2 Upgrading to New Node Pools.....	271
4.3 Creating a Node Pool.....	273
4.4 Scaling a Node Pool.....	278
4.5 Managing a Node Pool.....	279
4.5.1 Updating a Node Pool.....	279

4.5.2 Updating an AS Configuration.....	283
4.5.3 Modifying Node Pool Configurations.....	286
4.5.4 Accepting Nodes in a Node Pool.....	303
4.5.5 Copying a Node Pool.....	304
4.5.6 Synchronizing Node Pools.....	305
4.5.7 Upgrading an OS.....	306
4.5.8 Migrating a Node.....	308
4.5.9 Deleting a Node Pool.....	309
<b>5 Workloads.....</b>	<b>310</b>
5.1 Overview.....	310
5.2 Creating a Workload.....	314
5.2.1 Creating a Deployment.....	314
5.2.2 Creating a StatefulSet.....	321
5.2.3 Creating a DaemonSet.....	328
5.2.4 Creating a Job.....	333
5.2.5 Creating a Cron Job.....	340
5.3 Configuring a Workload.....	345
5.3.1 Configuring Time Zone Synchronization.....	345
5.3.2 Configuring an Image Pull Policy.....	346
5.3.3 Using Third-Party Images.....	347
5.3.4 Configuring Container Specifications.....	348
5.3.5 Configuring Container Lifecycle Parameters.....	352
5.3.6 Configuring Container Health Check.....	355
5.3.7 Configuring Environment Variables.....	359
5.3.8 Configuring Workload Upgrade Policies.....	362
5.3.9 Configuring Tolerance Policies.....	365
5.3.10 Configuring Labels and Annotations.....	367
5.4 Scheduling a Workload.....	369
5.4.1 Overview.....	369
5.4.2 Configuring Specified Node Scheduling (nodeSelector).....	372
5.4.3 Configuring Node Affinity Scheduling (nodeAffinity).....	373
5.4.4 Configuring Workload Affinity or Anti-affinity Scheduling (podAffinity or podAntiAffinity).....	378
5.5 Logging In to a Container.....	386
5.6 Managing Workloads.....	387
5.7 Pod Security.....	393
5.7.1 Configuring a Pod Security Policy.....	393
5.7.2 Configuring Pod Security Admission.....	397
<b>6 Scheduling.....</b>	<b>401</b>
6.1 Overview.....	401
6.2 CPU Scheduling.....	403
6.2.1 CPU Policy.....	403
6.2.2 Enhanced CPU Policy.....	405

6.3 GPU Scheduling.....	408
6.3.1 Default GPU Scheduling in Kubernetes.....	408
6.3.2 GPU Virtualization.....	410
6.3.2.1 Overview.....	410
6.3.2.2 Preparing xGPU Resources.....	411
6.3.2.3 Using GPU Virtualization.....	414
6.3.2.4 Supporting Kubernetes' Default GPU Scheduling.....	419
6.3.3 Monitoring GPU Metrics.....	421
6.3.4 Configuring Workload Scaling Based on GPU Monitoring Metrics.....	427
6.4 NPU Scheduling.....	430
6.5 Volcano Scheduling.....	432
6.5.1 Scheduling Workloads.....	432
6.5.2 Resource Usage-based Scheduling.....	434
6.5.2.1 Bin Packing.....	434
6.5.2.2 Descheduling.....	437
6.5.2.3 Node Pool Affinity.....	447
6.5.2.4 Load-aware Scheduling.....	450
6.5.2.5 Configuration Cases for Resource Usage-based Scheduling.....	453
6.5.3 Priority-based Scheduling.....	457
6.5.3.1 Priority-based Scheduling.....	457
6.5.4 AI Performance-based Scheduling.....	461
6.5.4.1 DRF.....	461
6.5.4.2 Gang.....	463
6.5.5 NUMA Affinity Scheduling.....	466
6.5.6 Application Scaling Priority Policies.....	475
6.6 Cloud Native Hybrid Deployment.....	484
6.6.1 Overview.....	484
6.6.2 Dynamic Resource Oversubscription.....	488
6.6.3 CPU Burst.....	499
6.6.4 Guaranteed Egress Network Bandwidth.....	503
<b>7 Network.....</b>	<b>508</b>
7.1 Overview.....	508
7.2 Container Network.....	511
7.2.1 Overview.....	511
7.2.2 Cloud Native Network 2.0 Settings.....	515
7.2.2.1 Cloud Native 2.0 Network Model.....	515
7.2.2.2 Configuring Pod Subnets of a Cluster.....	519
7.2.2.3 Binding a Security Group to a Workload Using a Security Group Policy.....	520
7.2.2.4 Binding a Subnet and Security Group to a Namespace or Workload Using a Container Network Configuration.....	523
7.2.2.5 Configuring Shared Bandwidth for a Pod with IPv6 Dual-Stack ENIs.....	534
7.2.3 VPC Network Settings.....	536



7.2.3.1 VPC Network Model.....	536
7.2.3.2 Adding a Container CIDR Block for a Cluster.....	541
7.2.4 Tunnel Network Settings.....	542
7.2.4.1 Tunnel Network Model.....	542
7.2.5 Pod Network Settings.....	547
7.2.5.1 Configuring hostNetwork for Pods.....	547
7.2.5.2 Configuring QoS for a Pod.....	548
7.2.5.3 Configuring Network Policies to Restrict Pod Access.....	550
7.3 Service.....	553
7.3.1 Overview.....	553
7.3.2 ClusterIP.....	564
7.3.3 NodePort.....	567
7.3.4 LoadBalancer.....	571
7.3.4.1 Creating a LoadBalancer Service.....	571
7.3.4.2 Configuring LoadBalancer Services Using Annotations.....	588
7.3.4.3 Configuring HTTP/HTTPS for a LoadBalancer Service.....	601
7.3.4.4 Configuring SNI for a LoadBalancer Service.....	603
7.3.4.5 Configuring HTTP/2 for a LoadBalancer Service.....	606
7.3.4.6 Configuring Timeout for a LoadBalancer Service.....	609
7.3.4.7 Configuring Health Check on Multiple Ports of a LoadBalancer Service.....	611
7.3.4.8 Configuring Passthrough Networking for a LoadBalancer Service.....	613
7.3.4.9 Setting the Pod Ready Status Through the ELB Health Check.....	616
7.3.5 Headless Services.....	618
7.4 Ingresses.....	619
7.4.1 Overview.....	619
7.4.2 LoadBalancer Ingresses.....	623
7.4.2.1 Creating a LoadBalancer Ingress on the Console.....	623
7.4.2.2 Creating a LoadBalancer Ingress Using kubectl.....	630
7.4.2.3 Annotations for Configuring LoadBalancer Ingresses.....	641
7.4.2.4 Advanced Setting Examples of LoadBalancer Ingresses.....	649
7.4.2.4.1 Configuring an HTTPS Certificate for a LoadBalancer Ingress.....	649
7.4.2.4.2 Configuring SNI for a LoadBalancer Ingress.....	661
7.4.2.4.3 Configuring Multiple Forwarding Policies for a LoadBalancer Ingress.....	669
7.4.2.4.4 Configuring HTTP/2 for a LoadBalancer Ingress.....	674
7.4.2.4.5 Configuring HTTPS Backend Services for a LoadBalancer Ingress.....	677
7.4.2.4.6 Configuring Timeout for a LoadBalancer Ingress.....	681
7.4.2.4.7 Configuring a Slow Start for a LoadBalancer Ingress.....	686
7.4.2.4.8 Configuring a Range of Listening Ports for a LoadBalancer Ingress.....	688
7.4.3 Nginx Ingresses.....	690
7.4.3.1 Creating an Nginx Ingress on the Console.....	690
7.4.3.2 Creating an Nginx Ingress Using kubectl.....	692
7.4.3.3 Annotations for Configuring Nginx Ingresses.....	697

7.4.3.4 Advanced Setting Examples of Nginx Ingresses.....	705
7.4.3.4.1 Configuring an HTTPS Certificate for an Nginx Ingress.....	705
7.4.3.4.2 Configuring Redirection Rules for an Nginx Ingress.....	707
7.4.3.4.3 Configuring URL Rewriting Rules for an Nginx Ingress.....	713
7.4.3.4.4 Configuring HTTPS Backend Services for an Nginx Ingress.....	716
7.4.3.4.5 Configuring Consistent Hashing for Load Balancing of an Nginx Ingress.....	717
7.5 DNS.....	719
7.5.1 Overview.....	719
7.5.2 DNS Configuration.....	722
7.5.3 Using CoreDNS for Custom Domain Name Resolution.....	729
7.5.4 Using NodeLocal DNSCache to Improve DNS Performance.....	734
7.6 Configuring Intra-VPC Access.....	739
7.7 Accessing the Internet from a Container.....	741
<b>8 Storage.....</b>	<b>744</b>
8.1 Overview.....	744
8.2 Storage Basics.....	750
8.3 Elastic Volume Service.....	755
8.3.1 Overview.....	755
8.3.2 Using an Existing EVS Disk Through a Static PV.....	756
8.3.3 Using an EVS Disk Through a Dynamic PV.....	766
8.3.4 Dynamically Mounting an EVS Disk to a StatefulSet.....	774
8.3.5 Snapshots and Backups.....	780
8.4 Scalable File Service.....	782
8.4.1 Overview.....	782
8.4.2 Using an Existing SFS File System Through a Static PV.....	784
8.4.3 Using an SFS File System Through a Dynamic PV.....	793
8.4.4 Configuring SFS Volume Mount Options.....	798
8.5 SFS Turbo.....	802
8.5.1 Overview.....	802
8.5.2 Using an Existing SFS Turbo File System Through a Static PV.....	803
8.5.3 Configuring SFS Turbo Mount Options.....	816
8.5.4 Dynamically Creating an SFS Turbo Subdirectory Using StorageClass.....	818
8.6 Object Storage Service.....	823
8.6.1 Overview.....	823
8.6.2 Using an Existing OBS Bucket Through a Static PV.....	825
8.6.3 Using an OBS Bucket Through a Dynamic PV.....	836
8.6.4 Configuring OBS Mount Options.....	844
8.6.5 Using a Custom Access Key (AK/SK) to Mount an OBS Volume.....	847
8.7 Local PVs.....	852
8.7.1 Overview.....	852
8.7.2 Importing a PV to a Storage Pool.....	853
8.7.3 Using a Local PV Through a Dynamic PV.....	854

8.7.4 Dynamically Mounting a Local PV to a StatefulSet.....	860
8.8 Ephemeral Volumes.....	865
8.8.1 Overview.....	865
8.8.2 Importing an EV to a Storage Pool.....	866
8.8.3 Using a Local EV.....	868
8.8.4 Using a Temporary Path.....	870
8.9 hostPath.....	872
8.10 StorageClass.....	875
<b>9 Observability.....</b>	<b>885</b>
9.1 Logging.....	885
9.1.1 Overview.....	885
9.1.2 Collecting Container Logs.....	885
9.1.2.1 Collecting Container Logs Using ICAgent.....	885
9.1.3 Collecting Control Plane Component Logs.....	890
9.1.4 Collecting Kubernetes Audit Logs.....	893
9.2 Best Practices.....	896
9.2.1 Monitoring Custom Metrics Using Cloud Native Cluster Monitoring.....	896
9.2.2 Monitoring Custom Metrics on AOM.....	905
9.2.3 Monitoring Metrics of Master Node Components Using Prometheus.....	909
<b>10 Auto Scaling.....</b>	<b>914</b>
10.1 Overview.....	914
10.2 Scaling a Workload.....	915
10.2.1 Workload Scaling Rules.....	916
10.2.2 Creating an HPA Policy.....	917
10.2.3 Creating a Scheduled CronHPA Policy.....	920
10.2.4 Creating a CustomedHPA Policy.....	933
10.2.5 Managing Workload Scaling Policies.....	936
10.3 Scaling a Node.....	937
10.3.1 Node Scaling Rules.....	937
10.3.2 Creating a Node Scaling Policy.....	944
10.3.3 Managing Node Scaling Policies.....	953
10.4 Using HPA and CA for Auto Scaling of Workloads and Nodes.....	954
<b>11 Namespaces.....</b>	<b>962</b>
11.1 Creating a Namespace.....	962
11.2 Managing Namespaces.....	964
11.3 Configuring Resource Quotas.....	966
<b>12 ConfigMaps and Secrets.....</b>	<b>969</b>
12.1 Creating a ConfigMap.....	969
12.2 Using a ConfigMap.....	971
12.3 Creating a Secret.....	979
12.4 Using a Secret.....	983

12.5 Cluster Secrets.....	989
<b>13 Add-ons.....</b>	<b>991</b>
13.1 Overview.....	991
13.2 Scheduling and Elasticity Add-ons.....	996
13.2.1 Volcano Scheduler.....	996
13.2.2 CCE Cluster Autoscaler.....	1020
13.2.3 CCE Advanced HPA.....	1036
13.3 Cloud Native Observability Add-ons.....	1041
13.3.1 Cloud Native Cluster Monitoring.....	1041
13.3.2 CCE Node Problem Detector.....	1051
13.3.3 CCE Network Metrics Exporter.....	1068
13.3.4 Kubernetes Metrics Server.....	1083
13.3.5 Prometheus .....	1087
13.4 Cloud Native Heterogeneous Computing Add-ons.....	1091
13.4.1 CCE AI Suite (NVIDIA GPU).....	1091
13.4.2 CCE AI Suite (Ascend NPU).....	1099
13.5 Container Network Add-ons.....	1104
13.5.1 CoreDNS.....	1104
13.5.2 NGINX Ingress Controller.....	1115
13.5.3 NodeLocal DNSCache.....	1124
13.6 Container Storage Add-ons.....	1130
13.6.1 CCE Container Storage (Everest).....	1130
13.7 Other Add-ons.....	1140
13.7.1 Kubernetes Dashboard.....	1140
<b>14 Helm Chart.....</b>	<b>1146</b>
14.1 Overview of a Chart.....	1146
14.2 Deploying an Application from a Chart.....	1147
14.3 Differences Between Helm v2 and Helm v3 and Adaptation Solutions.....	1151
14.4 Deploying an Application Through the Helm v2 Client.....	1152
14.5 Deploying an Application Through the Helm v3 Client.....	1155
14.6 Converting a Release from Helm v2 to v3.....	1157
<b>15 Permissions.....</b>	<b>1161</b>
15.1 Permissions Overview.....	1161
15.2 Granting Cluster Permissions to an IAM User.....	1163
15.3 Namespace Permissions (Kubernetes RBAC-based).....	1168
15.4 Example: Designing and Configuring Permissions for Users in a Department.....	1176
15.5 Permission Dependency of the CCE Console.....	1179
15.6 Service Account Token Security Improvement.....	1183
<b>16 Settings.....</b>	<b>1185</b>
16.1 Dashboard.....	1185
16.2 Cluster Access.....	1187

16.3 Network.....	1190
16.4 Scheduling.....	1196
16.5 Auto Scaling.....	1201
16.6 Kubernetes.....	1203
16.7 Heterogeneous Resources.....	1217
<b>17 Old Console.....</b>	<b>1218</b>
17.1 What Is Cloud Container Engine?.....	1218
17.2 High-Risk Operations and Solutions.....	1219
17.3 Clusters.....	1223
17.3.1 Cluster Overview.....	1223
17.3.2 Buying a CCE Cluster.....	1226
17.3.3 Using kubectl to Run a Cluster.....	1241
17.3.3.1 Connecting to a Cluster Using kubectl.....	1242
17.3.3.2 Common kubectl Commands.....	1245
17.3.3.3 kubectl Usage Guide.....	1251
17.3.4 Setting Cluster Auto Scaling.....	1252
17.3.5 Upgrading a Cluster.....	1255
17.3.5.1 Overview.....	1255
17.3.5.2 Before You Start.....	1258
17.3.5.3 Performing Replace/Rolling Upgrade (v1.13 and Earlier).....	1261
17.3.5.4 Performing In-place Upgrade (v1.15 and Later).....	1265
17.3.5.5 Migrating Services Across Clusters of Different Versions.....	1269
17.3.5.6 CCE Kubernetes Release Notes.....	1270
17.3.6 Managing a Cluster.....	1271
17.3.6.1 Deleting a Cluster (Pay-per-Use).....	1271
17.3.6.2 Deleting, Unsubscribing From, or Releasing a Yearly/Monthly-Billed Cluster.....	1273
17.3.6.3 Renewing a Yearly/Monthly-Billed Cluster.....	1276
17.3.6.4 Hibernating and Waking Up a Cluster (Pay-per-Use).....	1278
17.3.6.5 Changing the Billing Mode from Pay-per-Use to Yearly/Monthly.....	1279
17.3.6.6 Configuring Kubernetes Parameters.....	1280
17.3.7 Obtaining a Cluster Certificate.....	1284
17.3.8 Changing Cluster Scale.....	1284
17.3.9 Controlling Cluster Permissions.....	1286
17.3.10 Cluster Parameters.....	1288
17.3.10.1 Maximum Number of Pods That Can Be Created on a Node.....	1288
17.3.10.2 Comparing iptables and IPVS.....	1289
17.3.10.3 CPU Policy.....	1290
17.4 Nodes.....	1290
17.4.1 Overview.....	1291
17.4.2 Buying a Node.....	1292
17.4.3 Accepting ECSs as Nodes into a Cluster.....	1300
17.4.4 Removing a Node.....	1302

17.4.5 Logging In to a Node.....	1304
17.4.6 Managing Node Labels.....	1304
17.4.7 Synchronizing Node Data.....	1308
17.4.8 Configuring Node Scheduling (Tainting).....	1309
17.4.9 Resetting a Node.....	1311
17.4.10 Deleting a Node.....	1313
17.4.11 Stopping a Node.....	1314
17.4.12 Performing Rolling Upgrade for Nodes.....	1315
17.4.13 Formula for Calculating the Reserved Resources of a Node.....	1319
17.4.14 Creating a Linux LVM Disk Partition for Docker.....	1321
17.4.15 Data Disk Space Allocation.....	1323
17.4.16 Adding a Second Data Disk to a Node in a CCE Cluster.....	1325
17.5 Node Pools.....	1327
17.5.1 Node Pool Overview.....	1327
17.5.2 Creating a Node Pool.....	1331
17.5.3 Managing a Node Pool.....	1339
17.6 Workloads.....	1345
17.6.1 Overview.....	1345
17.6.2 Creating a Deployment.....	1349
17.6.3 Creating a StatefulSet.....	1358
17.6.4 Creating a DaemonSet.....	1366
17.6.5 Creating a Job.....	1371
17.6.6 Creating a Cron Job.....	1377
17.6.7 Managing Pods.....	1383
17.6.8 GPU Scheduling.....	1385
17.6.9 NPU Scheduling.....	1387
17.6.10 Managing Workloads and Jobs.....	1389
17.6.11 Scaling a Workload.....	1397
17.6.12 Configuring a Container.....	1402
17.6.12.1 Using a Third-Party Image.....	1402
17.6.12.2 Setting Container Specifications.....	1404
17.6.12.3 Setting Container Lifecycle Parameters.....	1406
17.6.12.4 Setting Container Startup Commands.....	1410
17.6.12.5 Setting Health Check for a Container.....	1415
17.6.12.6 Setting an Environment Variable.....	1418
17.6.12.7 Enabling ICMP Security Group Rules.....	1420
17.6.12.8 Configuring an Image Pull Policy.....	1421
17.6.12.9 Configuring Time Zone Synchronization.....	1422
17.6.12.10 DNS Configuration.....	1423
17.6.12.11 Pod Scale-in Priorities.....	1430
17.6.13 Configuring QoS Rate Limiting for Inter-Pod Access.....	1430
17.6.14 Adding Pod Annotations.....	1431

17.7 Affinity and Anti-Affinity Scheduling.....	1432
17.7.1 Scheduling Policy Overview.....	1432
17.7.2 Custom Scheduling Policies.....	1434
17.7.2.1 Node Affinity.....	1434
17.7.2.2 Workload Affinity.....	1437
17.7.2.3 Workload Anti-Affinity.....	1440
17.7.3 Simple Scheduling Policies.....	1443
17.7.3.1 Workload-AZ Affinity.....	1443
17.7.3.2 Workload-AZ Anti-Affinity.....	1445
17.7.3.3 Workload-Node Affinity.....	1447
17.7.3.4 Workload-Node Anti-Affinity.....	1449
17.7.3.5 Workload-Workload Affinity.....	1451
17.7.3.6 Workload-Workload Anti-Affinity.....	1453
17.8 Networking.....	1454
17.8.1 Overview.....	1455
17.8.2 Container Network Models.....	1458
17.8.2.1 Overview.....	1458
17.8.2.2 Container Tunnel Network.....	1460
17.8.2.3 VPC Network.....	1464
17.8.3 Services.....	1469
17.8.3.1 Overview.....	1469
17.8.3.2 Intra-Cluster Access (ClusterIP).....	1470
17.8.3.3 NodePort.....	1475
17.8.3.4 LoadBalancer.....	1481
17.8.3.5 Service Annotations.....	1497
17.8.4 Ingress.....	1499
17.8.4.1 Overview.....	1499
17.8.4.2 Using ELB Ingresses on the Console.....	1501
17.8.4.3 Using kubectl to Create an ELB Ingress.....	1506
17.8.4.4 Using Nginx Ingresses on the Console.....	1519
17.8.5 DNS.....	1522
17.8.5.1 Overview.....	1522
17.8.5.2 DNS Configuration.....	1523
17.8.5.3 Using CoreDNS for Custom Domain Name Resolution.....	1530
17.8.5.4 Using NodeLocal DNSCache to Improve DNS Performance.....	1534
17.8.6 How Does a Container Access an Intranet in a VPC?.....	1536
17.8.7 Accessing Public Networks from a Container.....	1539
17.8.8 Configuring Network Policies to Restrict Pod Access.....	1541
17.9 Storage (CSI).....	1544
17.9.1 Overview.....	1545
17.9.2 Using Local Disks as Storage Volumes.....	1550
17.9.3 PersistentVolumes (PVs).....	1557

17.9.4 PersistentVolumeClaims (PVCs).....	1567
17.9.5 StorageClass.....	1573
17.9.6 Snapshots and Backups.....	1579
17.9.7 Using a Custom AK/SK to Mount an OBS Volume.....	1581
17.9.8 Setting Mount Options.....	1586
17.9.9 Deployment Examples.....	1589
17.9.9.1 EVS Volumes.....	1589
17.9.9.1.1 Using EVS Volumes.....	1589
17.9.9.1.2 Creating a Pod Mounted with an EVS Volume.....	1594
17.9.9.2 SFS Turbo Volumes.....	1598
17.9.9.2.1 Using SFS Turbo Volumes.....	1598
17.9.9.2.2 Creating a Deployment Mounted with an SFS Turbo Volume.....	1600
17.9.9.2.3 Creating a StatefulSet Mounted with an SFS Turbo Volume.....	1601
17.9.9.3 OBS Volumes.....	1604
17.9.9.3.1 Using OBS Volumes.....	1604
17.9.9.3.2 Creating a Deployment Mounted with an OBS Volume.....	1608
17.9.9.3.3 Creating a StatefulSet Mounted with an OBS Volume.....	1610
17.9.9.4 SFS Volumes.....	1612
17.9.9.4.1 Using SFS Volumes.....	1613
17.9.9.4.2 Creating a Deployment Mounted with an SFS Volume.....	1616
17.9.9.4.3 Creating a StatefulSet Mounted with an SFS Volume.....	1617
17.10 Monitoring and Logs.....	1620
17.10.1 Monitoring Overview.....	1620
17.10.2 Custom Monitoring.....	1624
17.10.3 Monitoring by Using the prometheus Add-on.....	1629
17.10.4 Container Logs.....	1634
17.11 Namespaces.....	1641
17.11.1 Creating a Namespace.....	1641
17.11.2 Managing Namespaces.....	1643
17.11.3 Configuring a Namespace-level Network Policy.....	1645
17.11.4 Setting a Resource Quota.....	1646
17.12 Configuration Center.....	1649
17.12.1 Creating a ConfigMap.....	1649
17.12.2 Using a ConfigMap.....	1652
17.12.3 Creating a Secret.....	1654
17.12.4 Using a Secret.....	1657
17.12.5 Cluster Secrets.....	1659
17.13 Charts (Helm).....	1661
17.13.1 My Charts.....	1661
17.13.1.1 Overview.....	1661
17.13.1.2 Preparing a Chart.....	1662
17.13.1.3 Uploading a Chart.....	1664



17.13.1.4 Creating a Workload from a Chart.....	1665
17.14 Add-ons.....	1667
17.14.1 Overview.....	1667
17.14.2 coredns (System Resource Add-on, Mandatory).....	1668
17.14.3 everest (System Resource Add-on, Mandatory).....	1673
17.14.4 autoscaler.....	1674
17.14.5 nginx-ingress.....	1680
17.14.6 metrics-server.....	1684
17.14.7 cce-hpa-controller.....	1685
17.14.8 prometheus.....	1686
17.14.9 gpu-beta.....	1691
17.14.10 huawei-npu.....	1694
17.14.11 node-local-dns.....	1695
17.15 Auto Scaling.....	1697
17.15.1 Overview.....	1697
17.15.2 Scaling a Workload.....	1699
17.15.2.1 Workload Scaling Mechanisms.....	1699
17.15.2.2 Creating an HPA Policy for Workload Auto Scaling.....	1702
17.15.2.3 Creating a CustomedHPA Policy for Workload Auto Scaling.....	1704
17.15.2.4 Managing Workload Scaling Policies.....	1709
17.15.3 Scaling a Cluster/Node.....	1711
17.15.3.1 Node Scaling Mechanisms.....	1712
17.15.3.2 Creating a Node Scaling Policy.....	1714
17.15.3.3 Managing Node Scaling Policies.....	1720
17.15.4 Using HPA and CA for Auto Scaling of Workloads and Nodes.....	1721
17.16 Permissions Management.....	1728
17.16.1 Permissions Overview.....	1728
17.16.2 Cluster Permissions (IAM-based).....	1730
17.16.3 Namespace Permissions (Kubernetes RBAC-based).....	1735
17.16.4 Example: Designing and Configuring Permissions for Users in a Department.....	1742
17.16.5 Permission Dependency of the CCE Console.....	1748
17.16.6 Pod Security.....	1756
17.16.6.1 Configuring a Pod Security Policy.....	1756
17.16.6.2 Configuring Pod Security Admission.....	1759
17.16.7 Service Account Token Security Improvement.....	1762
17.17 Cloud Trace Service (CTS).....	1763
17.17.1 CCE Operations Supported by CTS.....	1763
17.17.2 Querying CTS Logs.....	1768

# 1 High-Risk Operations

During service deployment or running, you may trigger high-risk operations at different levels, causing service faults or interruption. To help you better estimate and avoid operation risks, this section introduces the consequences and solutions of high-risk operations from multiple dimensions, such as clusters, nodes, networking, load balancing, logs, and EVS disks.

## Clusters and Nodes

**Table 1-1** High-risk operations and solutions

Category	Operation	Impact	Solution
Master node	Modifying the security group of a node in a cluster <b>NOTE</b> Naming rule of a security group: <i>Cluster name-cce-control-Random digits</i>	The master node may be unavailable.	Restore the security group by referring to "Creating a Cluster" and allow traffic from the security group to pass through.
	Letting the node expire or destroying the node	The master node will be unavailable.	This operation cannot be undone.
	Reinstalling the OS	Components on the master node will be deleted.	This operation cannot be undone.
	Upgrading components on the master or etcd node	The cluster may be unavailable.	Roll back to the original version.

Category	Operation	Impact	Solution
	Deleting or formatting core directory data such as <b>/etc/kubernetes</b> on the node	The master node will be unavailable.	This operation cannot be undone.
	Changing the node IP address	The master node will be unavailable.	Change the IP address back to the original one.
	Modifying parameters of core components (such as etcd, kube-apiserver, and docker)	The master node may be unavailable.	Restore the parameter settings to the recommended values. For details, see <a href="#">Modifying Cluster Configurations</a> .
	Replacing the master or etcd certificate	The cluster may be unavailable.	This operation cannot be undone.
Worker node	Modifying the security group of a node in a cluster <b>NOTE</b> Naming rule of a security group: <i>Cluster name-cce-node-Random digits</i>	The node may be unavailable.	Restore the security group and allow traffic from the security group to pass through.
	Modifying the DNS configuration ( <b>/etc/resolv.conf</b> ) of a node	Internal domain names cannot be accessed, which may lead to errors in functions such as add-on errors or errors in in-place node upgrade. <b>NOTE</b> If your service needs to use an on-premises DNS, configure the DNS in the workload. Do not change node's DNS address. For details, see <a href="#">DNS Configuration</a> .	Restore the DNS configuration based on the DNS configuration of a new node.
	Deleting the node	The node will become unavailable.	This operation cannot be undone.

Category	Operation	Impact	Solution
	Reinstalling the OS	Node components are deleted, and the node becomes unavailable.	Reset the node. For details, see <a href="#">Resetting a Node</a> .
	Upgrading the kernel or components on which the container platform depends (such as Open vSwitch, IPVLAN, Docker, and containerd)	The node may be unavailable or the network may be abnormal. <b>NOTE</b> Node running depends on the system kernel version. Do not use the <b>yum update</b> command to update or reinstall the operating system kernel of a node unless necessary. (Reinstalling the operating system kernel using the original image or other images is a risky operation.)	For details, see <a href="#">Resetting a Node</a> .
	Changing the node IP address	The node will become unavailable.	Change the IP address back to the original one.
	Modifying parameters of core components (such as kubelet and kube-proxy)	The node may become unavailable, and components may be insecure if security-related configurations are modified.	Restore the parameter settings to the recommended values. For details, see <a href="#">Modifying Node Pool Configurations</a> .
	Modifying OS configuration	The node may be unavailable.	Restore the configuration items or reset the node. For details, see <a href="#">Resetting a Node</a> .
	Deleting or modifying the <b>/opt/cloud/cce</b> and <b>/var/paas</b> directories, and deleting the data disk	The node will become unavailable.	Reset the node. For details, see <a href="#">Resetting a Node</a> .
	Modifying the node directory permission and the container directory permission	The permissions will be abnormal.	Do not modify the permissions. Restore the permissions if they have been modified.

Category	Operation	Impact	Solution
	Formatting or partitioning system disks, Docker disks, and kubelet disks on nodes.	The node may be unavailable.	Reset the node. For details, see <a href="#">Resetting a Node</a> .
	Installing other software on nodes	This may cause exceptions on Kubernetes components installed on the node, and make the node unavailable.	Uninstall the software that has been installed and restore or reset the node. For details, see <a href="#">Resetting a Node</a> .
	Modifying NetworkManager configurations	The node will become unavailable.	Reset the node. For details, see <a href="#">Resetting a Node</a> .
	Deleting system images such as <b>cce-pause</b> from the node	Containers cannot be created and system images cannot be pulled.	Copy the image from a functional node for restoration.
	Changing the flavor of a node in a node pool on the ECS console	If a node flavor is different from the flavor specified in the node pool where the node resides, the increased number of nodes in a node pool scale-out is different from the expected number.	Change the node flavor to the one specified in the node pool, or delete the node and perform a node pool scale-out again.

## Network

Table 1-2 Network

Operation	Impact	Solution
Changing the value of the kernel parameter <b>net.ipv4.ip_forward</b> to <b>0</b>	The network becomes inaccessible.	Change the value to <b>1</b> .
Changing the value of the kernel parameter <b>net.ipv4.tcp_tw_recycle</b> to <b>1</b>	The NAT service becomes abnormal.	Change the value to <b>0</b> .

Operation	Impact	Solution
Changing the value of the kernel parameter <b>net.ipv4.tcp_tw_reuse</b> to <b>1</b>	The network becomes abnormal.	Change the value to <b>0</b> .
Not configuring the node security group to allow UDP packets to pass through port 53 of the container CIDR block	The DNS in the cluster cannot work properly.	Restore the security group by referring to <a href="#">Buying a CCE Standard/Turbo Cluster</a> and allow traffic from the security group to pass through.
Deleting CRD resources of network-attachment-definitions of default-network	The container network is disconnected, or the cluster fails to be deleted.	If the resources are deleted by mistake, use the correct configurations to create the default-network resources.
Enabling the iptables firewall	By default, the iptables firewall is disabled on CCE. Enabling the firewall can leave the network inaccessible. <b>NOTE</b> Do not enable the iptables firewall. If the iptables firewall must be enabled, check whether the rules configured in <b>/etc/sysconfig/iptables</b> and <b>/etc/sysconfig/ip6tables</b> in the test environment will affect the network.	Disable the iptables firewall and check the rules configured in <b>/etc/sysconfig/iptables</b> and <b>/etc/sysconfig/ip6tables</b> .

## Containers

**Table 1-3** Containers

Operation	Impact	Solution
<p>Configuring privileged containers for a workload and directly operating the host hardware, which are prone to misoperations on the system files of the node</p> <p>For example, if you set the startup command to <code>/usr/sbin/init</code> and run <code>systemctl</code> in containers, the system files located in the <code>/lib</code> directory of the node may be damaged.</p>	<p>All mount points of the node will be unmounted. As a result, the node will be malfunctioning, resulting in failed pods and affected storage add-on functions.</p>	<p>Do not remove the mount points in the <code>/lib</code> directory of a node. Reset the node for recovery. For details, see <a href="#">Resetting a Node</a>.</p>

## Load Balancing

**Table 1-4** Service ELB

Operation	Impact	Solution
Deleting a load balancer that has been bound to a CCE cluster on the ELB console	Accessing the target Service or ingress will fail.	Do not delete such a load balancer.
Disabling a load balancer that has been bound to a CCE cluster on the ELB console	Accessing the target Service or ingress will fail.	Do not disable such a load balancer. If a load balancer has been disabled, enable it.
Changing the private IPv4 address of a load balancer on the ELB console	<ul style="list-style-type: none"> <li>The network traffic forwarded using the private IPv4 addresses will be interrupted.</li> <li>The IP addresses in the <code>status</code> field of Service or ingress YAML files will be changed.</li> </ul>	Do not change private IPv4 addresses of load balancers. Change them back if they have been changed.

Operation	Impact	Solution
Unbinding the IPv4 EIP from a load balancer on the ELB console	After the EIP is unbound from the load balancer, the load balancer will not be able to forward Internet traffic.	Restore the EIP binding.
Creating a custom listener on the ELB console for the load balancer managed by CCE	If a load balancer is automatically created when a Service or an ingress is created, the custom listener of the load balancer cannot be deleted when the Service or ingress is deleted. In this case, the load balancer cannot be automatically deleted.	Use the listener automatically created when a Service or an ingress is created. If a custom listener is used, manually delete the target load balancer.
Deleting a listener automatically created by CCE on the ELB console	<ul style="list-style-type: none"> <li>• Accessing the target Service or ingress will fail.</li> <li>• After master nodes are restarted, for example, due to a cluster upgrade, all your modifications will be reset by CCE.</li> </ul>	Re-create or update the Service or ingress.
Modifying the basic configurations such as the name, access control, timeout, or description of a listener created by CCE on the ELB console	After master nodes are restarted, for example, due to a cluster upgrade, all your modifications will be reset by CCE if the listener is deleted.	Do not modify the basic configurations of the listener created by CCE. Restore the configurations if they have been modified.
Modifying the backend server group of a listener created by CCE on the ELB console, including adding or deleting backend servers to or from the server group	<ul style="list-style-type: none"> <li>• Accessing the target Service or ingress will fail.</li> <li>• After master nodes are restarted, for example, due to a cluster upgrade, all your modifications will be reset by CCE. <ul style="list-style-type: none"> <li>- Deleted backend servers will be restored.</li> <li>- Added backend servers will be removed.</li> </ul> </li> </ul>	Re-create or update the Service or ingress.



Operation	Impact	Solution
Replacing the backend server group of a listener created by CCE on the ELB console	<ul style="list-style-type: none"> <li>• Accessing the target Service or ingress will fail.</li> <li>• After master nodes are restarted, for example, due to a cluster upgrade, all servers in the backend server group will be reset by CCE.</li> </ul>	Re-create or update the Service or ingress.
Modifying the forwarding policy of a listener created by CCE on the ELB console, including adding or deleting forwarding rules	<ul style="list-style-type: none"> <li>• Accessing the target Service or ingress will fail.</li> <li>• After master nodes are restarted, for example, due to a cluster upgrade, all your modifications will be reset by CCE if the forwarding rules are added using an ingress.</li> </ul>	Do not modify the forwarding policy of such a listener. Restore the configurations if they have been modified.
Changing the ELB certificate on the ELB console for a load balancer managed by CCE	After master nodes are restarted, for example, due to a cluster upgrade, all servers in the backend server group will be reset by CCE.	Use the YAML file of the ingress to automatically manage certificates.

## Logs

Table 1-5 Logs

Operation	Impact	Solution
Deleting the <b>/tmp/ccs-log-collector/pos</b> directory on the host machine	Logs are collected repeatedly.	None
Deleting the <b>/tmp/ccs-log-collector/buffer</b> directory on the host machine	Logs are lost.	None

## EVS Disks

**Table 1-6** EVS disks

Operation	Impact	Solution	Remarks
Manually unmounting an EVS disk on the console	An I/O error occurs when data is written into a pod.	Delete the mount path from the node and schedule the pod again.	The file in the pod records the location where files are to be collected.
Unmounting the disk mount path on the node	Pod data is written into a local disk.	Remount the corresponding path to the pod.	The buffer contains log cache files to be consumed.
Operating EVS disks on the node	Pod data is written into a local disk.	None	None

## Add-ons

**Table 1-7** Add-ons

Operation	Impact	Solution
Modifying add-on resources on the backend	The add-on becomes malfunctioning or other unexpected issues occur.	Perform operations on the add-on configuration page or using open add-on management APIs.

# 2 Clusters

---

## 2.1 Cluster Overview

### 2.1.1 Basic Cluster Information

**Kubernetes** is an open source container orchestration engine for automating deployment, scaling, and management of containerized applications.

For developers, Kubernetes is a cluster operating system. Kubernetes provides service discovery, scaling, load balancing, self-healing, and even leader election, freeing developers from infrastructure-related configurations.

#### Cluster Network

A cluster network can be divided into three network types:

- Node network: IP addresses are assigned to nodes in a cluster.
- Container network: IP addresses are assigned to containers in a cluster for communication. Currently, multiple container network models are supported, and each model has its own working mechanism.
- Service network: A Service is a Kubernetes object used to access containers. Each Service has a static IP address.

When you create a cluster, select a proper CIDR block for each network. Ensure that the CIDR blocks do not conflict with each other and have sufficient available IP addresses. **You cannot change the container network model after the cluster is created.** Plan the container network model properly in advance.

You are advised to learn about the cluster network and container network models before creating a cluster. For details, see [Container Network](#).

#### Master Nodes and Cluster Scale

When you create a cluster on CCE, you can have one or three master nodes. Three master nodes will be deployed in a cluster for HA.

The master node specifications decide the number of nodes that can be managed by a cluster. You can select the cluster management scale, for example, 50 or 200 nodes.

## Cluster Lifecycle

**Table 2-1** Cluster status

Status	Description
Creating	A cluster is being created and is requesting for cloud resources.
Running	A cluster is running properly.
Hibernating	A cluster is hibernating.
Awaking	A cluster is being woken up.
Upgrading	A cluster is being upgraded.
Resizing	The cluster flavor is being changed.
Unavailable	A cluster is unavailable.
Deleting	A cluster is being deleted.

## 2.1.2 Kubernetes Version Release Notes

### 2.1.2.1 Kubernetes 1.29 Release Notes

CCE has passed the Certified Kubernetes Conformance Program and is a certified Kubernetes offering. CCE allows you to create Kubernetes clusters 1.29. This section describes the changes made in Kubernetes 1.29.

### Indexes

- [New and Enhanced Features](#)
- [API Changes and Removals](#)
- [Enhanced Kubernetes 1.29 on CCE](#)
- [References](#)

### New and Enhanced Features

- The load balancer IP mode for Services is in the alpha state.  
The load balancer IP mode for Services is promoted to alpha. Kubernetes 1.29 adds the **ipMode** field to the Services' **status** field for configuring traffic forwarding from Services within a cluster to pods. If **ipMode** is set to **VIP**, traffic delivered to a node with the destination set to the load balancer's IP address and port will be redirected to the target node by kube-proxy. If it is set to **Proxy**, traffic delivered to a node will be sent to the load balancer and

then redirected to the target node by the load balancer. This feature addresses the issue of load balancer functions being missed due to traffic bypassing it. For details, see [Load Balancer IP Mode for Services](#).

- The **nftables proxy mode** is in the alpha state.  
The **nftables proxy mode** is promoted to alpha. This feature allows kube-proxy to run in **nftables mode**. In this mode, kube-proxy configures packet forwarding rules using the **nftables API** of the kernel netfilter subsystem. For details, see [nftables proxy mode](#).
- **Garbage collection for unused container images** is in the alpha state.  
The garbage collection for unused container images is promoted to alpha. This feature allows you to specify the maximum time a local image can be unused for each node. If the time expires, the image will be garbage collected. To configure the setting, specify the **ImageMaximumGCAge** field for kubelet. For details, see [Garbage collection for unused container images](#).
- **PodLifecycleSleepAction** is in the alpha state.  
**PodLifecycleSleepAction** is promoted to alpha. This feature introduces the sleep hook to the container lifecycle hooks. You can pause a container for a specified duration after it starts or before it is stopped by enabling this feature. For details, see [Hook handler implementations](#).
- **KubeletSeparateDiskGC** is in the alpha state.  
**KubeletSeparateDiskGC** is promoted to alpha. With this feature enabled, container images and containers can be garbage collected even if they are on separate file systems.
- **matchLabelKeys** and **mismatchLabelKeys** are in the alpha state.  
**matchLabelKeys** and **mismatchLabelKeys** are promoted to alpha. With these features enabled, the **matchLabelKeys** and **mismatchLabelKeys** fields are added to the pod affinity and anti-affinity configurations. This allows for configurations of more affinity and anti-affinity policies between pods. For details, see [matchLabelKeys and mismatchLabelKeys](#).
- The **clusterTrustBundle** projected volumes are in the alpha state.  
The **clusterTrustBundle** projected volumes are promoted to alpha. With this feature enabled, the **clusterTrustBundle** projected volume source injects the contents of one or more **ClusterTrustBundle** objects as an automatically-updating file. For details, see [clusterTrustBundle projected volumes](#).
- Pulling images based on runtime classes of pods is in the alpha state.  
Pulling images based on runtime classes is promoted to alpha. With this feature enabled, the kubelet references container images by a tuple (of image name or runtime handler) rather than just the image name or digest. Your container runtime may adapt its behavior based on the selected runtime handler. Pulling images based on runtime classes will be helpful for VM based containers. For details, see [Image pull per runtime class](#).
- The **PodReadyToStartContainers** condition is in the beta state.  
The **PodReadyToStartContainers** condition is promoted to beta. Kubernetes 1.29 introduces the **PodReadyToStartContainers** condition to the pods' **status** field. If it is set to **true**, the sandbox of a pod is ready and service containers can be created. This feature enables cluster administrators to gain a clearer and more comprehensive view of pod sandbox creation completion and container readiness. This enhanced visibility allows them to make better-

informed decisions and troubleshoot issues more effectively. For details, see [PodReadyToStartContainers Condition Moves to Beta](#).

- Two job-related features are in the beta state.
  - Pod replacement policy (beta)

The pod replacement policy feature moves to beta. This feature ensures that a pod is replaced only when it reaches the **Failed** state, which means that **status.phase** becomes **Failed**. It does not recreate a pod when the deletion timestamp is not empty and the pod is still being deleted. This prevents two pods from occupying index and node resources concurrently.
  - Backoff limit per index (beta)

The backoff limit per index moves to beta. By default, pod failures for indexed jobs are counted and restricted by the global limit of retries, specified by **.spec.backoffLimit**. This means that if there is a consistently failing index in a job, pods specified by the job will be restarted repeatedly until pod failures exhaust the limit. Once the limit is reached, the job is marked failed and pods for other indexes in the job may never be even started. The feature allows you to complete execution of all indexes, despite some indexes failing, and to better use the compute resources by avoiding unnecessary retries of consistently failing indexes.
- Native sidecar containers are in the beta state.

Native sidecar containers are promoted to beta. The **restartPolicy** field is added to **initContainers**. When this field is set to **Always**, the sidecar container is enabled. The sidecar container and service container are deployed in the same pod. This cannot prolong the pod lifecycle. Sidecar containers are commonly used in scenarios such as network proxy and log collection. For details, see [Sidecar Containers](#).
- The legacy ServiceAccount token cleaner is in the beta state.

Legacy ServiceAccount token cleaner is promoted to beta. It runs as part of **kube-controller-manager** and checks every 24 hours to see if any auto-generated legacy ServiceAccount token has not been used in a specific amount of time (one year by default, specified by **--legacy-service-account-token-clean-up-period**). If so, the cleaner marks those tokens as invalid and adds the **kubernetes.io/legacy-token-invalid-since** label whose value is the current date. If an invalid token is not used for a specific period of time (one year by default, specified by **--legacy-service-account-token-clean-up-period**), the cleaner deletes it. For details, see [Legacy ServiceAccount Token Cleaner](#).
- **DevicePluginCDIDevices** is in the beta state.

**DevicePluginCDIDevices** moves to beta. With this feature enabled, plugin developers can use the **CDIDevices** field added to **DeviceRunContainerOptions** to pass CDI device names directly to CDI enabled runtimes.
- **PodHostIPs** is in the beta state.

The **PodHostIPs** feature moves to beta. With this feature enabled, Kubernetes adds the **hostIPs** field to **Status** of pods and downward API to expose node IP addresses to workloads. This field specifies the dual-stack protocol version of the host IP address. The first IP address is always the same as the host IP address.

- The API Priority and Fairness feature (APF) is in the GA state.  
APF moves to GA. APF classifies and isolates requests in a more fine-grained way. It improves max-inflight limitations. It also introduces a limited amount of queuing, so that the API server does not reject any request in cases of very brief bursts. Requests are dispatched from queues using a fair queuing technique so that, for example, a poorly-behaved controller does not cause others (even at the same priority level) to become abnormal. For details, see [API Priority and Fairness](#).
- **APIListChunking** is in the GA state.  
The **APIListChunking** feature moves to GA. This feature allows clients to perform pagination in List requests to avoid performance problems caused by returning too much data at a time.
- **ServiceNodePortStaticSubrange** is in the GA state.  
The **ServiceNodePortStaticSubrange** feature moves to GA. With this feature enabled, kubelet calculates the size of reserved IP addresses based on the ranges of the NodePort Services and divides node ports into static band and dynamic band. During automatic node port assignment, dynamic band is preferentially assigned, which helps avoid port conflicts during static band assignment. For details, see [ServiceNodePortStaticSubrange](#).
- The phase transition timestamp of PersistentVolume (PV) is in the beta state.  
The PV phase transition timestamp moves to beta. With this feature enabled, Kubernetes adds the **lastPhaseTransitionTime** field to the **status** field of a PV to indicate the time when the PV phase changes last time. Cluster administrators are now able to track the last time a PV transitioned to a different phase, allowing for more efficient and informed resource management. For details, see [PersistentVolume Last Phase Transition Time in Kubernetes](#).
- **ReadWriteOncePod** is in the GA state.  
The **ReadWriteOncePod** feature moves to GA. With this feature enabled, you can set the access mode to **ReadWriteOncePod** in a PersistentVolumeClaim (PVC) to ensure that only one pod can modify data in the volume at a time. This can prevent data conflicts or damage. For details, see [ReadWriteOncePod](#).
- **CSINodeExpandSecret** is in the GA state.  
The **CSINodeExpandSecret** feature moves to GA. This feature allows secret authentication data to be passed to a CSI driver for use when a node is added.
- The CEL-based CustomResourceDefinition (CRD) verification capability is in the GA state.  
The CEL-based CRD verification capability moves to GA. With this feature enabled, you are allowed to use the CEL to define validation rules in CRDs, which are more efficient than webhook. For details, see [CRD verification rules](#).

## API Changes and Removals

- The time zone of a newly created cron job cannot be configured using **TZ** or **CRON\_TZ** in **.spec.schedule**. Use **.spec.timeZone** instead. Cron jobs that have been created are not affected by this change.

- The alpha API **ClusterCIDR** is removed.
- The startup parameter **--authentication-config** is added to kube-apiserver to specify the address of the **AuthenticationConfiguration** file. This startup parameter is mutually exclusive with the **--oidc-\*** startup parameter.
- The API version **kubescheduler.config.k8s.io/v1beta3** of **KubeSchedulerConfiguration** is removed. Migrate **kube-scheduler** configuration files to **kubescheduler.config.k8s.io/v1**.
- The CEL expressions are added to **v1alpha1 AuthenticationConfiguration**.
- The **ServiceCIDR** type is added. It allows you to dynamically configure the IP address range used by a cluster to allocate the Service ClusterIPs.
- The startup parameters **--contrack-udp-timeout** and **--contrack-udp-timeout-stream** are added to **kube-proxy**. They are options for configuring the kernel parameters **nf\_contrack\_udp\_timeout** and **nf\_contrack\_udp\_timeout\_stream**.
- Support for CEL expressions is added to **WebhookMatchCondition** of **v1alpha1 AuthenticationConfiguration**.
- The type of **PVC.spec.Resource** is changed from **ResourceRequirements** to **VolumeResourceRequirements**.
- **onPodConditions** in **PodFailurePolicyRule** is marked as optional.
- The API version **flowcontrol.apiserver.k8s.io/v1beta3** of **FlowSchema** and **PriorityLevelConfiguration** has been promoted to **flowcontrol.apiserver.k8s.io/v1**, and the following changes have been made:
  - **PriorityLevelConfiguration**:  
The **.spec.limited.nominalConcurrencyShares** field defaults to **30** if the field is omitted. To ensure compatibility with 1.28 API servers, specifying an explicit **0** value is not allowed in the **v1** version in 1.29. In 1.30, explicit **0** values will be allowed in this field in the **v1** API. The **flowcontrol.apiserver.k8s.io/v1beta3** APIs are deprecated and will no longer be served in 1.32.
- The kube-proxy command line document is updated. kube-proxy does not bind any socket to the IP address specified by **--bind-address**.
- The **selectorSpread** scheduler plugin is replaced by **podTopologySpread**.
- If CSI-Node-Driver is not running, NodeStageVolume calls will be retried.
- **ValidatingAdmissionPolicy** type checking now supports CRDs. To use this feature, the **ValidatingAdmissionPolicy** feature gate must be enabled.
- The startup parameter **--nf-contrack-tcp-be-liberal** is added to **kube-proxy**. You can configure it by setting the kernel parameter **nf\_contrack\_tcp\_be\_liberal**.
- The startup parameter **--init-only** is added to **kube-proxy**. Setting the flag makes **kube-proxy** init container run in the privileged mode, perform its initial configuration, and then exit.
- The **fileSystem** field of container is added to the response body of CRI. It specifies the file system usage of a container. Originally, the **fileSystem** field contains only the file system of the container images.
- All built-in cloud providers are disabled by default. If you still need to use them, you can configure the **DisableCloudProviders** and **DisableKubeletCloudCredentialProvider** feature gates to disable or enable cloud providers.



- **--node-ips** can be used in kubelet to configure IPv4/IPv6 dual-stack. If **--cloud-provider** is set to **external**, you are allowed to use **--node-ips** to configure IPv4/IPv6 dual-stack for node IP addresses. To use **--node-ips**, you need to enable the **CloudDualStackNodeIPs** feature gate.

## Enhanced Kubernetes 1.29 on CCE

During a version maintenance period, CCE periodically updates Kubernetes 1.29 and provides enhanced functions.

For details about cluster version updates, see [Patch Versions](#).

## References

For more details about the performance comparison and function evolution between Kubernetes 1.29 and other versions, see [Kubernetes v1.29 Release Notes](#).

### 2.1.2.2 Kubernetes 1.28 Release Notes

CCE has passed the Certified Kubernetes Conformance Program and is a certified Kubernetes offering. CCE allows you to create Kubernetes clusters 1.28. This section describes the changes made in Kubernetes 1.28.

## Indexes

- [Important Notes](#)
- [New and Enhanced Features](#)
- [API Changes and Removals](#)
- [Feature Gate and Command Line Parameter Changes and Removals](#)
- [Enhanced Kubernetes 1.28 on CCE](#)
- [References](#)

## Important Notes

- In Kubernetes 1.28, the scheduling framework is improved to reduce useless retries. The overall scheduling performance is enhanced. If a custom scheduler plugin is used in a cluster, you can perform the adaptation upgrade following instructions in [GitHub](#).
- The Ceph FS in-tree plugin has been deprecated in Kubernetes 1.28 and will be removed in Kubernetes 1.31. (The community does not plan to support CSI migration.) Use [Ceph CSI driver](#) instead.
- The Ceph RBD in-tree plugin has been deprecated in Kubernetes 1.28 and will be removed in Kubernetes 1.31. (The community does not plan to support CSI migration.) Use RBD [Ceph CSI driver](#) instead.

## New and Enhanced Features

Features in alpha stage are disabled by default, those in beta stage are enabled by default, and those in GA stage are always enabled and they cannot be disabled. The function of turning on or off the features in GA stage will be removed in later

Kubernetes versions. CCE policies for new features are the same as those in the community.

- The version skew policy is expanded to three versions.  
Starting with control planes 1.28 and worker nodes 1.25, the Kubernetes skew policy expands the supported control plane and worker node skew to three versions. This enables annual minor version upgrades of nodes while staying on supported minor versions. For details, see [Version Skew Policy](#).
- Retroactive Default StorageClass moves to GA.  
The retroactive default StorageClass assignment graduates to GA. This enhancement brings a significant improvement to how default StorageClasses are assigned to PersistentVolumeClaims (PVCs).  
The PV controller has been modified to automatically assign a default StorageClass to any unbound PVC with **storageClassName** not configured. Additionally, the PVC admission validation mechanism within the API server has been adjusted to allow changing values from an unset state to an actual StorageClass name. For details, see [Retroactive default StorageClass assignment](#).
- Native sidecar containers are introduced.  
The native sidecar containers are available in alpha. Kubernetes 1.28 adds **restartPolicy** to Init containers. This field is available when the SidecarContainers feature gate is enabled. However, there are still some problems to be solved in the native sidecar containers. Therefore, the Kubernetes community recommends only using this feature gate in [short lived testing clusters](#) at the alpha phase. For details, see [Introducing native sidecar containers](#).
- Mixed version proxy is introduced.  
A new mechanism (mixed version proxy) is released to improve cluster upgrade. It is an alpha feature in Kubernetes 1.28. When a cluster undergoes an upgrade, API servers of different versions in the cluster can serve different sets (groups, versions, or resources) of built-in resources. A resource request made in this scenario may be served by any of the available API servers, potentially resulting in the request ending up at an API server that may not be aware of the requested resource. As a result, the request fails. This feature can solve this problem. (Note that CCE provides hitless upgrade. Therefore, this feature is not used in CCE clusters.) For details, see [A New \(alpha\) Mechanism For Safer Cluster Upgrades](#).
- Non-graceful node shutdown moves to GA.  
The non-graceful node shutdown is now GA in Kubernetes 1.28. When a node was shut down and that shutdown was not detected by the Kubelet's Node Shutdown Manager, the StatefulSet pods that run on this node will stay in the terminated state and cannot be moved to a running node. If you have confirmed that the shutdown node is unrecoverable, you can add an **out-of-service** taint to the node. This ensures that the StatefulSet pods and VolumeAttachments on this node can be forcibly deleted and the corresponding pods will be created on a healthy node. For details, see [Non-Graceful Node Shutdown Moves to GA](#).
- NodeSwap moves to beta.  
Support for NodeSwap goes to beta in Kubernetes 1.28. NodeSwap is disabled by default and can be enabled using the NodeSwap feature gate. NodeSwap

allows you to configure swap memory usage for Kubernetes workloads running on Linux on a per-node basis. Note that although NodeSwap has reached beta, there are still some problems to be solved and security risks to be enhanced. For details, see [Beta Support for Using Swap on Linux](#).

- Two job-related features are added.

Two alpha features are introduced: [delayed creation of replacement pods](#) and [backoff limit per index](#).

- Delayed creation of replacement pods

By default, when a pod enters the terminating state (for example, due to the preemption or eviction), Kubernetes immediately creates a replacement pod. Therefore, both pods are running concurrently.

In Kubernetes 1.28, this feature can be enabled by turning on the JobPodReplacementPolicy feature gate. With this feature gate enabled, you can set the **podReplacementPolicy** field under **spec** of a job to **Failed**. In this way, pods would only be replaced when they reached the failed phase, and not when they are terminating. Additionally, you can check the **.status.termination** field of a job. The value of this field is the number of pods owned by the job that are currently terminating.

- Backoff limit per index

By default, pod failures for indexed jobs are recorded and restricted by the global limit of retries, specified by **.spec.backoffLimit**. This means that if there is a consistently failing index in a job, pods specified by the job will be restarted repeatedly until pod failures exhaust the limit. Once the limit is reached, the job is marked failed and pods for other indexes in the job may never be even started.

In Kubernetes 1.28, this feature can be enabled by turning on the JobBackoffLimitPerIndex feature gate of a cluster. With this feature gate enabled, **.spec.backoffLimitPerIndex** can be specified when an indexed job is created. Only if the failures of pods with all indexes specified in this job exceed the upper limit, pods specified by the job will not be restarted.

- Some CEL related features are improved.

CEL related capabilities are enhanced.

- CEL used to validate CRDs moves to beta.

This feature has been upgraded to beta since Kubernetes 1.25. By embedding CEL expressions into CRDs, developers can solve most of the CR validation use cases without using webhooks. More CEL functions, such as support for default value and CRD conversion, will be developed in later Kubernetes versions.

- CEL admission control graduates to beta.

CEL admission control is customizable. With CEL expressions, you can decide whether to accept or reject requests received by kube-apiserver. CEL expressions can also serve as a substitute for admission webhooks. Kubernetes 1.28 has upgraded CEL admission control to beta and introduced new functions, such as:

- ValidatingAdmissionPolicy can correctly handle the **authorizer** variable.
- ValidatingAdmissionPolicy can have the **messageExpression** field checked.

- The ValidatingAdmissionPolicy controller is added to kube-controller-manager to check the type of the CEL expression in ValidatingAdmissionPolicy and save the reason in the **status** field.
- CEL expressions can contain a combination of one or more variables, which can be defined in ValidatingAdmissionPolicy. These variables can be used to define other variables.
- CEL library functions can be used to parse resources specified by **resource.Quantity** in Kubernetes.
- Other features
  - The ServiceNodePortStaticSubrange feature gate moves to beta. With this feature enabled, static port range can be reserved to avoid conflicts with dynamically allocated ports. For details, see [Avoiding Collisions Assigning Ports to NodePort Services](#).
  - The alpha feature ConsistentListFromCache is added to allow the API server to serve consistent lists from cache. Get and list requests can read data from the cache instead of etcd.
  - In Kubernetes 1.28, kubelet can configure the drop-in directory (alpha). This feature allows you to add support for the **--config-dir** flag to kubelet so that you can specify an insert directory that overwrites the kubelet configuration in **/etc/kubernetes/kubelet.conf**.
  - ExpandedDNSConfig moves to GA and is enabled by default. With this feature enabled, DNS configurations can be expanded.
  - The alpha feature CRDValidationRatcheting is added. This feature allows CRs with failing validations to pass if a Patch or Update request does not alter any of the invalid fields.
  - **--concurrent-cron-job-syncs** is added to kube-controller-manager to configure the number of workers for the cron job controller.

## API Changes and Removals

- **NetworkPolicyStatus** is removed. There is no status attribute in a network policy.
- **annotationbatch.kubernetes.io/cronJob-scheduled-timestamp** is added to job objects to indicate the creation time of a job.
- The **podReplacementPolicy** and **terminating** fields are added to job APIs. With these fields specified, once a previously created pod is terminated in a job, the job immediately starts a new pod to replace the pod. The new fields allow you to specify whether to replace the pod immediately after the previous pod is terminated (original behavior) or replace the pod after the existing pod is completely terminated (new behavior). This is an alpha feature, and you can enable it by turning on the [JobPodReplacementPolicy](#) feature gate in your cluster.
- The **BackoffLimitPerIndex** field is available in a job. Pods specified by a job share a backoff mechanism. When backoff times of the job reach the limit, this job is marked as failed and resources, including indexes that are not running, are cleared up. This field allows you to configure backoff limit for a single index. For details, see [Backoff limit per index](#).

- The **ServedVersions** field is added to the **StorageVersion** API. This change is introduced by mixed version proxy. The new field is used to indicate a version that can be provided by the API server.
- **SelfSubjectReview** is added to **authentication.k8s.io/v1**, and **kubectl auth whoami** goes to GA.
- **LastPhaseTransitionTime** is added to **PersistentVolume**. The new field is used to store the last time when a volume changes to a different phase.
- **resizeStatus** in **PVC.Status** is replaced by **AllocatedResourceStatus**. The new field indicates the statuses of the storage resize operation. The default value is an empty string.
- If **hostNetwork** is set to **true** and ports are specified for a pod, the **hostport** field will be automatically configured.
- StatefulSet pods have the pod index set as a pod label **statefulset.kubernetes.io/pod-index**.
- **PodHasNetwork** in the **Condition** field of pods has been renamed to **PodReadyToStartContainers**. The new field specifies that containers are ready to start after the network, volumes, and sandbox pod have been created.
- A new configuration option **delayCacheUntilActive** is added to **KubeSchedulerConfiguration**. If **delayCacheUntilActive** is set to **true**, kube-scheduler on the leader will not cache scheduling information. This reduces the memory pressure of other master nodes, but slows down the failover speed after the leader failed.
- The **namespaceParamRef** field is added to **admissionregistration.k8s.io/v1alpha1.ValidatingAdmissionPolicy**.
- The **reason** and **fieldPath** fields are added to CRD validation rules to allow you to specify reason and field path after verification failed.
- The CEL expression of ValidatingAdmissionPolicy supports namespace access via namespaceObject.
- API groups ValidatingAdmissionPolicy and ValidatingAdmissionPolicyBinding are promoted to beta1.
- A ValidatingAdmissionPolicy now has its **messageExpression** field checked against resolved types.

## Feature Gate and Command Line Parameter Changes and Removals

- **-short** is removed from kubelet. Therefore, the default output of **kubectl version** is the same as that of **kubectl version -short**.
- **--volume-host-cidr-denylist** and **--volume-host-allow-local-loopback** are removed from kube-controller-manager. **--volume-host-cidr-denylist** is a comma-separated list of CIDR ranges. Volume plugins at these IP addresses are not allowed. If **--volume-host-allow-local-loopback** is set to **false**, the local loopback IP address and the CIDR ranges specified in **--volume-host-cidr-denylist** are disabled.
- **--azure-container-registry-config** is deprecated in kubelet and will be deleted in later Kubernetes versions. Use **--image-credential-provider-config** and **--image-credential-provider-bin-dir** instead.
- **--lock-object-namespace** and **--lock-object-name** are removed from kube-scheduler. Use **--leader-elect-resource-namespace** and **--leader-elect-**

**resource-name** or **ComponentConfig** instead. (**--lock-object-namespace** is used to define the namespace of a lock object, and **--lock-object-name** is used to define the name of a lock object.)

- KMS v1 is deprecated and will only receive security updates. Use KMS v2 instead. In later Kubernetes versions, use **--feature-gates=KMSv1=true** to configure a KMS v1 provider.
- The **DelegateFSGroupToCSIDriver**, **DevicePlugins**, **KubeletCredentialProviders**, **MixedProtocolLBService**, **ServiceInternalTrafficPolicy**, **ServiceIPStaticSubrange**, and **EndpointSliceTerminatingCondition** feature gates are removed.

## Enhanced Kubernetes 1.28 on CCE

During a version maintenance period, CCE periodically updates Kubernetes 1.28 and provides enhanced functions.

For details about cluster version updates, see [Release Notes for CCE Cluster Versions](#).

## References

For more details about the performance comparison and function evolution between Kubernetes 1.28 and other versions, see [Kubernetes v1.28 Release Notes](#).

### 2.1.2.3 Kubernetes 1.27 Release Notes

CCE has passed the Certified Kubernetes Conformance Program and is a certified Kubernetes offering. CCE allows you to create clusters of Kubernetes 1.27. This section describes the changes made in Kubernetes 1.27 compared with Kubernetes 1.25.

## Indexes

- [New Features](#)
- [Deprecations and Removals](#)
- [Enhanced Kubernetes 1.27 on CCE](#)
- [References](#)

## New Features

### Kubernetes 1.27

- **SeccompDefault** is stable.  
To use **SeccompDefault**, add the **--seccomp-default** [command line flag](#) using kubelet on each node. If this feature is enabled, the **RuntimeDefault** profile will be used for all workloads by default, instead of the **Unconfined** (seccomp disabled) profile.
- Jobs' scheduling directives are configurable.  
This feature was introduced in Kubernetes 1.22 and is stable in Kubernetes 1.27. In most cases, you use a job to influence where the pods will run, like all in the same AZ. This feature allows scheduling directives to be modified before a job starts. You can use the **suspend** field to suspend a job. In the

suspension phase, the scheduling directives (such as the node selector, node affinity, anti-affinity, and tolerations) in the job's pod template can be modified. For details, see [Mutable Scheduling Directives](#).

- Downward API hugepages are stable.

In Kubernetes 1.20, **requests.hugepages-*<pagesize>*** and **limits.hugepages-*<pagesize>*** were introduced to the [downward API](#). Requests and limits can be configured for hugepages like other resources.

- Pod scheduling readiness moves to beta.

After a pod is created, the Kubernetes scheduler selects an appropriate node to run the pod in the pending state. In practice, some pods may stay in the pending state for a long period due to insufficient resources. These pods may affect the running of other components like Cluster Autoscaler in the cluster. By specifying or deleting **.spec. schedulingGates** for a pod, you can control when the pod is ready for scheduling. For details, see [Pod Scheduling Readiness](#).

- Accessing node logs using Kubernetes APIs is supported.

This function is in the alpha phase. The cluster administrator can directly query node logs to help debug malfunctioning services running on the node. To use this function, ensure that the NodeLogQuery [feature gate](#) is enabled for that node and the kubelet configuration options **enableSystemLogHandler** and **enableSystemLogQuery** are set to **true**.

- ReadWriteOncePod access mode moves to beta.

Kubernetes 1.22 introduced a ReadWriteOncePod access mode for PVs and PVCs. This feature has evolved into the beta phase. A volume can be mounted to a single pod in read/write mode. Use this access mode if you want to ensure that only one pod in the cluster can read that PVC or write to it. For details, see [Access Modes](#).

- The **matchLabelKeys** field in the pod topology spread constraint moves to beta.

**matchLabelKeys** is a list of pod label keys. It is used to select a group of pods over which spreading will be calculated. With **matchLabelKeys**, you do not need to update **pod.spec** between different revisions. The controller or operator just needs to set different values to the same label key for different revisions. The scheduler will automatically determine the values based on **matchLabelKeys**. For details, see [Pod Topology Distribution Constraints](#).

- The function of efficiently labeling SELinux volumes moves to beta.

By default, the container runtime recursively assigns the SELinux label to all files on all pod volumes. To speed up this process, Kubernetes uses the mount option **-o context=*<label>*** to immediately change the SELinux label of the volume. For details, see [Efficient SELinux volume relabeling](#).

- VolumeManager reconstruction goes to beta.

After the VolumeManager is reconstructed, if the NewVolumeManagerReconstruction [feature gate](#) is enabled, mounted volumes will be obtained in a more effective way during kubelet startup.

- Server side field validation and OpenAPI V3 are stable.

OpenAPI V3 was added in Kubernetes 1.23. In Kubernetes 1.24, it moved to beta. In Kubernetes 1.27, it is stable.

- StatefulSet start ordinal moves to beta.  
Kubernetes 1.26 introduced a new, alpha-level feature for StatefulSets to control the ordinal numbering of pod replicas. Since Kubernetes 1.27, this feature moves to beta. The ordinals can start from arbitrary non-negative numbers. For details, see [Kubernetes 1.27: StatefulSet Start Ordinal Simplifies Migration](#).
- **ContainerResource** metric in HorizontalPodAutoscaler moves to beta.  
Kubernetes 1.20 introduced the **ContainerResource** metric in HorizontalPodAutoscaler (HPA). In Kubernetes 1.27, this feature moves to beta, and the HPAContainerMetrics feature gate is enabled by default.
- StatefulSet PVC auto deletion moves to beta.  
Kubernetes 1.27 provides a new policy to control the lifecycle of PVCs of StatefulSets. This policy allows users to specify if the PVCs generated from the StatefulSet spec template should be automatically deleted or retained when the StatefulSet is deleted or replicas in the StatefulSet are scaled down. For details, see [PersistentVolumeClaim retention](#).
- Volume group snapshots are introduced.  
Volume group snapshots are introduced as an alpha feature in Kubernetes 1.27. This feature allows users to create snapshots for multiple volumes to ensure data consistency when a fault occurs. It uses a label selector to group multiple PVCs for snapshot. This feature only supports CSI volume drivers. For details, see [Kubernetes 1.27: Introducing an API for Volume Group Snapshots](#).
- **kubectl apply** pruning is more secure and efficient.  
In Kubernetes 1.5, the **--prune** flag was introduced in **kubectl apply** to delete resources that are no longer needed. This allowed **kubectl apply** to automatically clear resources removed from the current configuration. However, the existing implementation of **--prune** has design defects that degrade its performance and lead to unexpected behaviors. In Kubernetes 1.27, **kubectl apply** provides ApplySet-based pruning, which is in the alpha phase. For details, see [Declarative Management of Kubernetes Objects Using Configuration Files](#).
- Conflicts during port allocation to NodePort Service can be avoided.  
In Kubernetes 1.27, you can enable a new **feature gate** `ServiceNodePortStaticSubrange` to use different port allocation policies for NodePort Services. This mitigates the risk of port conflicts. This feature is in the alpha phase.
- Resizing resources assigned to pods without restarting the containers is supported.  
Kubernetes 1.27 allows users to resize CPU and memory resources assigned to pods without restarting the container. This feature is in the alpha phase. For details, see [Kubernetes 1.27: In-place Resource Resize for Kubernetes Pods \(alpha\)](#).
- Pod startup is accelerated.  
A series of parameter adjustments like parallel image pulls and increased default API query limit for kubelet per second are made in Kubernetes 1.27 to accelerate pod startup. For details, see [Kubernetes 1.27: updates on speeding up Pod startup](#).



- KMS V2 moves to beta.  
The key management KMS V2 API goes to beta. This has greatly improved the performance of the KMS encryption provider. For details, see [Using a KMS provider for data encryption](#).

### Kubernetes 1.26

- CRI v1alpha2 is removed.  
Kubernetes 1.26 does not support CRI v1alpha2 any longer. Use CRI v1 (containerd version must be later than or equal to 1.5.0). containerd 1.5.x or earlier is not supported by Kubernetes 1.26. Update the containerd version to 1.6.x or later before upgrading kubelet to 1.26.

#### NOTE

The containerd version used by CCE is 1.6.14, which meets the requirements. If the existing nodes do not meet the containerd version requirements, reset them to the latest version.

- Alpha API for dynamic resource allocation is added.  
In Kubernetes 1.26, [Dynamic Resource Allocation](#) is added to request and share resources between pods and between containers in a pod. Resources can be initialized based on parameters provided by the user. This function is still in the alpha phase. You need to enable the DynamicResourceAllocation feature gate and the `resource.k8s.io/v1alpha1` API group. You need to install drivers for specific resources to be managed. For details, see [Kubernetes 1.26: Alpha API for Dynamic Resource Allocation](#).
- The non-graceful node shutdown feature goes to beta.  
In Kubernetes 1.26, the non-graceful node shutdown feature goes to beta and is enabled by default. A node shutdown can be graceful only if the kubelet's node shutdown manager can detect the upcoming node shutdown action. For details, see [Non-graceful node shutdown handling](#).
- Passing pod `fsGroup` to CSI drivers during mounting is supported.  
In Kubernetes 1.22, delegation of `fsGroup` to CSI drivers was first introduced as an alpha feature. In Kubernetes 1.25, it moved to beta. In Kubernetes 1.26, this feature enters the official release phase. For details, see [Delegating volume permission and ownership change to CSI driver](#).
- Pod scheduling readiness is introduced.  
Kubernetes 1.26 introduces a new feature `schedulingGates`, which enables the scheduler to detect when pod scheduling can be performed. For details, see [Pod Scheduling Readiness](#).
- CPU manager is officially released.  
The CPU manager is a part of kubelet. Since Kubernetes 1.10, it has moved to [beta](#). The CPU manager can allocate exclusive CPUs to containers. This feature is stable in Kubernetes 1.26. For details, see [Control CPU Management Policies on the Node](#).
- Kubernetes traffic engineering is advanced.  
[Internal node-local traffic optimization](#) and [EndpointSlice conditions](#) are upgraded to the official release version. [ProxyTerminatingEndpoints](#) moves to beta.
- Cross-namespace volume data sources are supported.

This feature allows you to specify a data source that belongs to different namespaces for a PVC. This feature is in the alpha phase. For details, see [Cross namespace data sources](#).

- Retroactive default StorageClass assignment moves to beta.

In Kubernetes 1.25, an alpha feature was introduced to change the way how a default StorageClass is allocated to a PVC. After this feature is enabled, you no longer need to create a default StorageClass and then create a PVC to assign the class. Additionally, any PVCs without a StorageClass assigned can be updated later. This feature moves to beta in Kubernetes 1.26. For details, see [Retroactive default StorageClass assignment](#).

- PodDisruptionBudget allows users to specify the eviction policies for unhealthy pods.

You are allowed to specify unhealthy pod eviction policies for [PodDisruptionBudget](#) (PDB). This feature helps ensure node availability during node management. This feature is in the beta phase. For details, see [Unhealthy Pod Eviction Policy](#).

- The number of Horizontal Pod Autoscaler (HPA) can be configured. **kube-controller-manager** allows **--concurrent-horizontal-pod-autoscaler-syncs** to configure the number of worker nodes of the pod autoscaler for horizontal scaling.

## Deprecations and Removals

### Kubernetes 1.27

- In Kubernetes 1.27, the feature gates that are used for volume extension and in the GA status, including `ExpandCSIVolumes`, `ExpandInUsePersistentVolumes`, and `ExpandPersistentVolumes` are removed and can no longer be referenced in the **--feature-gates** flag.
- The **--master-service-namespace** parameter is removed. This parameter specifies where to create a Service named **kubernetes** to represent the API server. This parameter was deprecated in Kubernetes 1.26 and is removed from Kubernetes 1.27.
- The `ControllerManagerLeaderMigration` feature gate is removed. [Leader Migration](#) provides a mechanism for HA clusters to safely migrate "cloud specific" controllers using a resource lock shared between `kube-controller-manager` and `cloud-controller-manager` when upgrading the replicated control plane. This feature has been enabled unconditionally since its release in Kubernetes 1.24. In Kubernetes 1.27, this feature is removed.
- The **--enable-taint-manager** parameter is removed. The feature that it supports, taint-based eviction, is enabled by default and will continue to be implicitly enabled when the flag is removed.
- The **--pod-eviction-timeout** parameter is removed from `kube-controller-manager`.
- The `CSIMigration` feature gate is removed. The [CSI migration](#) program allows smooth migration from the in-tree volume plug-ins to the out-of-tree CSI drivers. This feature was officially released in Kubernetes 1.16.
- The `CSIInlineVolume` feature gate is removed. The feature ([CSI Ephemeral Volume](#)) allows CSI volumes to be specified directly in the pod specification for ephemeral use cases. They can be used to inject arbitrary states, such as

configuration, secrets, identity, variables, or similar information, directly inside the pod using a mounted volume. This feature graduated to GA in Kubernetes 1.25 and is removed in Kubernetes 1.27.

- The EphemeralContainers feature gate is removed. For Kubernetes 1.27, API support for ephemeral containers is unconditionally enabled.
- The LocalStorageCapacityIsolation feature gate is removed. This feature gate (**Local Ephemeral Storage Capacity Isolation**) moved to GA in Kubernetes 1.25. The feature provides support for capacity isolation of local ephemeral storage between pods, such as emptyDir volumes, so that a pod can be limited in its consumption of shared resources. kubelet will evict a pod if its consumption of local ephemeral storage exceeds the configured limit.
- The NetworkPolicyEndPort feature gate is removed. In Kubernetes 1.25, **endPort** in NetworkPolicy moved to GA. NetworkPolicy providers that support the **endPort** field can be used to specify a range of ports to apply NetworkPolicy.
- The StatefulSetMinReadySeconds feature gate is removed. For a pod that is part of a StatefulSet, Kubernetes marks the pod as read-only when the pod is available (and passes the check) at least within the period specified in **minReadySeconds**. This feature was officially released in Kubernetes 1.25. It is locked to **true** and removed from Kubernetes 1.27.
- The IdentifyPodOS feature gate is removed. If this feature is enabled, you can specify an OS for a pod. It has been stable since Kubernetes 1.25. This feature is removed from Kubernetes 1.27.
- The DaemonSetUpdateSurge feature gate is removed. In Kubernetes 1.25, this feature was stable. It was implemented to minimize DaemonSet downtime during deployment, but it is removed from Kubernetes 1.27.
- The **--container-runtime** parameter is removed. kubelet accepts a deprecated parameter **--container-runtime**, and the only valid value will be **remote** after the dockershim code is removed. This parameter was deprecated in 1.24 and later versions and is removed from Kubernetes 1.27.

### Kubernetes 1.26

- HorizontalPodAutoscaler API for v2beta2 is removed.  
The autoscaling/v2beta2 API of HorizontalPodAutoscaler is no longer available in Kubernetes 1.26. For details, see [Removed APIs by release](#). Use autoscaling/v2 API instead.
- The **flowcontrol.apiserver.k8s.io/v1beta1** API is removed.  
In Kubernetes 1.26 and later versions, the API of the **flowcontrol.apiserver.k8s.io/v1beta1** version for FlowSchema and PriorityLevelConfiguration is no longer served. For details, see [Removed APIs by release](#). The **flowcontrol.apiserver.k8s.io/v1beta2** version is available in Kubernetes 1.23 and later versions, and the **flowcontrol.apiserver.k8s.io/v1beta3** version is available in Kubernetes 1.26 and later versions.
- The cloud service vendors' in-tree storage drivers are removed.
- The kube-proxy userspace mode is removed.  
The deprecated userspace mode is no longer supported by Linux or Windows. Linux users can use iptables or IPVS, and Windows users can use the KernelSpace mode. Errors are returned if you use **--mode userspace**.

- Windows winkernel kube-proxy no longer supports Windows HNS v1 APIs.
- **--prune-whitelist** flag is deprecated.  
The **--prune-whitelist** flag is **deprecated** and replaced by **--prune-allowlist** to support **Inclusive Naming Initiative**. This deprecated flag will be completely removed in later versions.
- The DynamicKubeletConfig feature gate is removed.  
The kubelet configuration of nodes can be dynamically updated through the API. The feature gate is removed from the kubelet in Kubernetes 1.24 and removed from the API server in Kubernetes 1.26. This simplifies the code and improves stability. It is recommended that you modify the kubelet configuration file instead and then restart the kubelet. For details, see [Remove DynamicKubeletConfig feature gate from the code](#).
- A kube-apiserver command line parameter is removed.  
The **--master-service-namespace** parameter is deprecated. It is unused in the API Server.
- Several **kubectl run** parameters are deprecated.  
Several unused kubectl subcommands are marked as **deprecated** and will be removed in later versions. These subcommands include **--cascade**, **--filename**, **--force**, **--grace-period**, **--kustomize**, **--recursive**, **--timeout**, and **--wait**.
- Some command line parameters related to logging are removed.  
Some logging-related command line parameters are **removed**. These parameters were **deprecated** in earlier versions.

## Enhanced Kubernetes 1.27 on CCE

During a version maintenance period, CCE periodically updates Kubernetes 1.27 and provides enhanced functions.

For details about cluster version updates, see [Release Notes for CCE Cluster Versions](#).

## References

For more details about the performance comparison and function evolution between Kubernetes 1.27 and other versions, see the following documents:

- [Kubernetes v1.27 Release Notes](#)
- [Kubernetes v1.26 Release Notes](#)

### 2.1.2.4 Kubernetes 1.25 Release Notes

CCE has passed the Certified Kubernetes Conformance Program and is a certified Kubernetes offering. This section describes the changes made in Kubernetes 1.25 compared with Kubernetes 1.23.

## Indexes

- [New Features](#)
- [Deprecations and Removals](#)

- [Enhanced Kubernetes 1.25 on CCE](#)
- [References](#)

## New Features

### Kubernetes 1.25

- Pod Security Admission is stable. PodSecurityPolicy is deprecated. PodSecurityPolicy is replaced by Pod Security Admission. For details about the migration, see [Migrate from PodSecurityPolicy to the Built-In PodSecurity Admission Controller](#).
- The ephemeral container is stable. An [ephemeral container](#) is a container that runs temporarily in an existing pod. It is useful for troubleshooting, especially when kubectl exec cannot be used to check a container that breaks down or its image lacks a debugging tool.
- Support for cgroups v2 enters the stable phase. Kubernetes supports cgroups v2. cgroups v2 provides some improvements over cgroup v1. For details, see [About cgroup v2](#).
- SeccompDefault moves to beta. To enable this feature, add the startup parameter `--seccomp-default=true` to kubelet. In this way, `seccomp` is set to `RuntimeDefault` by default, improving system security. Clusters of v1.25 no longer support `seccomp.security.alpha.kubernetes.io/pod` and `container.seccomp.security.alpha.kubernetes.io/annotation`. Replace them with the `securityContext.seccompProfile` field in pods or containers. For details, see [Configure a Security Context for a Pod or Container](#).

#### NOTE

After this feature is enabled, the system calls required by the application may be restricted by the runtime. Ensure that the debugging is performed in the test environment, so that application is not affected.

- The EndPort in the network policy moves to stable. EndPort in Network Policy is stable. This feature is incorporated in version 1.21. EndPort is added to NetworkPolicy. You can specify a port range.
- Local ephemeral storage capacity isolation is stable. This feature provides support for capacity isolation of local ephemeral storage between pods, such as EmptyDir. If a pod's consumption of shared resources exceeds the limit, it will be evicted.
- The CRD verification expression language moves to beta. This makes it possible to declare how to validate custom resources using [CEL](#). For details, see [Extend the Kubernetes API with CustomResourceDefinitions](#).
- KMS v2 APIs are introduced. The KMS v2 alpha1 API is introduced to add performance, rotation, and observability improvements. This API uses AES-GCM to replace AES-CBC and uses DEK to encrypt data at rest (Kubernetes Secrets). No additional operation is required during this process. Additionally, data can be read

through AES-GCM and AES-CBC. For details, see [Using a KMS provider for data encryption](#).

- Pod network readiness is introduced.  
Kubernetes 1.25 introduces Alpha support for PodHasNetwork. This status is in the **status** field of the pod. For details, see [Pod network readiness](#).
- The two features used for application rollout are stable.
  - In Kubernetes 1.25, **minReadySeconds** for StatefulSets is stable. It allows each pod to wait for an expected period of time to slow down the rollout of a StatefulSet. For details, see [Minimum ready seconds](#).
  - In Kubernetes 1.25, **maxSurge** for DaemonSets is stable. It allows a DaemonSet workload to run multiple instances of the same pod on one node during a rollout. This minimizes DaemonSet downtime for users. DaemonSet does not allow **maxSurge** and **hostPort** to be used at the same time because two active pods cannot share the same port on the same node. For details, see [Perform a Rolling Update on a DaemonSet](#).
- Alpha support for running pods with user namespaces is provided.  
This feature maps the **root** user in a pod to a non-zero ID outside the container. In this way, the container runs as the **root** user and the node runs as a regular unprivileged user. This feature is still in the internal test phase. The UserNamespacesStatelessPodsSupport gate needs to be enabled, and the container runtime must support this function. For details, see [Kubernetes 1.25: alpha support for running Pods with user namespaces](#).

#### Kubernetes 1.24

- Dockershim is removed from kubelet.  
Dockershim was marked deprecated in Kubernetes 1.20 and officially removed from kubelet in Kubernetes 1.24. If you want to use Docker container, switch to cri-dockerd or other runtimes that support CRI, such as containerd and CRI-O.  
For details about how to switch from Docker to containerd, see [Migrating Nodes from Docker to containerd](#).

#### NOTE

Check whether there are agents or applications that depend on Docker Engine. For example, if **docker ps**, **docker run**, and **docker inspect** are used, ensure that multiple runtimes are compatible and switch to the standard CRI.

- Beta APIs are disabled by default.  
The Kubernetes community found 90% cluster administrators did not care about the beta APIs and left them enabled. However, the beta features are not recommended because these APIs enabled in the production environment by default incur risks. Therefore, in 1.24 and later versions, beta APIs are disabled by default, but the existing beta APIs will retain the original settings.
- OpenAPI v3 is supported.  
In Kubernetes 1.24 and later versions, OpenAPI V3 is enabled by default.
- Storage capacity tracking is stable.  
In Kubernetes 1.24 and later versions, the CSIStorageCapacity API supports exposing the available storage capacity. This ensures that pods are scheduled to nodes with sufficient storage capacity, which reduces pod scheduling delay.

caused by volume creation and mounting failures. For details, see [Storage Capacity](#).

- gRPC container probe moves to beta.  
In Kubernetes 1.24 and later versions, the gRPC probe goes to beta. The feature gate GRPCContainerProbe is available by default. For details about how to use this probe, see [Configure Probes](#).
- LegacyServiceAccountTokenNoAutoGeneration is enabled by default.  
LegacyServiceAccountTokenNoAutoGeneration moves to beta. By default, this feature is enabled, where no secret token is automatically generated for a service account. To use a token that never expires, create a secret to hold the token. For details, see [Service account token Secrets](#).
- IP address conflict is prevented.  
In Kubernetes 1.24, [an IP address pool is soft reserved for the static IP addresses of Services](#). After you manually enable this function, Service IP addresses will be automatically from the IP address pool to minimize IP address conflict.
- Clusters are compiled based on Go 1.18.  
Kubernetes clusters of versions later than 1.24 are compiled based on Go 1.18. By default, the SHA-1 hash algorithm, such as SHA1WithRSA and ECDSAWithSHA1, is no longer supported for certificate signature verification. Use the certificate generated by the SHA256 algorithm instead.
- The maximum number of unavailable StatefulSet replicas is configurable.  
In Kubernetes 1.24 and later versions, the **maxUnavailable** parameter can be configured for StatefulSets so that pods can be stopped more quickly during a rolling update.
- Alpha support for non-graceful node shutdown is introduced.  
The non-graceful node shutdown is introduced as alpha in Kubernetes v1.24. A node shutdown is considered graceful only if kubelet's node shutdown manager can detect the upcoming node shutdown action. For details, see [Non-graceful node shutdown handling](#).

## Deprecations and Removals

### Kubernetes 1.25

- The iptables chain ownership is cleared up.  
Kubernetes typically creates iptables chains to ensure data packets can be sent to the destination. These iptables chains and their names are for internal use only. These chains were never intended to be part of any Kubernetes API/ABI guarantees. For details, see [Kubernetes's IPTables Chains Are Not API](#).  
In versions later than Kubernetes 1.25, Kubelet uses IPTablesCleanup to migrate the Kubernetes-generated iptables chains used by the components outside of Kubernetes in phases so that iptables chains such as KUBE-MARK-DROP, KUBE-MARK-MASQ, and KUBE-POSTROUTING will not be created in the NAT table. For more details, see [Cleaning Up IPTables Chain Ownership](#).
- In-tree volume drivers from cloud service vendors are removed.

### Kubernetes 1.24

- In Kubernetes 1.24 and later versions, `Service.Spec.LoadBalancerIP` is deprecated because it cannot be used for dual-stack protocols. Instead, use custom annotations.
- In Kubernetes 1.24 and later versions, the `--address`, `--insecure-bind-address`, `--port`, and `--insecure-port=0` parameters are removed from `kube-apiserver`.
- In Kubernetes 1.24 and later versions, startup parameters `--port=0` and `--address` are removed from `kube-controller-manager` and `kube-scheduler`.
- In Kubernetes 1.24 and later versions, `kube-apiserver --audit-log-version` and `--audit-webhook-version` support only `audit.k8s.io/v1`. In Kubernetes 1.24, `audit.k8s.io/v1[alpha|beta]1` is removed, and only `audit.k8s.io/v1` can be used.
- In Kubernetes 1.24 and later versions, the startup parameter `--network-plugin` is removed from kubelet. This Docker-specific parameter is available only when the container runtime environment is `Docker` and it is deleted with Dockershim.
- In Kubernetes 1.24 and later versions, dynamic log clearance has been discarded and removed accordingly. A log filter is introduced to the logs of all Kubernetes system components to prevent sensitive information from being leaked through logs. However, this function may block logs and therefore is discarded. For more details, see [Dynamic log sanitization](#) and [KEP-1753](#).
- VolumeSnapshot v1beta1 CRD is discarded in Kubernetes 1.20 and removed in Kubernetes 1.24. Use VolumeSnapshot v1 instead.
- In Kubernetes 1.24 and later versions, `service annotation tolerate-unready-endpoints` discarded in Kubernetes 1.11 is replaced by `Service.spec.publishNotReadyAddresses`.
- In Kubernetes 1.24 and later versions, the `metadata.clusterName` field is discarded and will be deleted in the next version.
- In Kubernetes 1.24 and later versions, the logic for kube-proxy to listen to NodePorts is removed. If NodePorts conflict with `kernel net.ipv4.ip_local_port_range`, TCP connections may fail occasionally, which leads to a health check failure or service exception. Before the upgrade, ensure that cluster NodePorts do not conflict with `net.ipv4.ip_local_port_range` of all nodes in the cluster. For more details, see [Kubernetes PR](#).

## Enhanced Kubernetes 1.25 on CCE

During a version maintenance period, CCE periodically updates Kubernetes 1.25 and provides enhanced functions.

For details about cluster version updates, see [Release Notes for CCE Cluster Versions](#).

## References

For more details about the performance comparison and function evolution between Kubernetes 1.25 and other versions, see the following documents:

- [Kubernetes v1.25 Release Notes](#)
- [Kubernetes v1.24 Release Notes](#)



## 2.1.2.5 Kubernetes 1.23 Release Notes

CCE has passed the Certified Kubernetes Conformance Program and is a certified Kubernetes offering. This section describes the updates in CCE Kubernetes 1.23.

### Resource Changes and Deprecations

#### Kubernetes 1.23 Release Notes

- FlexVolume is deprecated. Use CSI.
- HorizontalPodAutoscaler v2 is promoted to GA, and HorizontalPodAutoscaler API v2 is gradually stable in version 1.23. The HorizontalPodAutoscaler v2beta2 API is not recommended. Use the v2 API.
- **PodSecurity** moves to beta, replacing the deprecated PodSecurityPolicy. PodSecurity is an admission controller that enforces pod security standards on pods in the namespace based on specific namespace labels that set the enforcement level. PodSecurity is enabled by default in version 1.23.

#### Kubernetes 1.22 Release Notes

- Ingresses no longer support networking.k8s.io/v1beta1 and extensions/v1beta1 APIs. If you use the API of an earlier version to manage ingresses, an application cannot be exposed to external services. Use networking.k8s.io/v1.
- CustomResourceDefinitions no longer support the apiextensions.k8s.io/v1beta1 API. If you use the API of an earlier version to create a CRD, the creation will fail, which affects the controller that reconciles this CRD. Use apiextensions.k8s.io/v1.
- ClusterRoles, ClusterRoleBindings, Roles, and RoleBindings no longer support the rbac.authorization.k8s.io/v1beta1 API. If you use the API of an earlier version to manage RBAC resources, application permissions control is affected and even cannot work in the cluster. Use rbac.authorization.k8s.io/v1.
- The Kubernetes release cycle is changed from four releases a year to three releases a year.
- StatefulSets support **minReadySeconds**.
- During scale-in, pods are randomly selected and deleted based on the pod UID by default (LogarithmicScaleDown). This feature enhances the randomness of the pods to be deleted and alleviates the problems caused by pod topology spread constraints. For more information, see [KEP-2185](#) and [issue 96748](#).
- The **BoundServiceAccountTokenVolume** feature is stable, which has changed the method of mounting tokens into pods for enhanced token security of the service account. This feature is enabled by default in Kubernetes clusters of v1.21 and later versions.

### References

For more details about the performance comparison and function evolution between Kubernetes 1.23 and other versions, see the following documents:

- [Kubernetes v1.23 Release Notes](#)
- [Kubernetes v1.22 Release Notes](#)

## 2.1.2.6 Kubernetes 1.21 (EOM) Release Notes

CCE has passed the Certified Kubernetes Conformance Program and is a certified Kubernetes offering. This section describes the updates in CCE Kubernetes 1.21.

### Resource Changes and Deprecations

#### Kubernetes 1.21 Release Notes

- CronJob is now in the stable state, and the version number changes to batch/v1.
- The immutable Secret and ConfigMap have now been upgraded to the stable state. A new immutable field is added to these objects to reject changes. The rejection protects clusters from accidental updates that may cause application outages. As these resources are immutable, kubelet does not monitor or poll for changes. This reduces the load of kube-apiserver and improves scalability and performance of your clusters. For more information, see [Immutable ConfigMaps](#).
- Graceful node shutdown has been upgraded to the test state. With this update, kubelet can detect that a node is shut down and gracefully terminate the pods on the node. Prior to this update, when the node was shut down, its pod did not follow the expected termination lifecycle, which caused workload problems. Now kubelet can use systemd to detect the systems that are about to be shut down and notify the running pods to terminate them gracefully.
- For a pod with multiple containers, you can use [kubectl.kubernetes.io/](#) to pre-select containers.
- PodSecurityPolicy is deprecated. For details, see <https://kubernetes.io/blog/2021/04/06/podsecuritypolicy-deprecation-past-present-and-future/>.
- The [BoundServiceAccountTokenVolume](#) feature is in beta testing, which has changed the method of mounting tokens into pods for enhanced token security of the service account. This feature is enabled by default in Kubernetes clusters of v1.21 and later versions.

#### Kubernetes 1.20 Release Notes

- The API priority and fairness have reached the test state and are enabled by default. This allows kube-apiserver to classify incoming requests by priority. For more information, see [API Priority and Fairness](#).
- The bug of **exec probe timeouts** is fixed. Before this bug is fixed, the exec probe does not consider the **timeoutSeconds** field. Instead, the probe will run indefinitely, even beyond its configured deadline. It will stop until the result is returned. Now, if no value is specified, the default value is used, that is, one second. If the detection time exceeds one second, the application health check may fail. During the upgrade, update the **timeoutSeconds** field for the applications that use this feature. The repair provided by the newly introduced ExecProbeTimeout feature gating enables the cluster operator to restore the previous behavior, but this behavior will be locked and removed in later versions.
- RuntimeClass enters the stable state. RuntimeClass provides a mechanism to support multiple runtimes in a cluster and expose information about the container runtime to the control plane.

- kubectl debugging has reached the test state. kubectl debugging provides support for common debugging workflows.
- dockershim was marked as deprecated in Kubernetes 1.20. Currently, you can continue to use Docker in the cluster. This change is irrelevant to the container image used by clusters. You can still use Docker to build your images. For more information, see [Dockershim Deprecation FAQ](#).

## References

For more details about the performance comparison and function evolution between Kubernetes 1.21 and other versions, see the following documents:

- [Kubernetes v1.21 Release Notes](#)
- [Kubernetes v1.20 Release Notes](#)

### 2.1.2.7 Kubernetes 1.19 (EOM) Release Notes

CCE has passed the Certified Kubernetes Conformance Program and is a certified Kubernetes offering. This section describes the updates in CCE Kubernetes 1.19.

## Resource Changes and Deprecations

### Kubernetes v1.19 Release Notes

- vSphere in-tree volumes can be migrated to vSphere CSI drivers. The in-tree vSphere Volume plugin is no longer used and will be deleted in later versions.
- **apiextensions.k8s.io/v1beta1** has been deprecated. You are advised to use **apiextensions.k8s.io/v1**.
- **apiregistration.k8s.io/v1beta1** has been deprecated. You are advised to use **apiregistration.k8s.io/v1**.
- **authentication.k8s.io/v1beta1** and **authorization.k8s.io/v1beta1** have been deprecated and will be removed from Kubernetes 1.22. You are advised to use **authentication.k8s.io/v1** and **authorization.k8s.io/v1**.
- **autoscaling/v2beta1** has been deprecated. You are advised to use **autoscaling/v2beta2**.
- **coordination.k8s.io/v1beta1** has been deprecated in Kubernetes 1.19 and will be removed from version 1.22. You are advised to use **coordination.k8s.io/v1**.
- kube-apiserver: The **componentstatus** API has been deprecated.
- kubeadm: The **kubeadm config view** command has been deprecated and will be deleted in later versions. Use **kubectl get cm -o yaml -n kube-system kubeadm-config** to directly obtain the kubeadm configuration.
- kubeadm: The **kubeadm alpha kubelet config enable-dynamic** command has been deprecated.
- kubeadm: The **--use-api** flag in the **kubeadm alpha certs renew** command has been deprecated.
- Kubernetes no longer supports **hyperkube** image creation.
- The **--export** flag is removed from the **kubectl get** command.
- The alpha feature **ResourceLimitsPriorityFunction** has been deleted.

- **storage.k8s.io/v1beta1** has been deprecated. You are advised to use **storage.k8s.io/v1**.

### Kubernetes v1.18 Release Notes

- kube-apiserver
  - All resources in the **apps/v1beta1** and **apps/v1beta2** API versions are no longer served. You can use the **apps/v1** API version.
  - DaemonSets, Deployments, and ReplicaSets in the **extensions/v1beta1** API version are no longer served. You can use the **apps/v1** API version.
  - NetworkPolicies in the **extensions/v1beta1** API version are no longer served. You can use the **networking.k8s.io/v1** API version.
  - PodSecurityPolicies in the **extensions/v1beta1** API version are no longer served. Migrate to use the **policy/v1beta1** API version.
- kubelet
  - **--redirect-container-streaming** is not recommended and will be deprecated in v1.20.
  - The resource measurement endpoint **/metrics/resource/v1alpha1** and all measurement standards under this endpoint have been deprecated. Use the measurement standards under the endpoint **/metrics/resource** instead:
    - `scrape_error --> scrape_error`
    - `node_cpu_usage_seconds_total --> node_cpu_usage_seconds`
    - `node_memory_working_set_bytes --> node_memory_working_set_bytes`
    - `container_cpu_usage_seconds_total --> container_cpu_usage_seconds`
    - `container_memory_working_set_bytes --> container_memory_working_set_bytes`
    - `scrape_error --> scrape_error`
  - In future releases, kubelet will no longer create the target directory **CSI NodePublishVolume** according to the CSI specifications. You may need to update the CSI driver accordingly to correctly create and process the target path.
- kube-proxy
  - You are not advised to use the **--healthz-port** and **--metrics-port** flags. Use **--healthz-bind-address** and **--metrics-bind-address** instead.
  - The **EndpointSliceProxying** function option is added to control the use of EndpointSlices in kube-proxy. This function is disabled by default.
- kubeadm
  - The **--kubelet-version** flag of **kubeadm upgrade node** has been deprecated and will be deleted in later versions.
  - The **--use-api** flag in the **kubeadm alpha certs renew** command has been deprecated.
  - kube-dns has been deprecated and will no longer be supported in future versions.

- The ClusterStatus structure in the kubeadm-config ConfigMap has been deprecated and will be deleted in later versions.
- kubectl
  - You are not advised to use boolean and unset values for **--dry-run.server|client|none** is used in the new version.
  - **--server-dry-run** has been deprecated for **kubectl apply** and replaced by **--dry-run=server**.
- add-ons

The cluster-monitoring add-on is deleted.

- kube-scheduler
  - The **scheduling\_duration\_seconds** metric has been deprecated.
  - The **scheduling\_algorithm\_predicate\_evaluation\_seconds** and **scheduling\_algorithm\_priority\_evaluation\_seconds** metrics are no longer used and are replaced by **framework\_extension\_point\_duration\_seconds[extension\_point="Filter"]** and **framework\_extension\_point\_duration\_seconds[extension\_point="Score"]**.
  - The scheduler policy AlwaysCheckAllPredicates has been deprecated.
- Other changes
  - The k8s.io/node-api component is no longer updated. Instead, you can use the **RuntimeClass** type in **k8s.io/api** and the generated clients in **k8s.io/client-go**.
  - The **client** label has been deleted from **apiserver\_request\_total**.

## References

For more details about the performance comparison and function evolution between Kubernetes 1.19 and other versions, see the following documents:

- [Kubernetes v1.19.0 Release Notes](#)
- [Kubernetes v1.18.0 Release Notes](#)

### 2.1.2.8 Kubernetes 1.17 (EOM) Release Notes

CCE has passed the Certified Kubernetes Conformance Program and is a certified Kubernetes offering. This section describes the updates in CCE Kubernetes 1.17.

## Resource Changes and Deprecations

- All resources in the **apps/v1beta1** and **apps/v1beta2** API versions are no longer served. Migrate to use the **apps/v1** API version.
- DaemonSets, Deployments, and ReplicaSets in the **extensions/v1beta1** API version are no longer served. You can use the **apps/v1** API version.
- NetworkPolicies in the **extensions/v1beta1** API version are no longer served. Migrate to use the **networking.k8s.io/v1** API version.
- PodSecurityPolicies in the **extensions/v1beta1** API version are no longer served. Migrate to use the **policy/v1beta1** API version.

- Ingresses in the **extensions/v1beta1** API version will no longer be served in v1.20. Migrate to use the **networking.k8s.io/v1beta1** API version.
- **PriorityClass** in the **scheduling.k8s.io/v1beta1** and **scheduling.k8s.io/v1alpha1** API versions is no longer served in v1.17. Migrate to use the **scheduling.k8s.io/v1** API version.
- The **event series.state** field in the **events.k8s.io/v1beta1** API version has been deprecated and will be removed from v1.18.
- **CustomResourceDefinition** in the **apiextensions.k8s.io/v1beta1** API version has been deprecated and will no longer be served in v1.19. Use the **apiextensions.k8s.io/v1** API version.
- **MutatingWebhookConfiguration** and **ValidatingWebhookConfiguration** in the **admissionregistration.k8s.io/v1beta1** API version have been deprecated and will no longer be served in v1.19. You can use the **admissionregistration.k8s.io/v1** API version.
- The **rbac.authorization.k8s.io/v1alpha1** and **rbac.authorization.k8s.io/v1beta1** API versions have been deprecated and will no longer be served in v1.20. Use the **rbac.authorization.k8s.io/v1** API version.
- The **CSINode** object of **storage.k8s.io/v1beta1** has been deprecated and will be removed in later versions.

## Other Deprecations and Removals

- **OutOfDisk node condition** is removed in favor of **DiskPressure**.
- The **scheduler.alpha.kubernetes.io/critical-pod** annotation is removed in favor of **priorityClassName**.
- **beta.kubernetes.io/os** and **beta.kubernetes.io/arch** have been deprecated in v1.14 and will be removed in v1.18.
- Do not use **--node-labels** to set labels prefixed with **kubernetes.io** and **k8s.io**. The **kubernetes.io/availablezone** label in earlier versions is removed in v1.17 and changed to **failure-domain.beta.kubernetes.io/zone**.
- The **beta.kubernetes.io/instance-type** is deprecated in favor of **node.kubernetes.io/instance-type**.
- Remove the **{kubelet\_root\_dir}/plugins** path.
- Remove the built-in cluster roles **system:csi-external-provisioner** and **system:csi-external-attacher**.

## References

For more details about the performance comparison and function evolution between Kubernetes 1.17 and other versions, see the following documents:

- [Kubernetes v1.17.0 Release Notes](#)
- [Kubernetes v1.16.0 Release Notes](#)

### 2.1.2.9 Kubernetes 1.15 (EOM) Release Notes

CCE has passed the Certified Kubernetes Conformance Program and is a certified Kubernetes offering. This section describes the updates in CCE Kubernetes 1.15.

To enable interoperability from one Kubernetes installation to the next, you must upgrade your Kubernetes clusters before the maintenance period ends.

## Description

CCE provides full-link component optimization and upgrade for Kubernetes v1.15, which includes two minor versions v1.15.11 and v1.15.6-r1.

## Resource Changes and Deprecations

- Ingress in the **extensions/v1beta1** API version has been deprecated. It will be no longer served from Kubernetes 1.19. You can use the **networking.k8s.io/v1beta1** API version.
- NetworkPolicy in the **extensions/v1beta1** API version will be officially suspended in 1.16. Migrate to use the **networking.k8s.io/v1** API version.
- PodSecurityPolicy in the **extensions/v1beta1** API version will be officially suspended in 1.16. Migrate to use the **policy/v1beta1** API version.
- DaemonSets, Deployments, and ReplicaSets in the **extensions/v1beta1**, **apps/v1beta1**, and **apps/v1beta2** API versions will not be served in 1.16. You can use the **apps/v1** API version.
- PriorityClass is upgraded to **scheduling.k8s.io/v1**, **scheduling.k8s.io/v1beta1**, and **scheduling.k8s.io/v1alpha1**. It will be deprecated in 1.17.
- The **series.state** field in the **events.k8s.io/v1beta1** Event API version has been deprecated and will be removed from 1.18.

## References

Changelog from v1.13 to v1.15

- Changelog from v1.14 to v1.15:  
<https://github.com/kubernetes/kubernetes/blob/master/CHANGELOG/CHANGELOG-1.15.md>
- Changelog from v1.13 to v1.14:  
<https://github.com/kubernetes/kubernetes/blob/master/CHANGELOG/CHANGELOG-1.14.md>

### 2.1.2.10 Kubernetes 1.13 (EOM) Release Notes

CCE has passed the Certified Kubernetes Conformance Program and is a certified Kubernetes offering. This section describes the updates in CCE Kubernetes 1.13.

**Table 2-2** v1.13 cluster description

Kubernetes (CCE Enhanced Version)	Description
v1.13.10-r0	<p><b>Highlights:</b></p> <ul style="list-style-type: none"> <li>● Arm nodes can be added to a CCE cluster.</li> <li>● The load balancer name is configurable.</li> <li>● Layer-4 load balancing supports health check, and layer-7 load balancing supports health check, allocation policy, and sticky session.</li> <li>● BMS nodes can be created in a CCE cluster (when the tunnel network model is used).</li> <li>● Ascend-accelerated nodes (powered by HiSilicon Ascend 310 AI processors) apply to scenarios such as image recognition, video processing, inference computing, and machine learning.</li> <li>● The docker baseSize is configurable.</li> <li>● Namespace affinity scheduling is supported.</li> <li>● User space can be partitioned in node data disks.</li> <li>● Cluster CPU management policies can be configured.</li> <li>● Nodes in a cluster can be configured across subnets (when the tunnel network mode is used).</li> </ul>
v1.13.7-r0	<p><b>Highlights:</b></p> <ul style="list-style-type: none"> <li>● Features of Kubernetes v1.13.7 are incorporated.</li> <li>● The network attachment definition is supported.</li> </ul>

## References

Changelog from v1.11 to v1.13

- Changelog from v1.12 to v1.13:  
<https://github.com/kubernetes/kubernetes/blob/master/CHANGELOG/CHANGELOG-1.13.md>
- Changelog from v1.11 to v1.12:  
<https://github.com/kubernetes/kubernetes/blob/master/CHANGELOG/CHANGELOG-1.12.md>

### 2.1.2.11 Kubernetes 1.11 (EOM) Release Notes

CCE has passed the Certified Kubernetes Conformance Program and is a certified Kubernetes offering. This section describes the updates in CCE Kubernetes 1.11.



**Table 2-3** v1.11 cluster description

Kubernetes (CCE Enhanced Version)	Description
v1.11.7-r2	<b>Highlights:</b> <ul style="list-style-type: none"> <li>• Support for GPU V100.</li> <li>• Support for permission management.</li> </ul>
v1.11.7-r0	<b>Highlights:</b> <ul style="list-style-type: none"> <li>• Features of Kubernetes v1.11.7 are incorporated.</li> <li>• Node pools, VMs, and Kunpeng clusters can be created.</li> <li>• BMS nodes can be created in a CCE cluster (when the VPC network model is used), and hybrid deployment of BMSs and VMs is supported.</li> <li>• Support for GPU V100.</li> <li>• AOM notifies users when alarms are generated for container clusters of v1.11.</li> <li>• Access type switching is supported for Services.</li> <li>• Service network segments can be configured.</li> <li>• The number of IP addresses allocated to a node in a cluster can be customized.</li> </ul>
v1.11.3-r2	<b>Highlights:</b> <ul style="list-style-type: none"> <li>• Clusters support IPv6 dual stack.</li> <li>• ELB load balancing algorithms: source IP hash and sticky sessions with backend servers.</li> </ul>
v1.11.3-r1	<b>Highlights:</b> <ul style="list-style-type: none"> <li>• Perl regular expressions can be used for matching ingress URLs.</li> </ul>
v1.11.3-r0	<b>Highlights:</b> <ul style="list-style-type: none"> <li>• Features of Kubernetes v1.11.3 are incorporated.</li> <li>• Master nodes of a cluster can be deployed across multiple AZs.</li> <li>• CCE works with SFS Turbo to provide container storage.</li> </ul>

## References

Changelog from v1.9 to v1.11

- Changelog from v1.10 to v1.11:  
<https://github.com/kubernetes/kubernetes/blob/master/CHANGELOG/CHANGELOG-1.11.md>

- Changelog from v1.9 to v1.10:  
<https://github.com/kubernetes/kubernetes/blob/master/CHANGELOG/CHANGELOG-1.10.md>

### 2.1.2.12 Release Notes for Kubernetes 1.9 (EOM) and Earlier Versions

CCE has passed the Certified Kubernetes Conformance Program and is a certified Kubernetes offering. This section describes the updates in CCE Kubernetes 1.9 and earlier versions.

**Table 2-4** Description of v1.9 and earlier version clusters

Kubernetes (CCE Enhanced Version)	Description
v1.9.10-r2	<p><b>Highlights:</b></p> <ul style="list-style-type: none"> <li>• ELB load balancing algorithms: source IP hash and sticky sessions with backend servers.</li> </ul>
v1.9.10-r1	<p><b>Highlights:</b></p> <ul style="list-style-type: none"> <li>• CCE works with SFS.</li> <li>• Enhanced ELBs can be automatically created for Services.</li> <li>• Transparent transmission of source IP addresses is supported for enhanced ELBs on the public network.</li> <li>• The maximum number of pods on a node can be set.</li> </ul>
v1.9.10-r0	<p><b>Highlights:</b></p> <ul style="list-style-type: none"> <li>• Use of ELB/ingress for Kubernetes clusters; new traffic control mechanism.</li> <li>• Features of Kubernetes v1.9.10 are incorporated.</li> <li>• Kubernetes RBAC capability authorization is supported.</li> </ul> <p><b>Fault rectification:</b></p> <ul style="list-style-type: none"> <li>• Occasional memory leak on nodes, which is caused by kernel cgroup bugs.</li> </ul>

Kubernetes (CCE Enhanced Version)	Description
v1.9.7-r1	<p><b>Highlights:</b></p> <ul style="list-style-type: none"> <li>• The mechanism for reporting PVC and PV events is enhanced. Events can be viewed on the PVC details page.</li> <li>• CCE works with a third-party authentication system.</li> <li>• Physical machines that use EulerOS 2.3 can be managed.</li> <li>• Data disk allocation can be user-defined.</li> <li>• Elastic Volume Service (EVS) disks are supported for BMSs.</li> <li>• InfiniBand NICs are supported for BMSs.</li> <li>• Nodes can be created using the CM-v3 API in BMS scenarios.</li> </ul>
v1.9.7-r0	<p><b>Highlights:</b></p> <ul style="list-style-type: none"> <li>• The Docker version of new clusters is upgraded to v17.06.</li> <li>• DNS cascading is supported.</li> <li>• Add-ons can be managed.</li> <li>• Features of Kubernetes v1.9.7 are incorporated.</li> <li>• The HTTPS of layer-7 ingress is supported.</li> <li>• StatefulSets can be migrated, scheduled, updated, and upgraded.</li> </ul>
v1.9.2-r3	<p><b>Highlights:</b></p> <ul style="list-style-type: none"> <li>• Cluster nodes that use CentOS 7.4 can be created or managed.</li> <li>• DNAT Services are supported.</li> <li>• NetworkPolicy APIs are provided.</li> <li>• Multiple ports can be configured for a Kubernetes Service that uses an ELB.</li> </ul> <p><b>Fault rectification:</b></p> <ul style="list-style-type: none"> <li>• Incomplete pod resource recycling caused by a disconnection with kube-apiserver</li> <li>• Data inaccuracy during auto node scaling</li> </ul>

Kubernetes (CCE Enhanced Version)	Description
v1.9.2-r2	<p><b>Highlights:</b></p> <ul style="list-style-type: none"> <li>• Custom health check ports can be configured for classic load balancers.</li> <li>• Performance of classic load balancers is enhanced.</li> <li>• Kubernetes Service ports can be configured for layer-4 load balancing.</li> </ul> <p><b>Fault rectification:</b></p> <ul style="list-style-type: none"> <li>• Bugs in network add-ons, which cause deadlocks in health checks.</li> <li>• A limited number of HAProxy connections in an HA cluster.</li> </ul>
v1.9.2-r1	<p><b>Highlights:</b></p> <ul style="list-style-type: none"> <li>• Features of Kubernetes v1.9.2 are incorporated.</li> <li>• Cluster nodes support CentOS 7.1.</li> <li>• GPU nodes are supported and GPU resource use can be restricted.</li> <li>• The web-terminal add-on is supported.</li> </ul>
v1.7.3-r13	<p><b>Highlights:</b></p> <ul style="list-style-type: none"> <li>• The Docker version of new clusters is upgraded to v17.06.</li> <li>• DNS cascading is supported.</li> <li>• Add-ons can be managed.</li> <li>• The mechanism for reporting PVC and PV events is enhanced.</li> <li>• OBS is supported for BMS clusters.</li> </ul>
v1.7.3-r12	<p><b>Highlights:</b></p> <ul style="list-style-type: none"> <li>• Cluster nodes that use CentOS 7.4 can be created or managed.</li> <li>• DNAT Services are supported.</li> <li>• NetworkPolicy APIs are provided.</li> <li>• Multiple ports can be configured for a Kubernetes Service that uses an ELB.</li> </ul> <p><b>Fault rectification:</b></p> <ul style="list-style-type: none"> <li>• Incomplete pod resource recycling caused by a disconnection with kube-apiserver</li> <li>• Data inaccuracy during auto node scaling</li> <li>• The event aging period prompt is modified. The cluster aging period is 1 hour.</li> </ul>

Kubernetes (CCE Enhanced Version)	Description
v1.7.3-r11	<p><b>Highlights:</b></p> <ul style="list-style-type: none"> <li>● Custom health check ports can be configured for classic load balancers.</li> <li>● Performance of classic load balancers is enhanced.</li> <li>● Kubernetes Service ports can be configured for layer-4 load balancing.</li> <li>● Namespaces can be deleted.</li> <li>● EVS disks can be unbound.</li> <li>● Migration policies can be configured.</li> </ul> <p><b>Fault rectification:</b></p> <ul style="list-style-type: none"> <li>● Bugs in network add-ons, which cause deadlocks in health checks.</li> <li>● A limited number of HAProxy connections in an HA cluster.</li> </ul>
v1.7.3-r10	<p><b>Highlights:</b></p> <ul style="list-style-type: none"> <li>● Overlay L2 container networks are supported.</li> <li>● Cluster nodes can be GPU-accelerated VMs.</li> <li>● Cluster nodes support CentOS 7.1 and the operating system can be selected.</li> <li>● Windows clusters support ELBs.</li> <li>● CCE allows exporting data from SFS.</li> <li>● BMS clusters support SFS.</li> </ul>
v1.7.3-r9	<p><b>Highlights:</b></p> <ul style="list-style-type: none"> <li>● Cross-AZ deployment is supported for workloads.</li> <li>● Containers support OBS.</li> <li>● Layer-7 load balancing is supported.</li> <li>● Windows clusters support EVS.</li> <li>● Device mapper in direct-lvm mode is supported in BMS scenarios.</li> </ul>
v1.7.3-r8	<p><b>Highlights:</b></p> <ul style="list-style-type: none"> <li>● Auto scaling is supported for cluster nodes.</li> <li>● Arm nodes can be managed.</li> </ul>

Kubernetes (CCE Enhanced Version)	Description
v1.7.3-r7	<b>Highlights:</b> <ul style="list-style-type: none"> <li>• SUSE 12 sp2 nodes can be managed in the container clusters (in the tunnel network mode).</li> <li>• Docker supports the device mapper in direct-lvm mode.</li> <li>• Clusters support the dashboard add-on.</li> <li>• Windows clusters can be created.</li> </ul>
v1.7.3-r6	<b>Highlights:</b> <ul style="list-style-type: none"> <li>• Native EVS APIs are supported for clusters.</li> </ul>
v1.7.3-r5	<b>Highlights:</b> <ul style="list-style-type: none"> <li>• HA clusters can be created.</li> </ul> <b>Fault rectification:</b> <ul style="list-style-type: none"> <li>• Container network disconnection after a node restart</li> </ul>
v1.7.3-r4	<b>Highlights:</b> <ul style="list-style-type: none"> <li>• Cluster performance is enhanced.</li> <li>• Interconnection with ELB is allowed in BMS scenarios.</li> </ul>
v1.7.3-r3	<b>Highlights:</b> <ul style="list-style-type: none"> <li>• Storage can be attached to kernel-based virtual machines (KVMs).</li> </ul>
v1.7.3-r2	<b>Highlights:</b> <ul style="list-style-type: none"> <li>• SFS is supported to provide container storage.</li> <li>• Custom logs can be configured for workloads.</li> <li>• Graceful scaling-in is supported for workloads.</li> </ul> <b>Fault rectification:</b> <ul style="list-style-type: none"> <li>• Expiration of Access Key ID/Secret Access Key (AK/SK) of container storage volumes.</li> </ul>
v1.7.3-r1	<b>Highlights:</b> <ul style="list-style-type: none"> <li>• External domain names can be resolved by kube-dns.</li> </ul>
v1.7.3-r0	<b>Highlights:</b> <ul style="list-style-type: none"> <li>• Features of Kubernetes v1.7.3 are incorporated.</li> <li>• Elastic Load Balance (ELB) is supported.</li> <li>• Storage can be attached to Xen VMs.</li> <li>• EVS is supported to provide container storage.</li> </ul>

## 2.1.3 Patch Version Release Notes

### Version 1.29

Table 2-5 Release notes for the v1.29 patch

CCE Cluster Patch Version	Kubernetes Version	Feature Updates	Optimization	Vulnerability Fixing
v1.29.2-r4	<a href="#">v1.29.3</a>	None	The stability of ELB has been improved during upgrades that span across multiple versions.	Fixed some security issues.

CCE Cluster Patch Version	Kubernetes Version	Feature Updates	Optimization	Vulnerability Fixing
v1.29.2-r0	<a href="#">v1.29.3</a>	<ul style="list-style-type: none"> <li>• CCE ingresses support traffic distribution based on custom HTTP headers.</li> <li>• Scaling priority policies can be configured for third-party workloads.</li> <li>• You can configure a security group for a pod using annotations. This feature is only available for CCE Turbo clusters.</li> <li>• You can bind an existing EIP to a pod. This feature is only available for CCE Turbo clusters.</li> </ul>	<ul style="list-style-type: none"> <li>• An in-progress node drainage can be canceled.</li> <li>• When updating a node pool, you can change its agency name, prefix, and suffix.</li> <li>• Kubernetes labels and taints of a node are retained after the node is reset.</li> <li>• Both the Kubernetes service account token volume projection and the load scaling controller can be configured.</li> </ul>	Fixed some security issues.
v1.29.1-r0	<a href="#">v1.29.1</a>	CCE clusters of v1.29 are released for the first time. For more information, see <a href="#">Kubernetes 1.29 Release Notes</a> .	None	None



## Version 1.28

**Table 2-6** Release notes for the v1.28 patch

CCE Cluster Patch Version	Kubernetes Version	Feature Updates	Optimization	Vulnerability Fixing
v1.28.6-r4	<a href="#">v1.28.8</a>	None	The stability of ELB has been improved during upgrades that span across multiple versions.	Fixed some security issues.

CCE Cluster Patch Version	Kubernetes Version	Feature Updates	Optimization	Vulnerability Fixing
v1.28.6-r0	<a href="#">v1.28.8</a>	<ul style="list-style-type: none"> <li>• CCE ingresses support traffic distribution based on custom HTTP headers.</li> <li>• Scaling priority policies can be configured for third-party workloads.</li> <li>• You can configure a security group for a pod using annotations. This feature is only available for CCE Turbo clusters.</li> <li>• You can bind an existing EIP to a pod. This feature is only available for CCE Turbo clusters.</li> </ul>	<ul style="list-style-type: none"> <li>• An in-progress node drainage can be canceled.</li> <li>• When updating a node pool, you can change its agency name, prefix, and suffix.</li> <li>• Kubernetes labels and taints of a node are retained after the node is reset.</li> <li>• Both the Kubernetes service account token volume projection and the load scaling controller can be configured.</li> </ul>	Fixed some security issues.

CCE Cluster Patch Version	Kubernetes Version	Feature Updates	Optimization	Vulnerability Fixing
v1.28.4-r0	<a href="#">v1.28.5</a>	Docker can be selected when you create a node.	The configurations of frequently used cluster parameters and node pool parameters are publicly available.	Fixed some security issues.
v1.28.3-r0	<a href="#">v1.28.3</a>	LoadBalancer Services and ingresses allow you to: <ul style="list-style-type: none"> <li>• Configure SNI.</li> <li>• Enable HTTP/2.</li> <li>• Configure idle timeout, request timeout, and response timeout.</li> </ul>	None	Fixed some security issues.
v1.28.2-r0	<a href="#">v1.28.3</a>	<ul style="list-style-type: none"> <li>• You can configure an ELB blocklist/trustlist for access control when creating a Service or ingress.</li> </ul>	None	Fixed some security issues.
v1.28.1-r4	<a href="#">v1.28.3</a>	None	None	Fixed <a href="#">CVE-2024-21626</a> issues.

CCE Cluster Patch Version	Kubernetes Version	Feature Updates	Optimization	Vulnerability Fixing
v1.28.1-r0	<a href="#">v1.28.3</a>	<p>CCE clusters of v1.28 are released for the first time. For more information, see <a href="#">Kubernetes 1.28 Release Notes</a>.</p> <ul style="list-style-type: none"> <li>• The prefix and suffix of a node name can be customized in node pools.</li> <li>• In CCE Turbo clusters, you can create container networks for workloads and specify pod subnets.</li> <li>• LoadBalancer ingresses support gRPC.</li> <li>• LoadBalancer Services allow you to specify a private IP address for a load balancer during Service creation using YAML.</li> </ul>	<ul style="list-style-type: none"> <li>• Accelerated the startup speed for creating a large number of Kata containers in a CCE Turbo cluster.</li> <li>• Improved the stability when Kata containers are repeatedly created or deleted in a CCE Turbo cluster.</li> </ul>	None

## Version 1.27

### NOTICE

dockershim has been removed since Kubernetes v1.24, and Docker is not supported in v1.24 and later versions by default. Use containerd. To migrate nodes from Docker to containerd, follow the operations described in [Migrating Nodes from Docker to containerd](#).

**Table 2-7** Release notes for the v1.27 patch

CCE Cluster Patch Version	Kubernetes Version	Feature Updates	Optimization	Vulnerability Fixing
v1.27.8-r4	<a href="#">v1.27.12</a>	None	The stability of ELB has been improved during upgrades that span across multiple versions.	Fixed some security issues.
v1.27.8-r0	<a href="#">v1.27.12</a>	<ul style="list-style-type: none"> <li>• CCE ingresses support traffic distribution based on custom HTTP headers.</li> <li>• Scaling priority policies can be configured for third-party workloads.</li> <li>• You can configure a security group for a pod using annotations. This feature is only available for CCE Turbo clusters.</li> <li>• You can bind an existing EIP to a pod. This feature is only available for CCE Turbo clusters.</li> </ul>	<ul style="list-style-type: none"> <li>• An in-progress node drainage can be canceled.</li> <li>• When updating a node pool, you can change its agency name, prefix, and suffix.</li> <li>• Kubernetes labels and taints of a node are retained after the node is reset.</li> <li>• Both the Kubernetes service account token volume projection and the load scaling controller can be configured.</li> </ul>	Fixed some security issues.

CCE Cluster Patch Version	Kubernetes Version	Feature Updates	Optimization	Vulnerability Fixing
v1.27.6-r0	<a href="#">v1.27.9</a>	Docker can be selected when you create a node.	The configurations of frequently used cluster parameters and node pool parameters are publicly available.	Fixed some security issues.
v1.27.3-r4	<a href="#">v1.27.4</a>	None	None	Fixed <a href="#">CVE-2024-21626</a> issues.
v1.27.2-r0	<a href="#">v1.27.2</a>	<ul style="list-style-type: none"> <li>Volcano supports node pool affinity scheduling.</li> <li>Volcano supports workload rescheduling.</li> </ul>	None	Fixed some security issues.
v1.27.1-r10	<a href="#">v1.27.2</a>	None	Optimized the events generated during node pool scaling.	Fixed some security issues.
v1.27.1-r0	<a href="#">v1.27.2</a>	<p>CCE clusters of v1.27 are released for the first time. For more information, see <a href="#">Kubernetes 1.27 Release Notes</a>.</p> <ul style="list-style-type: none"> <li>Both soft eviction and hard eviction are supported in node pool configurations.</li> </ul>	None	None

## Version 1.25

### NOTICE

All nodes in the CCE clusters of version 1.25, except the ones running EulerOS 2.5, use containerd by default.

**Table 2-8** Release notes for the v1.25 patch

CCE Cluster Patch Version	Kubernetes Version	Feature Updates	Optimization	Vulnerability Fixing
v1.25.11-r4	<a href="#">v1.25.16</a>	None	The stability of ELB has been improved during upgrades that span across multiple versions.	Fixed some security issues.
v1.25.11-r0	<a href="#">v1.25.16</a>	<ul style="list-style-type: none"> <li>• CCE ingresses support traffic distribution based on custom HTTP headers.</li> <li>• Scaling priority policies can be configured for third-party workloads.</li> <li>• You can configure a security group for a pod using annotations. This feature is only available for CCE Turbo clusters.</li> <li>• You can bind an existing EIP to a pod. This feature is only available for CCE Turbo clusters.</li> </ul>	<ul style="list-style-type: none"> <li>• An in-progress node drainage can be canceled.</li> <li>• When updating a node pool, you can change its agency name, prefix, and suffix.</li> <li>• Kubernetes labels and taints of a node are retained after the node is reset.</li> <li>• Both the Kubernetes service account token volume projection and the load scaling controller can be configured.</li> </ul>	Fixed some security issues.
v1.25.9-r0	<a href="#">v1.25.16</a>	Docker can be selected when you create a node.	The configurations of frequently used cluster parameters and node pool parameters are publicly available.	Fixed some security issues.
v1.25.6-r4	<a href="#">v1.25.10</a>	None	None	Fixed <a href="#">CVE-2024-21626</a> issues.

CCE Cluster Patch Version	Kubernetes Version	Feature Updates	Optimization	Vulnerability Fixing
v1.25.5-r0	<a href="#">v1.25.5</a>	<ul style="list-style-type: none"> <li>Volcano supports node pool affinity scheduling.</li> <li>Volcano supports workload rescheduling.</li> </ul>	None	Fixed some security issues.
v1.25.4-r10	<a href="#">v1.25.5</a>	None	Optimized the events generated during node pool scaling.	Fixed some security issues.
v1.25.4-r0	<a href="#">v1.25.5</a>	<ul style="list-style-type: none"> <li>Both soft eviction and hard eviction are supported in node pool configurations.</li> <li>TMS tags can be added to automatically created EVS disks to facilitate cost management.</li> </ul>	None	Fixed some security issues.
v1.25.3-r10	<a href="#">v1.25.5</a>	The timeout interval can be configured for a load balancer.	High-frequency parameters of kube-apiserver are configurable.	Fixed some security issues.
v1.25.3-r0	<a href="#">v1.25.5</a>	None	Enhanced network stability of CCE Turbo clusters when their specifications are modified.	Fixed some security issues.
v1.25.1-r0	<a href="#">v1.25.5</a>	CCE clusters of v1.25 are released for the first time. For more information, see <a href="#">Kubernetes 1.25 Release Notes</a> .	None	None



## Version 1.23

**Table 2-9** Release notes for the v1.23 patch

CCE Cluster Patch Version	Kubernetes Version	Feature Updates	Optimization	Vulnerability Fixing
v1.23.16-r4	<a href="#">v1.23.17</a>	None	The stability of ELB has been improved during upgrades that span across multiple versions.	Fixed some security issues.
v1.23.16-r0	<a href="#">v1.23.17</a>	<ul style="list-style-type: none"> <li>CCE ingress support traffic distribution based on custom HTTP headers.</li> <li>Scaling priority policies can be configured for third-party workloads.</li> <li>You can configure a security group for a pod using annotations. This feature is only available for CCE Turbo clusters.</li> <li>You can bind an existing EIP to a pod. This feature is only available for CCE Turbo clusters.</li> </ul>	<ul style="list-style-type: none"> <li>An in-progress node drainage can be canceled.</li> <li>When updating a node pool, you can change its agency name, prefix, and suffix.</li> <li>Kubernetes labels and taints of a node are retained after the node is reset.</li> <li>Both the Kubernetes service account token volume projection and the load scaling controller can be configured.</li> </ul>	Fixed some security issues.
v1.23.14-r0	<a href="#">v1.23.17</a>	None	The configurations of frequently used cluster parameters and node pool parameters are publicly available.	Fixed some security issues.
v1.23.11-r4	<a href="#">v1.23.17</a>	None	None	Fixed <a href="#">CVE-2024-21626</a> issues.

CCE Cluster Patch Version	Kubernetes Version	Feature Updates	Optimization	Vulnerability Fixing
v1.23.10-r0	<a href="#">v1.23.11</a>	<ul style="list-style-type: none"> <li>Volcano supports node pool affinity scheduling.</li> <li>Volcano supports workload rescheduling.</li> </ul>	None	Fixed some security issues.
v1.23.9-r10	<a href="#">v1.23.11</a>	None	Optimized the events generated during node pool scaling.	Fixed some security issues.
v1.23.9-r0	<a href="#">v1.23.11</a>	<ul style="list-style-type: none"> <li>Both soft eviction and hard eviction are supported in node pool configurations.</li> <li>TMS tags can be added to automatically created EVS disks to facilitate cost management.</li> </ul>	None	Fixed some security issues.
v1.23.8-r10	<a href="#">v1.23.11</a>	The timeout interval can be configured for a load balancer.	High-frequency parameters of kube-apiserver are configurable.	Fixed some security issues.
v1.23.8-r0	<a href="#">v1.23.11</a>	None	<ul style="list-style-type: none"> <li>Enhanced Docker reliability during upgrades.</li> <li>Optimized node time synchronization.</li> </ul>	Fixed some security issues.

CCE Cluster Patch Version	Kubernetes Version	Feature Updates	Optimization	Vulnerability Fixing
v1.23.5-r0	<a href="#">v1.23.11</a>	<ul style="list-style-type: none"> <li>• Fault detection and isolation are supported on GPU nodes.</li> <li>• Security groups can be customized by cluster.</li> <li>• CCE Turbo clusters support ENIs pre-binding by node.</li> <li>• containerd is supported.</li> </ul>	<ul style="list-style-type: none"> <li>• Upgraded the etcd version of the master node to the Kubernetes version 3.5.6.</li> <li>• Optimized scheduling so that pods are evenly distributed across AZs after pods are scaled in.</li> <li>• Optimized the memory usage of kube-apiserver when CRDs are frequently updated.</li> </ul>	Fixed some security issues and the following CVE vulnerabilities: <ul style="list-style-type: none"> <li>• <a href="#">CVE-2022-3294</a></li> <li>• <a href="#">CVE-2022-3162</a></li> <li>• <a href="#">CVE-2022-3172</a></li> <li>• <a href="#">CVE-2021-25749</a></li> </ul>
v1.23.1-r0	<a href="#">v1.23.4</a>	CCE clusters of v1.23 are released for the first time. For more information, see <a href="#">Kubernetes 1.23 Release Notes</a> .	None	None

## Version 1.21

**Table 2-10** Release notes for the v1.21 patch

CCE Cluster Patch Version	Kubernetes Version	Feature Updates	Optimization	Vulnerability Fixing
v1.21.15-r0	<a href="#">v1.21.14</a>	None	The configurations of frequently used cluster parameters and node pool parameters are publicly available.	Fixed some security issues.
v1.21.14-r0	<a href="#">v1.21.14</a>	A PVC can be used to dynamically create and mount an SFS Turbo subdirectory.	None	Fixed some security issues.
v1.21.12-r4	<a href="#">v1.21.14</a>	None	None	Fixed <a href="#">CVE-2024-21626</a> issues.
v1.21.11-r20	<a href="#">v1.21.14</a>	<ul style="list-style-type: none"> <li>Volcano supports node pool affinity scheduling.</li> <li>Volcano supports workload rescheduling.</li> </ul>	None	Fixed some security issues.
v1.21.11-r10	<a href="#">v1.21.14</a>	None	Optimized the events generated during node pool scaling.	Fixed some security issues.
v1.21.11-r0	<a href="#">v1.21.14</a>	<ul style="list-style-type: none"> <li>Both soft eviction and hard eviction are supported in node pool configurations.</li> <li>TMS tags can be added to automatically created EVS disks to facilitate cost management.</li> </ul>	None	Fixed some security issues.
v1.21.10-r10	<a href="#">v1.21.14</a>	The timeout interval can be configured for a load balancer.	High-frequency parameters of kube-apiserver are configurable.	Fixed some security issues.

CCE Cluster Patch Version	Kubernetes Version	Feature Updates	Optimization	Vulnerability Fixing
v1.21.10-r0	<a href="#">v1.21.14</a>	None	<ul style="list-style-type: none"> <li>Enhanced Docker reliability during upgrades.</li> <li>Optimized node time synchronization.</li> <li>Enhanced the stability of the Docker runtime for pulling images after nodes are restarted.</li> </ul>	Fixed some security issues.
v1.21.7-r0	<a href="#">v1.21.14</a>	<ul style="list-style-type: none"> <li>Fault detection and isolation are supported on GPU nodes.</li> <li>Security groups can be customized by cluster.</li> <li>CCE Turbo clusters support ENIs pre-binding by node.</li> <li>Control plane logs can be collected.</li> </ul>	Improved the stability of LoadBalancer Services/ingresses with a large number of connections.	Fixed some security issues and the following CVE vulnerabilities: <ul style="list-style-type: none"> <li><a href="#">CVE-2022-3294</a></li> <li><a href="#">CVE-2022-3162</a></li> <li><a href="#">CVE-2022-3172</a></li> </ul>
v1.21.1-r0	<a href="#">v1.21.7</a>	CCE clusters of v1.21 are released for the first time. For more information, see <a href="#">Kubernetes 1.21 Release Notes</a> .	None	None

## Version 1.19

**Table 2-11** Release notes for the v1.19 patch

CCE Cluster Patch Version	Kubernetes Version	Feature Updates	Optimization	Vulnerability Fixing
1.19.16-r84	<a href="#">v1.19.16</a>	None	None	Fixed <a href="#">CVE-2024-21626</a> issues.
v1.19.16-r60	<a href="#">v1.19.16</a>	<ul style="list-style-type: none"> <li>Volcano supports node pool affinity scheduling.</li> <li>Volcano supports workload rescheduling.</li> </ul>	None	Fixed some security issues.
v1.19.16-r50	<a href="#">v1.19.16</a>	None	Optimized the events generated during node pool scaling.	Fixed some security issues.
v1.19.16-r40	<a href="#">v1.19.16</a>	<ul style="list-style-type: none"> <li>Both soft eviction and hard eviction are supported in node pool configurations.</li> <li>TMS tags can be added to automatically created EVS disks to facilitate cost management.</li> </ul>	None	Fixed some security issues.
v1.19.16-r30	<a href="#">v1.19.16</a>	The timeout interval can be configured for a load balancer.	High-frequency parameters of kube-apiserver are configurable.	Fixed some security issues.

CCE Cluster Patch Version	Kubernetes Version	Feature Updates	Optimization	Vulnerability Fixing
v1.19.16-r20	<a href="#">v1.19.16</a>	None	<ul style="list-style-type: none"> <li>• Cloud Native 2.0 Networks allow you to specify subnets for a namespace.</li> <li>• Enhanced the stability of the Docker runtime for pulling images after nodes are restarted.</li> <li>• Optimized the performance of CCE Turbo clusters in allocating ENIs if not all ENIs are pre-bound.</li> </ul>	Fixed some security issues.
v1.19.16-r4	<a href="#">v1.19.16</a>	<ul style="list-style-type: none"> <li>• Containers support SFS 3.0 for storage.</li> <li>• Fault detection and isolation are supported on GPU nodes.</li> <li>• Security groups can be customized by cluster.</li> <li>• CCE Turbo clusters support ENIs pre-binding by node.</li> </ul>	<ul style="list-style-type: none"> <li>• Scheduling is optimized on taint nodes.</li> <li>• Enhanced the long-term running stability of containerd when cores are bound.</li> <li>• Improved the stability of LoadBalancer Services/ingresses with a large number of connections.</li> <li>• Optimized the memory usage of kube-apiserver when CRDs are frequently updated.</li> </ul>	Fixed some security issues and the following CVE vulnerabilities: <ul style="list-style-type: none"> <li>• <a href="#">CVE-2022-3294</a></li> <li>• <a href="#">CVE-2022-3162</a></li> <li>• <a href="#">CVE-2022-3172</a></li> </ul>

CCE Cluster Patch Version	Kubernetes Version	Feature Updates	Optimization	Vulnerability Fixing
v1.19.16-r0	<a href="#">v1.19.16</a>	None	Enhanced the stability in updating LoadBalancer Services when workloads are upgraded and nodes are scaled in or out.	Fixed some security issues and the following CVE vulnerabilities: <ul style="list-style-type: none"> <li>• <a href="#">CVE-2021-25741</a></li> <li>• <a href="#">CVE-2021-25737</a></li> </ul>
v1.19.10-r0	<a href="#">v1.19.10</a>	CCE clusters of v1.19 are released for the first time. For more information, see <a href="#">Kubernetes 1.19 Release Notes</a> .	None	None

## 2.2 Buying a Cluster

### 2.2.1 Comparison Between Cluster Types

#### Comparison

CCE provides different types of clusters for you to select. The following table lists the differences between them.

Category	Subcategory	CCE Standard	CCE Turbo
Positioning	-	Standard clusters that provide highly reliable and secure containers for commercial use	Next-generation clusters designed for Cloud Native 2.0, with accelerated compute, networking, and scheduling



Category	Subcategory	CCE Standard	CCE Turbo
Application scenario	-	For users who expect to use container clusters to manage applications, obtain elastic compute resources, and enable simplified management on compute, network, and storage resources	For users who have higher requirements on performance, resource utilization, and full-scenario coverage
Specification difference	Network model	Cloud native 1.0 networks: for scenarios where requirements on performance are not high and there are not so many containers <ul style="list-style-type: none"> <li>• Tunnel network</li> <li>• Virtual Private Cloud (VPC) network</li> </ul>	Cloud Native 2.0 networks: for scenarios where requirements on performance are high and there are many containers Max networking scale: 2,000 nodes
	Network performance	The container network is overlaid with the VPC network, causing certain performance loss.	The VPC network and container network are flattened into one for zero performance loss.
	Network isolation	<ul style="list-style-type: none"> <li>• Tunnel network model: network policies supported for communications within a cluster</li> <li>• VPC network model: isolation not supported</li> </ul>	Pods can be associated with security groups for isolation. This isolation policy, based on security groups, ensures consistent security isolation both within and outside of a cluster.
	Security isolation	cgroups are used to isolate common containers.	<ul style="list-style-type: none"> <li>• VM-level isolation is supported for secure containers that run only on physical machines.</li> <li>• cgroups are used to isolate common containers.</li> </ul>
	Edge infrastructure management	Not supported	Management of CloudPond edge sites

## 2.2.2 Buying a CCE Standard/Turbo Cluster

On the CCE console, you can easily create Kubernetes clusters. After a cluster is created, the master node is hosted by CCE. You only need to create worker nodes. In this way, you can implement cost-effective O&M and efficient service deployment.

### Precautions

- After a cluster is created, the following items cannot be changed:
  - Cluster type
  - Number of master nodes in the cluster
  - AZ of a master node
  - Network configurations of the cluster, such as the VPC, subnet, Service CIDR block, and kube-proxy ([service forwarding](#)) settings
  - Network model. For example, change **Tunnel network** to **VPC network**.

### Step 1: Log In to the CCE Console

**Step 1** Log in to the [CCE console](#).

**Step 2** On the **Clusters** page, click **Buy Cluster** in the upper right corner.

----End

### Step 2: Configure the Cluster

On the **Buy Cluster** page, configure the parameters.

#### Basic Settings

Parameter	Description
Type	Select <b>CCE Standard Cluster</b> or <b>CCE Turbo Cluster</b> as required. <ul style="list-style-type: none"><li>• CCE standard clusters provide highly reliable and secure containers for commercial use.</li><li>• CCE Turbo clusters use the high-performance cloud native network. Such clusters provide cloud native hybrid scheduling, achieving higher resource utilization and wider scenario coverage.</li></ul> For more details, see cluster types.

Parameter	Description
Billing Mode	<p>Select a billing mode for the cluster as required.</p> <ul style="list-style-type: none"> <li>• <b>Yearly/Monthly:</b> a prepaid billing mode. Resources will be billed based on the service duration. This cost-effective mode is ideal when the duration of resource usage is predictable. If you choose this billing mode, configure the required duration and determine whether to automatically renew the subscription. (If you purchase a monthly subscription, the automatic renewal period is one month. If you purchase a yearly subscription, the automatic renewal period is one year.)</li> <li>• <b>Pay-per-use:</b> a postpaid billing mode. It is suitable in scenarios where resources will be billed based on usage frequency and duration. You can provision or delete resources at any time.</li> </ul>
Cluster Name	<p>Enter a cluster name. Cluster names under the same account must be unique.</p>
Enterprise Project	<p>This parameter is available only for enterprise users who have enabled an enterprise project.</p> <p>After an enterprise project is selected, clusters and their security groups will be created in that project. To manage clusters and other resources like nodes, load balancers, and node security groups, you can use the Enterprise Project Management Service (EPS).</p>
Cluster Version	<p>Select the Kubernetes version used by the cluster.</p>
Cluster Scale	<p>Select a cluster scale for your cluster as required. These values specify the maximum number of nodes that can be managed by the cluster.</p>

Parameter	Description
Master Nodes	<p>Select the number of master nodes. The master nodes are automatically hosted by CCE and deployed with Kubernetes cluster management components such as kube-apiserver, kube-controller-manager, and kube-scheduler.</p> <ul style="list-style-type: none"> <li>• <b>Multiple:</b> Three master nodes will be created for high cluster availability.</li> <li>• <b>Single:</b> Only one master node will be created in your cluster.</li> </ul> <p><b>NOTE</b></p> <ul style="list-style-type: none"> <li>• If more than half of the master nodes in a cluster are faulty, the cluster will not run properly.</li> </ul> <p>You can also select AZs for deploying the master nodes of a specific cluster. By default, AZs are allocated automatically for the master nodes.</p> <ul style="list-style-type: none"> <li>• <b>Automatic:</b> Master nodes are randomly distributed in different AZs for cluster DR. If the number of available AZs is less than the number of nodes to be created, CCE will create the nodes in the AZs with sufficient resources to preferentially ensure cluster creation. In this case, AZ-level DR may not be ensured.</li> <li>• <b>Custom:</b> Master nodes are deployed in specific AZs. If there is one master node in your cluster, you can select one AZ for the master node. If there are multiple master nodes in your cluster, you can select multiple AZs for the master nodes. <ul style="list-style-type: none"> <li>- <b>AZ:</b> Master nodes are deployed in different AZs for cluster DR.</li> <li>- <b>Host:</b> Master nodes are deployed on different hosts in the same AZ for cluster DR.</li> <li>- <b>Custom:</b> Master nodes are deployed in the AZs you specified.</li> </ul> </li> </ul>

### Network Settings

The network settings cover nodes, containers, and Services. For details about the cluster networking and container network models, see [Overview](#).

**Table 2-12** Network settings

Parameter	Description
VPC	Select the VPC to which the cluster belongs. If no VPC is available, click <b>Create VPC</b> to create one. The value cannot be changed after the cluster is created.

Parameter	Description
Node Subnet	Select the subnet to which the master nodes belong. If no subnet is available, click <b>Create Subnet</b> to create one. The value cannot be changed after the cluster is created.
Default Node Security Group	Select the security group automatically generated by CCE or use the existing one as the default security group of the node. <b>NOTICE</b> The default security group must allow traffic from certain ports to ensure normal communication. Otherwise, the node cannot be created.

**Table 2-13** Network settings

Parameter	Description
Network Model	Select <b>VPC network</b> or <b>Tunnel network</b> for your CCE standard cluster. Select <b>Cloud Native Network 2.0</b> for your CCE Turbo cluster. For more information about their differences, see <a href="#">Overview</a> .
Container CIDR Block	Specify the CIDR block for containers, which determines the maximum number of containers allowed in the cluster. This parameter is available only for CCE standard clusters.
Pod Subnet	Select the subnet to which the pod belongs. If no subnet is available, click <b>Create Subnet</b> to create one. This parameter is available only for CCE Turbo clusters. The pod subnet determines the maximum number of containers in a cluster. You can add pod subnets after a cluster is created.

**Table 2-14** Service network

Parameter	Description
Service CIDR Block	Configure the Service CIDR blocks for containers in the same cluster to access each other. The value determines the maximum number of Services you can create. The value cannot be changed after the cluster is created.

Parameter	Description
Request Forwarding	<p>Select <b>IPVS</b> or <b>iptables</b> for your cluster. For details, see <a href="#">Comparing iptables and IPVS</a>.</p> <ul style="list-style-type: none"> <li>• iptables is the traditional kube-proxy mode. This mode applies to the scenario where the number of Services is small or a large number of short connections are concurrently sent on the client. IPv6 clusters do not support iptables.</li> <li>• IPVS allows higher throughput and faster forwarding. This mode applies to scenarios where the cluster scale is large or the number of Services is large.</li> </ul>

### (Optional) Advanced Settings

Parameter	Description
IAM Authentication	CCE clusters support IAM authentication. You can call IAM authenticated APIs to access CCE clusters.
Certificate Authentication	<ul style="list-style-type: none"> <li>• If <b>Automatically generated</b> is selected, the X509-based authentication mode will be enabled by default. X509 is a commonly used certificate format.</li> <li>• If <b>Bring your own</b> is selected, the cluster can identify users based on the header in the request body for authentication. Upload your CA root certificate, client certificate, and private key.</li> </ul> <p><b>CAUTION</b></p> <ul style="list-style-type: none"> <li>• Upload a file <b>smaller than 1 MB</b>. The CA certificate and client certificate can be in <b>.crt</b> or <b>.cer</b> format. The private key of the client certificate can only be uploaded <b>unencrypted</b>.</li> <li>• The validity period of the client certificate must be longer than five years.</li> <li>• The uploaded CA root certificate is used by the authentication proxy and for configuring the kube-apiserver aggregation layer. <b>If any of the uploaded certificates is invalid, the cluster cannot be created.</b></li> <li>• Starting from v1.25, Kubernetes no longer supports certificate authentication generated using the SHA1WithRSA or ECDSAWithSHA1 algorithm. The certificate authentication generated using the SHA256 algorithm is supported instead.</li> </ul>
CPU Management	If enabled, exclusive CPU cores can be allocated to workload pods. For details, see <a href="#">CPU Policy</a> .
Overload Control	After this function is enabled, concurrent requests 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 <a href="#">Enabling Overload Control for a Cluster</a> .

Parameter	Description
Resource Tag	You can add resource tags to classify resources. A maximum of 20 resource tags can be added.  You can create <b>predefined tags</b> on the TMS console. The predefined tags are available to all resources that support tags. You can use these tags to improve the tag creation and resource migration efficiency.
Description	You can enter description for the cluster. A maximum of 200 characters are allowed.

### Step 3: Select Add-ons

Click **Next: Select Add-on**. On the page displayed, select the add-ons to be installed during cluster creation.

#### Basic capabilities

Add-on Name	Description
CCE Container Network (Yangtse CNI)	This is the basic cluster add-on. It provides network connectivity, Internet access, and security isolation for pods in your cluster.
CCE Container Storage (Everest)	This add-on ( <b>CCE Container Storage (Everest)</b> ) is installed by default. It is a cloud native container storage system based on CSI and supports cloud storage services such as EVS.
CoreDNS	This add-on ( <b>CoreDNS</b> ) is installed by default. It provides DNS resolution for your cluster and can be used to access the in-cloud DNS server.
NodeLocal DNSCache	(Optional) If selected, this add-on ( <b>NodeLocal DNSCache</b> ) will be automatically installed. NodeLocal DNSCache improves cluster DNS performance by running a DNS caching agent on cluster nodes.
Volcano Scheduler	(Optional) After you select this option, CCE will automatically install <b>Volcano Scheduler</b> and set the default scheduler of the cluster to Volcano. This will enable you to access advanced scheduling capabilities for batch computing and high-performance computing.

#### Observability

Add-on Name	Description
CCE Node Problem Detector	(Optional) If selected, this add-on ( <a href="#">CCE Node Problem Detector</a> ) will be automatically installed to detect faults and isolate nodes for prompt cluster troubleshooting.

## Step 4: Configure Add-ons

Click **Next: Add-on Configuration**.

### Basic capabilities

Add-on Name	Description
CCE Container Network (Yangtse CNI)	This add-on is unconfigurable.
CCE Container Storage (Everest)	This add-on is unconfigurable. After the cluster is created, choose <b>Add-ons</b> in the navigation pane of the cluster console and modify the configuration.
CoreDNS	This add-on is unconfigurable. After the cluster is created, choose <b>Add-ons</b> in the navigation pane of the cluster console and modify the configuration.
NodeLocal DNSCache	This add-on is unconfigurable. After the cluster is created, choose <b>Add-ons</b> in the navigation pane of the cluster console and modify the configuration.
Volcano Scheduler	This add-on is unconfigurable. After the cluster is created, choose <b>Add-ons</b> in the navigation pane of the cluster console and modify the configuration.

### Observability

Add-on Name	Description
CCE Node Problem Detector	This add-on is unconfigurable. After the cluster is created, choose <b>Add-ons</b> in the navigation pane of the cluster console and modify the configuration.

## Step 5: Confirm the Configuration

After the parameters are specified, click **Next: Confirm configuration**. The cluster resource list is displayed. Confirm the information and click **Submit**.



It takes about 5 to 10 minutes to create a cluster. You can click **Back to Cluster List** to perform other operations on the cluster or click **Go to Cluster Events** to view the cluster details.

## Related Operations

- After creating a cluster, you can use the Kubernetes command line (CLI) tool `kubectl` to connect to the cluster. For details, see [Connecting to a Cluster Using kubectl](#).
- Add nodes to the cluster. For details, see [Creating a Node](#).

### 2.2.3 Comparing iptables and IPVS

kube-proxy is a key component of a Kubernetes cluster. It is used for load balancing and forwarding data between a Service and its backend pods.

CCE supports the iptables and IP Virtual Server (IPVS) forwarding modes.

Feature Difference	iptables	IPVS
Positioning	A mature, stable kube-proxy mode, but its performance is average. It applies to scenarios where the number of Services is small (less than 3000) or there are a large number of short concurrent connections on the client. For details, see <a href="#">iptables</a> .	A high-performance kube-proxy mode. It applies to scenarios where the cluster scale is large or the number of Services is large. For details, see <a href="#">IPVS</a> .
Throughput	Relatively low	Relatively high
Complexity	$O(n)$ . $n$ increases with the number of Services and backend pods in the cluster.	$O(1)$ . In most cases, the connection processing efficiency is irrelevant to the cluster scale.
Load balancing algorithm	iptables has only one algorithm for random selection.	IPVS involves multiple load balancing algorithms, such as round-robin, shortest expected delay, least connections, and various hashing methods.
Cluster IP connectivity	The internal IP address in the cluster cannot be pinged.	The internal IP address in the cluster can be pinged. <b>NOTE</b> The IP address in clusters of v1.27 or later cannot be pinged due to <a href="#">security hardening</a> .

Feature Difference	iptables	IPVS
Additional restrictions	When there are more than 3000 Services in a cluster, network delay may occur.	<ul style="list-style-type: none"> <li>If an ingress and a Service use the same load balancer, the ingress cannot be accessed from the nodes and containers in the cluster because kube-proxy mounts the LoadBalancer Service address to the ipvs-0 bridge. This bridge intercepts the traffic of the load balancer used by the ingress. Use different load balancers for the ingress and Service.</li> </ul>

## iptables

iptables is a Linux kernel function for processing and filtering a large number of data packets. It allows flexible sequences of rules to be attached to various hooks in the packet processing pipeline. When iptables is used, kube-proxy implements NAT and load balancing in the NAT pre-routing hook. For each Service, kube-proxy installs an iptables rule which captures the traffic destined for the Service's ClusterIP and ports and redirects the traffic to one of the backend pods. By default, iptables randomly selects a backend pod. For details, see [iptables proxy mode](#).

## IPVS

IPVS is constructed on top of Netfilter and balances transport-layer loads as part of the Linux kernel. IPVS can direct requests for TCP- or UDP-based services to the real servers, and make services of the real servers appear as virtual services on a single IP address.

In the IPVS mode, kube-proxy uses IPVS load balancing instead of iptables. IPVS is designed to balance loads for a large number of Services. It has a set of optimized APIs and uses optimized search algorithms instead of simply searching for rules from a list. For details, see [IPVS proxy mode](#).

## 2.3 Connecting to a Cluster

### 2.3.1 Connecting to a Cluster Using kubectl

#### Scenario

This section uses a CCE standard cluster as an example to describe how to access a CCE cluster using kubectl.

## Permissions

When you access a cluster using `kubectl`, CCE uses **kubeconfig** generated on the cluster for authentication. This file contains user information, based on which CCE determines which Kubernetes resources can be accessed by `kubectl`. The permissions recorded in a **kubeconfig** file vary from user to user.

For details about user permissions, see [Cluster Permissions \(IAM-based\) and Namespace Permissions \(Kubernetes RBAC-based\)](#).

## Using kubectl

To connect to a Kubernetes cluster from a PC, you can use `kubectl`, a Kubernetes command line tool. You can log in to the CCE console and click the name of the target cluster to access the cluster console. On the **Overview** page, view the access address and `kubectl` connection procedure.

CCE allows you to access a cluster through a private network or a public network.

- Intranet access: The client that accesses the cluster must be in the same VPC as the cluster.
- Public access: The client that accesses the cluster must be able to access public networks and the cluster has been bound with a public network IP.

---

### NOTICE

To bind an EIP to the cluster, go to the **Overview** page and click **Bind** next to **EIP** in the **Connection Information** area. In a cluster with an EIP bound, `kube-apiserver` will be exposed to the Internet and may be attacked. To solve this problem, you can configure Advanced Anti-DDoS for the EIP of the node on which `kube-apiserver` runs.

---

Download `kubectl` and the configuration file. Copy the file to your client, and configure `kubectl`. After the configuration is complete, you can access your Kubernetes clusters. The process is as follows:

### Step 1 Download kubectl.

Prepare a computer that can access the public network and install `kubectl` in CLI mode. You can run the **kubectl version** command to check whether `kubectl` has been installed. If `kubectl` has been installed, skip this step.

This section uses the Linux environment as an example to describe how to install and configure `kubectl`. For details, see [Installing kubectl](#).

1. Log in to your client and download `kubectl`.

```
cd /home
curl -LO https://dl.k8s.io/release/{v1.25.0}/bin/linux/amd64/kubectl
```

`{v1.25.0}` specifies the version. Replace it as required.
2. Install `kubectl`.

```
chmod +x kubectl
mv -f kubectl /usr/local/bin
```

### Step 2 Obtain the kubectl configuration file.

In the **Connection Info** pane on the **Overview** page, click **Configure** next to **kubect**l to check the kubectl connection. On the displayed page, choose **Intranet access** or **Public network access** and download the configuration file.

 **NOTE**

- The kubectl configuration file **kubeconfig** is used for cluster authentication. If the file is leaked, your clusters may be attacked.
- The Kubernetes permissions assigned by the configuration file downloaded by IAM users are the same as those assigned to the IAM users on the CCE console.
- If the KUBECONFIG environment variable is configured in the Linux OS, kubectl preferentially loads the KUBECONFIG environment variable instead of **\$HOME/.kube/config**.

### Step 3 Configure kubectl.

Configure kubectl (A Linux OS is used).

1. Log in to your client and copy the configuration file (for example, **kubeconfig.yaml**) downloaded in **Step 2** to the **/home** directory on your client.
2. Configure the kubectl authentication file.

```
cd /home
mkdir -p $HOME/.kube
mv -f kubeconfig.yaml $HOME/.kube/config
```

3. Switch the kubectl access mode based on service scenarios.

- Run this command to enable intra-VPC access:  
kubectl config use-context internal
- Run this command to enable public access (EIP required):  
kubectl config use-context external
- Run this command to enable public access and two-way authentication (EIP required):  
kubectl config use-context externalTLSVerify

For details about the cluster two-way authentication, see [Two-Way Authentication for Domain Names](#).

----End

## Two-Way Authentication for Domain Names

CCE supports two-way authentication for domain names.

- After an EIP is bound to an API Server, two-way domain name authentication is disabled by default if kubectl is used to access the cluster. You can run **kubect**l config use-context externalTLSVerify to enable the two-way domain name authentication.
- When an EIP is bound to or unbound from a cluster, or a custom domain name is configured or updated, the cluster server certificate will be added the latest cluster access address (including the EIP bound to the cluster and all custom domain names configured for the cluster).
- Asynchronous cluster synchronization takes about 5 to 10 minutes. You can view the synchronization result in **Synchronize Certificate** in **Operation Records**.

- For a cluster that has been bound to an EIP, if the authentication fails (x509: certificate is valid) when two-way authentication is used, bind the EIP again and download **kubeconfig.yaml** again.
- If the two-way domain name authentication is not supported, **kubeconfig.yaml** contains the **"insecure-skip-tls-verify": true** field, as shown in [Figure 2-1](#). To use two-way authentication, download the **kubeconfig.yaml** file again and enable two-way authentication for the domain names.

**Figure 2-1** Two-way authentication disabled for domain names

```
"clusters": [{  
  "name": "mycluster",  
  "cluster": {  
    "server": "https://10.100.0.52:5443",  
    "insecure-skip-tls-verify": true  
  }  
}]
```

## FAQs

- **Error from server Forbidden**

When you use kubectl to create or query Kubernetes resources, the following output is returned:

```
# 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"
```

The cause is that the user does not have the permissions to operate the Kubernetes resources. For details about how to assign permissions, see [Namespace Permissions \(Kubernetes RBAC-based\)](#).

- **The connection to the server localhost:8080 was refused**

When you use kubectl to create or query Kubernetes resources, the following output is returned:

```
The connection to the server localhost:8080 was refused - did you specify the right host or port?
```

The cause is that cluster authentication is not configured for the kubectl client. For details, see [Step 3](#).

## 2.3.2 Accessing a Cluster Using an X.509 Certificate

### Scenario

This section describes how to obtain the cluster certificate from the console and use it to access Kubernetes clusters.

### Procedure

- Step 1** Log in to the CCE console and click the cluster name to access the cluster console.
- Step 2** On the **Overview** page, locate the **Connection Info** area, and click **Download** next to **X.509 certificate**.
- Step 3** In the **Obtain Certificate** dialog box displayed, select the certificate expiration time and download the X.509 certificate of the cluster as prompted.

**NOTICE**

- The downloaded certificate contains three files: **client.key**, **client.crt**, and **ca.crt**. Keep these files secure.
- Certificates are not required for mutual access between containers in a cluster.

**Step 4** Call native Kubernetes APIs using the cluster certificate.

For example, run the **curl** command to call an API to obtain the pod information. In the following information, *192.168.0.18:5443* indicates the private IP address or EIP and port number of the API server in the cluster.

```
curl --cacert ./ca.crt --cert ./client.crt --key ./client.key https://192.168.0.18:5443/api/v1/namespaces/default/pods/
```

For more cluster APIs, see [Kubernetes API](#).

----End

## 2.3.3 Accessing a Cluster Using a Custom Domain Name

### Scenario

Subject Alternative Name (SAN) allows multiple values (including IP addresses, domain names, and so on) to be associated with certificates. A SAN is usually used by the client to verify the server validity in TLS handshakes. Specifically, the validity check includes whether the server certificate is issued by a CA trusted by the client and whether the SAN in the certificate matches the IP address or DNS domain name that the client actually accesses.

If the client cannot directly access the private IP or EIP of the cluster, you can sign the IP address or DNS domain name that can be directly accessed by the client into the cluster server certificate as a SAN to enable two-way authentication on the client, which improves security. Typical use cases include DNAT access and domain name access.


If you have particular proxy access requirements or need to access resources in other regions, you can customize a SAN. Typical domain name access scenarios:

- Add the domain name mapping by either adding the DNS domain name address in the host domain name configuration on the client or configuring **/etc/hosts** on the client host.
- Use domain name access in the intranet. DNS allows you to configure mappings between cluster EIPs and custom domain names. After an EIP is updated, you can continue to use two-way authentication and the domain name to access the cluster without downloading the **kubeconfig.json** file again.
- Add A records on a self-built DNS server.



### Prerequisites

A cluster of v1.19 or later is available.

## Customizing a SAN

- Step 1** Log in to the CCE console.
- Step 2** Click the name of the target cluster in the cluster list to go to the cluster **Overview** page.
- Step 3** In the **Connection Information** area, click  next to **Custom SAN**. In the dialog box displayed, enter the IP address or domain name and click **Save**.

**Figure 2-2** Custom SAN

Connection Info	
Intranet URL	https://192.168.63.137:5443 
EIP	-- <a href="#">Bind</a>
Custom SAN	-- 
kubectl	<a href="#">Configure</a>
Certificate Authentication	X.509 certificate <a href="#">Download</a>

### NOTE

1. This operation will restart kube-apiserver and update the **kubeconfig.json** file for a short period of time. Do not perform operations on the cluster during this period.
2. A maximum of 128 domain names or IP addresses, separated by commas (,), are allowed.
3. If a custom domain name needs to be bound to an EIP, ensure that an EIP has been configured.

----End

## Connecting to a Cluster Using the SAN

### Using kubectl to access the cluster

- Step 1** Download the **kubeconfig.json** file again after the SAN is modified.
  1. Log in to the CCE console and click the cluster name to access the cluster console.
  2. On the **Overview** page, locate the **Connection Info** area, click **Configure** next to **kubectl**. On the page displayed, download the configuration file.

- Step 2** Configure kubectl.

1. Log in to your client and copy the **kubeconfig.json** file downloaded in [Step 1.2](#) to the **/home** directory on your client.
2. Configure the kubectl authentication file.

```
cd /home
mkdir -p $HOME/.kube
mv -f kubeconfig.json $HOME/.kube/config
```

3. Change the kubectl access mode and use the SAN to access the cluster.  
`kubectl config use-context customSAN-0`

In the preceding command, *customSAN-0* indicates the configuration name of the custom SAN. If multiple SANs are configured, the number in the configuration name of each SAN starts from **0** and increases in ascending order, for example, *customSAN-0*, *customSAN-1*, and so on.

----End

### Using an X.509 certificate to access the cluster

**Step 1** After the SAN is modified, download the X509 certificate again.

1. Log in to the CCE console and click the cluster name to access the cluster console.
2. On the **Overview** page, locate the **Connection Info** area, and click **Download** next to **X.509 certificate**.
3. In the **Obtain Certificate** dialog box displayed, select the certificate expiration time and download the X.509 certificate of the cluster as prompted.

**Step 2** Call native Kubernetes APIs using the cluster certificate.

For example, run the **curl** command to call an API to obtain the pod information. In the following information, *example.com:5443* indicates the custom SAN.

```
curl --cacert ./ca.crt --cert ./client.crt --key ./client.key https://example.com:5443/api/v1/namespaces/default/pods/
```

For more cluster APIs, see [Kubernetes API](#).

----End

## 2.3.4 Revoking a Credential to Access the Corresponding Cluster

In multi-tenant scenarios, CCE generates a credential (kubeconfig or X509 certificate) for you to access the corresponding cluster. The credential contains user identity and authorization information so that you can access to the cluster and perform authorized operations on it. Credentials ensure security between IAM users for user management and authorization. However, the validity period of a credential is usually fixed. When an employee holding the credential resigns or an authorized credential is disclosed, you need to manually revoke the credential to ensure cluster security.

The credentials can be revoked in the following scenarios:

- IAM users (non-administrators) intend to revoke their own credentials.
- Accounts or administrators (users in the admin user group) can revoke the credentials of other users or delegated accounts.



**NOTICE**

After a credential is revoked, the holder of the credential cannot access the cluster, and the revoked credential cannot be restored. Exercise caution when performing this operation.

To access the same cluster, you need to regenerate a credential.

## Prerequisites

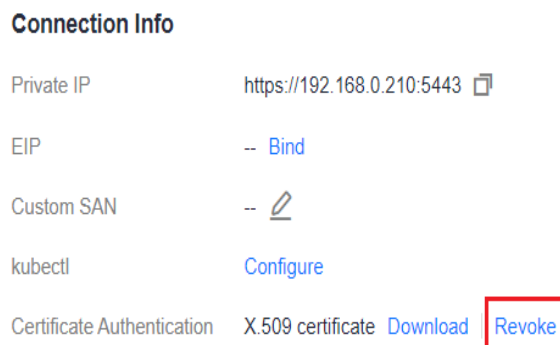
A cluster of v1.19.16-r50, v1.21.11-r10, v1.23.9-r10, v1.25.4-r10, v1.27.1-r10, or later is available.

## Operations Performed by IAM Users

IAM users can only revoke their own credentials. To revoke a credential, perform the following operations:

- Step 1** Log in to the CCE console and click the cluster name to access the cluster console.
- Step 2** Choose **Overview** in the navigation pane. In the **Connection Info** area on the right of the page, click **Revoke**.

**Figure 2-3** Revoking a credential



- Step 3** In the dialog box displayed, enter **REVOKE** in the confirmation box and click **OK** if you are sure you want to revoke the credential.

----End

## Operations Performed by Accounts or Administrators

Accounts or administrators (users in the admin user group) can revoke the credentials of other users or delegated accounts. To revoke a credential, see the following operations:

- Step 1** Log in to the CCE console and click the cluster name to access the cluster console.
- Step 2** Choose **Overview** in the navigation pane. In the **Connection Info** area on the right of the page, click **Revoke**.

**Figure 2-4** Revoking a credential

Connection Info	
Private IP	https://192.168.0.210:5443
EIP	-- <a href="#">Bind</a>
Custom SAN	--
kubectl	<a href="#">Configure</a>
Certificate Authentication	X.509 certificate <a href="#">Download</a> <a href="#">Revoke</a>

**Step 3** Select a certificate applicant and revoke the certificate. The revoked certificate cannot be restored. Exercise caution when performing this operation.

- **User:** Select an IAM user and revoke its credential.
- **Account:** Select an agency account and revoke its credentials.
- **Specify User ID/Delegacy ID:** If the user has been deleted or you log in to the system through an identity provider, manually enter the user ID. If the delegated account has been deleted, manually enter the delegated ID.

You can use the following solution to obtain the ID:

- If you can obtain the certificate downloaded by the applicant, the name (**CN - Common Name**) of the certificate is the required ID.
- If you cannot obtain the certificate downloaded by the applicant, use Cloud Trace Service (CTS) to obtain the events of deleting a user (**deleteUser**) and deleting an agency (**deleteAgency**). The resource IDs of the events are the IDs of the deleted user and delegated account, respectively.

If the required ID cannot be obtained using the preceding methods, submit a service ticket to O&M personnel.

**Step 4** In the dialog box displayed, enter **REVOKE** in the confirmation box and click **OK** if you are sure you want to revoke the credential.

----End

## 2.4 Managing a Cluster

### 2.4.1 Modifying Cluster Configurations

#### Scenario

CCE allows you to manage cluster parameters, through which you can let core components work under your requirements.

#### Procedure

**Step 1** Log in to the CCE console. In the navigation pane, choose **Clusters**.

**Step 2** Locate the target cluster, click ... to view more operations on the cluster, and choose **Manage**.

**Step 3** On the **Manage Component** page, change the values of the Kubernetes parameters listed in the following table.

**Table 2-15** kube-apiserver configurations

Item	Parameter	Description	Value
Toleration time for nodes in NotReady state	default-not-ready-toleration-seconds	Specifies the default tolerance time. The configuration takes effect for all pods by default. You can configure different tolerance time for pods. In this case, the tolerance time configured for the pod is used. For details, see <a href="#">Configuring Tolerance Policies</a> .  If the specified tolerance time is too short, pods may be frequently migrated in scenarios like a network jitter. If the specified tolerance time is too long, services may be interrupted during this period after the node is faulty.	Default: 300s
Toleration time for nodes in unreachable state	default-unreachable-toleration-seconds	Specifies the default tolerance time. The configuration takes effect for all pods by default. You can configure different tolerance time for pods. In this case, the tolerance time configured for the pod is used. For details, see <a href="#">Configuring Tolerance Policies</a> .  If the specified tolerance time is too short, pods may be frequently migrated in scenarios like a network jitter. If the specified tolerance time is too long, services may be interrupted during this period after the node is faulty.	Default: 300s

Item	Parameter	Description	Value
Maximum Number of Concurrent Modification API Calls	max-mutating-requests-inflight	<p>Maximum number of concurrent mutating requests. When the value of this parameter is exceeded, the server rejects requests.</p> <p>The value <b>0</b> indicates that there is no limitation on the maximum number of concurrent modification requests. This parameter is related to the cluster scale. You are advised not to change the value.</p>	<p>Manual configuration is no longer supported since cluster v1.21. The value is automatically specified based on the cluster scale.</p> <ul style="list-style-type: none"> <li>• <b>200</b> for clusters with 50 or 200 nodes</li> <li>• <b>500</b> for clusters with 1000 nodes</li> <li>• <b>1000</b> for clusters with 2000 nodes</li> </ul>
Maximum Number of Concurrent Non-Modification API Calls	max-requests-inflight	<p>Maximum number of concurrent non-mutating requests. When the value of this parameter is exceeded, the server rejects requests.</p> <p>The value <b>0</b> indicates that there is no limitation on the maximum number of concurrent non-modification requests. This parameter is related to the cluster scale. You are advised not to change the value.</p>	<p>Manual configuration is no longer supported since cluster v1.21. The value is automatically specified based on the cluster scale.</p> <ul style="list-style-type: none"> <li>• <b>400</b> for clusters with 50 or 200 nodes</li> <li>• <b>1000</b> for clusters with 1000 nodes</li> <li>• <b>2000</b> for clusters with 2000 nodes</li> </ul>

Item	Parameter	Description	Value
NodePort port range	service-node-port-range	<p>NodePort port range. After changing the value, go to the security group page and change the TCP/UDP port range of node security groups 30000 to 32767. Otherwise, ports other than the default port cannot be accessed externally.</p> <p>If the port number is smaller than <b>20106</b>, a conflict may occur between the port and the CCE health check port, which may further lead to unavailable cluster. If the port number is greater than <b>32767</b>, a conflict may occur between the port and the ports in <b>net.ipv4.ip_local_port_range</b>, which may further affect the network performance.</p>	<p>Default: 30000 to 32767</p> <p>Value range: Min &gt; 20105 Max &lt; 32768</p>
Overload Control	support-overload	<p>Cluster overload control. After this function is enabled, concurrent requests 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.</p> <p>This parameter is available only in clusters of v1.23 or later.</p>	<ul style="list-style-type: none"> <li>• false: Overload control is disabled.</li> <li>• true: Overload control is enabled.</li> </ul>
Node Restriction Add-on	enable-admission-plugin-node-restriction	<p>This add-on allows the kubelet of a node to operate only the objects of the current node for enhanced isolation in multi-tenant scenarios or the scenarios with high security requirements.</p> <p>This parameter is available only in clusters of v1.23.14-r0, v1.25.9-r0, v1.27.6-r0, v1.28.4-r0, or later versions.</p>	Default: true

Item	Parameter	Description	Value
Pod Node Selector Add-on	enable-admission-plugin-pod-node-selector	This add-on allows cluster administrators to configure the default node selector through namespace annotations. In this way, pods run only on specific nodes and configurations are simplified.  This parameter is available only in clusters of v1.23.14-r0, v1.25.9-r0, v1.27.6-r0, v1.28.4-r0, or later versions.	Default: true
Pod Toleration Limit Add-on	enable-admission-plugin-pod-toleration-restriction	This add-on allows cluster administrators to configure the default value and limits of pod tolerations through namespaces for fine-grained control over pod scheduling and key resource protection.  This parameter is available only in clusters of v1.23.14-r0, v1.25.9-r0, v1.27.6-r0, v1.28.4-r0, or later versions.	Default: false

**Table 2-16** Scheduler configurations

Item	Parameter	Description	Value
Default cluster scheduler	default-scheduler	<ul style="list-style-type: none"> <li>• kube-scheduler scheduler: provides the standard scheduling capability of the community.</li> <li>• volcano scheduler: compatible with kube-scheduler scheduling capabilities and provides enhanced scheduling capabilities. For details, see <a href="#">Volcano Scheduling</a>.</li> </ul>	Default: kube-scheduler

Item	Parameter	Description	Value
QPS for communicating with kube-apiserver	kube-api-qps	QPS for communicating with kube-apiserver.	<ul style="list-style-type: none"> <li>• If the number of nodes in a cluster is less than 1000, the default value is <b>100</b>.</li> <li>• If the number of nodes in a cluster is 1000 or more, the default value is <b>200</b>.</li> </ul>
Burst for communicating with kube-apiserver	kube-api-burst	Burst for communicating with kube-apiserver.	<ul style="list-style-type: none"> <li>• If the number of nodes in a cluster is less than 1000, the default value is <b>100</b>.</li> <li>• If the number of nodes in a cluster is 1000 or more, the default value is <b>200</b>.</li> </ul>

Item	Parameter	Description	Value
Whether to enable GPU sharing	enable-gpu-share	<p>Whether to enable GPU sharing. This parameter is supported only in clusters of v1.23.7-r10, v1.25.3-r0, or later versions.</p> <ul style="list-style-type: none"> <li>When disabled, ensure that pods in the cluster cannot use shared GPUs (no <b>cce.io/gpu-decision</b> annotation in pods) and that GPU virtualization is disabled.</li> <li>When enabled, ensure that there is a <b>cce.io/gpu-decision</b> annotation on all pods that use GPU resources in the cluster.</li> </ul>	Default: true

**Table 2-17** kube-controller-manager configurations

Item	Parameter	Description	Value
Number of concurrent processing of deployment	concurrent-deployment-syncs	Number of deployment objects that can be synchronized concurrently	Default: 5
Concurrent processing number of endpoint	concurrent-endpoint-syncs	Number of endpoint syncing operations that will be done concurrently	Default: 5
Concurrent number of garbage collector	concurrent-gc-syncs	Number of garbage collector workers that can be synchronized concurrently	Default: 20
Number of job objects allowed to sync simultaneously	concurrent-job-syncs	Number of job objects that can be synchronized concurrently	Default: 5



Item	Parameter	Description	Value
Number of CronJob objects allowed to sync simultaneously	concurrent-cron-job-syncs	Number of scheduled jobs that can be synchronized concurrently	Default: 5
Number of concurrent processing of namespace	concurrent-namespace-syncs	Number of namespace objects that can be synchronized concurrently	Default: 10
Concurrent processing number of replicaset	concurrent-replicaset-syncs	Number of replica sets that can be synchronized concurrently	Default: 5
Number of concurrent processing of resource quota	concurrent-resource-quota-syncs	Number of resource quotas that can be synchronized concurrently	Default: 5
Concurrent processing number of service	concurrent-service-syncs	Number of services that can be synchronized concurrently	Default: 10
Concurrent processing number of serviceaccount-token	concurrent-serviceaccount-token-syncs	Number of service account token objects that can be synchronized concurrently	Default: 5
Concurrent processing of ttl-after-finished	concurrent-ttl-after-finished-syncs	Number of <b>ttl-after-finished-controller</b> workers that can be synchronized concurrently	Default: 5
RC	concurrent_rc_syncs (used in clusters of v1.19 or earlier) concurrent-rc-syncs (used in clusters of v1.21 through v1.25.3-r0)	Number of replication controllers that can be synchronized concurrently <b>NOTE</b> This parameter is no longer supported in clusters of v1.25.3-r0 and later versions.	Default: 5

Item	Parameter	Description	Value
Cluster elastic computing period	horizontal-pod-autoscaler-sync-period	<p>Period for the horizontal pod autoscaler to perform auto scaling on pods. A smaller value will result in a faster auto scaling response and higher CPU load.</p> <p><b>NOTE</b> Make sure to configure this parameter properly as a lengthy period can cause the controller to respond slowly, while a short period may overload the cluster control plane.</p>	Default: 15 seconds
Horizontal Pod Scaling Tolerance	horizontal-pod-autoscaler-tolerance	<p>The configuration determines how quickly the horizontal pod autoscaler will act to auto scaling policies. If the parameter is set to <b>0</b>, auto scaling will be triggered immediately when the related metrics are met.</p> <p>Configuration suggestion: If the service resource usage increases sharply over time, retain a certain tolerance to prevent auto scaling which is beyond expectation in high resource usage scenarios.</p>	Default: 0.1
QPS for communicating with kube-apiserver	kube-api-qps	QPS for communicating with kube-apiserver	<ul style="list-style-type: none"> <li>• If the number of nodes in a cluster is less than 1000, the default value is <b>100</b>.</li> <li>• If the number of nodes in a cluster is 1000 or more, the default value is <b>200</b>.</li> </ul>

Item	Parameter	Description	Value
Burst for communicating with kube-apiserver	kube-api-burst	Burst for communicating with kube-apiserver	<ul style="list-style-type: none"> <li>If the number of nodes in a cluster is less than 1000, the default value is <b>100</b>.</li> <li>If the number of nodes in a cluster is 1000 or more, the default value is <b>200</b>.</li> </ul>
The maximum number of terminated pods that can be kept before the Pod GC deletes the terminated pod	terminated-pod-gc-threshold	<p>Number of terminated pods that can exist in a cluster. If there are more terminated pods than the expected number in the cluster, the terminated pods that exceed the number will be deleted.</p> <p><b>NOTE</b> If this parameter is set to <b>0</b>, all pods in the terminated state are retained.</p>	<p>Default: 1000</p> <p>Value range: 10 to 12500</p> <p>If the cluster version is v1.21.11-r40, v1.23.8-r0, v1.25.6-r0, v1.27.3-r0, or later, the value range is changed to 0 to 100000.</p>

Item	Parameter	Description	Value
Unhealthy AZ Threshold	unhealthy-zone-threshold	<p>When more than a certain proportion of pods in an AZ are unhealthy, the AZ itself will be considered unhealthy, and scheduling pods to nodes in that AZ will be restricted to limit the impacts of the unhealthy AZ.</p> <p>This parameter is available only in clusters of v1.23.14-r0, v1.25.9-r0, v1.27.6-r0, v1.28.4-r0, or later versions.</p> <p><b>NOTE</b> If the parameter is set to a large value, pods in unhealthy AZs will be migrated in a large scale, which may lead to risks such as overloaded clusters.</p>	<p>Default: 0.55</p> <p>Value range: 0 to 1</p>
Node Eviction Rate	node-eviction-rate	<p>This parameter specifies the number of nodes that pods are deleted from per second in a cluster when the AZ is healthy. The default value is <b>0.1</b>, indicating that pods can be evicted from at most one node every 10 seconds.</p> <p>This parameter is available only in clusters of v1.23.14-r0, v1.25.9-r0, v1.27.6-r0, v1.28.4-r0, or later versions.</p> <p><b>NOTE</b> If the parameter is set to a large value, the cluster may be overloaded. Additionally, if too many pods are evicted, they cannot be rescheduled, which will slow down fault recovery.</p>	<p>Default: 0.1</p>

Item	Parameter	Description	Value
Secondary Node Eviction Rate	secondary-node- eviction-rate	<p>This parameter specifies the number of nodes that pods are deleted from per second in a cluster when the AZ is unhealthy. The default value is <b>0.01</b>, indicating that pods can be evicted from at most one node every 100 seconds.</p> <p>This parameter is available only in clusters of v1.23.14-r0, v1.25.9-r0, v1.27.6-r0, v1.28.4-r0, or later versions.</p> <p><b>NOTE</b> There is no need to set the parameter to a large value for nodes in an unhealthy AZ, and this configuration may result in overloaded clusters.</p>	<p>Default: 0.01</p> <p>Configure this parameter with <b>node- eviction-rate</b> and set it to one-tenth of <b>node- eviction-rate</b>.</p>

Item	Parameter	Description	Value
Large Cluster Threshold	large-cluster-size-threshold	<p>If the number of nodes in a cluster is greater than the value of this parameter, this is a large cluster.</p> <p>This parameter is available only in clusters of v1.23.14-r0, v1.25.9-r0, v1.27.6-r0, v1.28.4-r0, or later versions.</p> <p><b>NOTE</b>                      kube-controller-manager automatically adjusts configurations for large clusters to optimize the cluster performance. Therefore, an excessively small threshold for small clusters will deteriorate the cluster performance.</p>	<p>Default: 50</p> <p>For the clusters with a large number of nodes, configure a relatively larger value than the default one for higher performance and faster responses of controllers. Retain the default value for small clusters. Before adjusting the value of this parameter in a production environment, check the impact of the change on cluster performance in a test environment.</p>

**Table 2-18** Networking component configurations (supported only by the clusters using a VPC network)

Item	Parameter	Description	Value
Retaining the non-masqueraded CIDR block of the original pod IP address	nonMasqueradeCIDRs	<p>In a CCE cluster using the VPC network model, if a container in the cluster needs to access externally, the source pod IP address must be masqueraded as the IP address of the node where the pod resides through SNAT. After the configuration, the node will not SNAT the IP addresses in the CIDR block by default. This function is available only in clusters of v1.23.14-r0, v1.25.9-r0, v1.27.6-r0, v1.28.4-r0, or later versions.</p> <p>By default, nodes in a cluster do not perform SNAT on packets destined for 10.0.0.0/8, 172.16.0.0/12, or 192.168.0.0/16 that is detected by CCE as a private CIDR block. Instead, these packets are directly transferred using the upper-layer VPC. (The three CIDR blocks are considered as internal networks in the cluster and are reachable at Layer 3 by default.)</p>	<p>Default: 10.0.0.0/8, 172.16.0.0/12, or 192.168.0.0/16</p> <p><b>NOTE</b> To enable cross-node pod access, the CIDR block of the node where the target pod runs must be added.</p> <p>Similarly, to enable cross-ECS pod access in a VPC, the CIDR block of the ECS where the target pod runs must be added.</p>

**Table 2-19** Extended controller configurations (supported only by clusters of v1.21 and later)

Item	Parameter	Description	Value
Enable resource quota management	enable-resource-quota	<p>Indicates whether to automatically create a ResourceQuota when creating a namespace. With quota management, you can control the number of workloads of each type and the upper limits of resources in a namespace or related dimensions.</p> <ul style="list-style-type: none"> <li>● <b>false</b>: Auto creation is disabled.</li> <li>● <b>true</b>: Auto creation is enabled. For details about the resource quota defaults, see <a href="#">Configuring Resource Quotas</a>.</li> </ul> <p><b>NOTE</b> In high-concurrency scenarios (for example, creating pods in batches), the resource quota management may cause some requests to fail due to conflicts. Do not enable this function unless necessary. To enable this function, ensure that there is a retry mechanism in the request client.</p>	Default: false

**Step 4** Click **OK**.

----End

## References

- [kube-apiserver](#)
- [kube-controller-manager](#)
- [kube-scheduler](#)

## 2.4.2 Enabling Overload Control for a Cluster

### Scenario

After overload control is enabled, the number of simultaneous requests is dynamically regulated according to the resource pressure on the master nodes. This ensures that both the nodes and the cluster remain accessible.



## Notes and Constraints

The cluster version must be 1.23 or later.

## Enabling Overload Control

### Method 1: Enabling it when creating a cluster

When creating a cluster of v1.23 or later, you can enable overload control during the cluster creation.

### Method 2: Enabling it in an existing cluster

**Step 1** Log in to the CCE console and click the name of an existing cluster whose version is v1.23 or later.

**Step 2** On the **Overview** page, check the master node information. If overload control is not enabled, a message will be displayed. You can click **Enable** to enable the function.

----End

## Disabling Cluster Overload Control

**Step 1** Log in to the CCE console and click the name of an existing cluster whose version is v1.23 or later.

**Step 2** In the navigation pane, choose **Settings**.

**Step 3** On the **Cluster Access** tab page, disable overload control.

**Step 4** Click **OK**.

----End

## 2.4.3 Changing Cluster Scale

### Scenario

CCE allows you to change the number of nodes managed in a cluster.

### Notes and Constraints

- A cluster that has only one master node supports fewer than 1000 worker nodes.
- The number of master nodes cannot be changed when you modify cluster specifications.
- A cluster can only be scaled out to a larger specification, but cannot be scaled in.
- To ensure proper cluster functionality, perform any specifications changes during off-peak hours. This is because master nodes will need to be powered off and on, which can disrupt normal operations.

## Procedure

- Step 1** Log in to the CCE console. In the navigation pane, choose **Clusters**.
- Step 2** Locate the cluster whose specifications need to be modified, click ... to view more operations on the cluster, and choose **Modify Specifications**.
- Step 3** On the page displayed, select a new cluster scale.
- Step 4** Click **Next** to confirm the specifications and click **OK**.

You can click **Operation Records** in the upper right corner to view the cluster change history. The status changes from **Executing** to **Successful**, indicating that the cluster specifications are successfully changed.

### NOTE

After the cluster scale is changed to 1000 nodes or more, some parameter values of the cluster will be automatically adjusted to ensure the cluster performance. For details, see [Modifying Cluster Configurations](#).

----End

## 2.4.4 Changing the Default Security Group of a Node

### Scenario

When creating a cluster, you can customize a node security group to centrally manage network security policies. For a created cluster, you can change its default node security group.

### Notes and Constraints

- Do not add more than 1000 pods to the same security group. Otherwise, the security group performance may be impacted.
- Exercise caution when modifying the security group rules of a master node.

### Procedure

- Step 1** Log in to the CCE console. In the navigation pane, choose **Clusters**.
- Step 2** Click the cluster name to access the **Overview** page.
- Step 3** In the **Networking Configuration** area, click **Edit** next to the **Default Node Security Group**.

**Figure 2-5** Default node security group

Networking Configuration		Go to Network Configuration
Network Model	Tunnel network	
VPC	<a href="#">vpc-9cf3</a>	
Subnet	<a href="#">subnet-40b4</a> (Available/All IPs: 242/251)	
Container CIDR Block	10.0.0.0/16	
IPv4 Service CIDR Block	10.247.0.0/16	
Forwarding	iptables	
Default Node Security Group	<a href="#">[Security Group ID]</a> <a href="#">Edit</a>	

**Step 4** Select an existing security group, confirm that the security group rules meet the cluster requirements, and click **OK**.

**NOTICE**

- Ensure that correct port rules are configured for the selected security group. Otherwise, the node cannot be created. The port rules that a security group must comply with vary with the cluster type.
- The new security group takes effect only for newly created or managed nodes. For existing nodes, modify the security group rules and reset the nodes in real time. The original security group is still used.

**Figure 2-6** Editing default node security group

**Edit Default Node Security Group** ✕

Default Node Security Group:  Default ▾

[Create Security Group](#)

Configure security group rules for worker nodes in your cluster. The rules will take effect on all worker nodes in the cluster. For additional security group configurations, go to the Security Groups page. The security group for master nodes is automatically created by CCE. The modified security group applies only to nodes newly created or accepted. For existing nodes, you need to manually modify the security group rules for them. You can [add a security group](#) to customize security group rules. The modified security group applies only to nodes newly created or accepted. For existing nodes, you need to manually modify the security group rules for them. [How to Modify?](#)

I have confirmed that the security group rules have been correctly configured for nodes to communicate with each other.

OK
Cancel

----End

## 2.4.5 Deleting a Cluster

### Scenario

- You can directly delete pay-per-use clusters. For details, see [Deleting a Pay-per-Use Cluster](#).
- Yearly/Monthly clusters cannot be deleted directly. Unsubscribe from clusters that have not expired or release clusters that have expired and have not been renewed. For details, see [Unsubscribing from or Releasing a Yearly/Monthly Cluster](#).

### Precautions

- Deleting a cluster will delete the workloads and Services in the cluster, and the deleted data cannot be recovered. Before performing this operation, ensure that related data has been backed up or migrated.
- If you choose to delete a cluster with the nodes in it, the system disks and data disks attached to the nodes will also be deleted. Back up data before the deletion.
- If you delete a cluster that is not running (for example, frozen or unavailable), associated resources, such as storage and networking resources, will remain.

### Deleting a Pay-per-Use Cluster

---

**NOTICE**

A hibernated cluster cannot be deleted. Wake up the cluster and try again.

---

- Step 1** Log in to the CCE console. In the navigation pane, choose **Clusters**.
- Step 2** Locate the cluster to be deleted, click ... to view more operations on the cluster, and choose **Delete**.
- Step 3** In the displayed **Delete Cluster** dialog box, select the resources to be released.
- When you delete nodes from a cluster, the following options are available:
    - **Retain:** The node as well as the data on the system disk and data disks will be retained.
    - **Delete:** The node will be deleted with the cluster. This option is available only for pay-per-use nodes. To delete a yearly/monthly node, manually unsubscribe from the node.
    - **Reset:** The node will be retained and reset. The data on the system disk and data disks will not be retained.
  - Delete cloud storage resources associated with workloads in the cluster.

 **NOTE**

When deleting underlying cloud storage resources bound to storage volumes in a cluster, pay attention to following constraints:

- The underlying storage resources are deleted according to the reclamation policy you defined for the storage volumes. For example, if the reclamation policy of storage volumes is **Retain**, the underlying storage resources will be retained after the cluster is deleted.
- If there are more than 1000 files in the OBS bucket, manually clear the files and then delete the cluster.

If the storage volume's reclamation policy is set to **Retain**, the following table lists the method of deleting PVs based on their type.

PV Type	Associated Underlying Storage	Whether to Delete Underlying Storage
EVS	Disks	Yes
DSS	Disks	Yes
SFS	SFS Capacity-Oriented or SFS 3.0	Yes
OBS	OBS buckets or parallel file systems	Yes
SFS Turbo	SFS Turbo file systems	Yes
Local PV	Logical volumes mounted to the node	Determined based on the policy selected when the cluster is deleted. If you choose to retain the node, the logical volumes will not be deleted. If you choose to delete or reset the node, the logical volumes will be deleted.
SFS 3.0 subdirectory	A directory in SFS 3.0	No
SFS Turbo subdirectory	A directory in SFS Turbo	No

- Delete network resources such as load balancers in a cluster. (Only automatically created load balancers will be deleted).

**Step 4** Enter **DELETE** and click **Yes** to start deleting the cluster.

The delete operation takes 1 to 3 minutes to complete.

----**End**

## Unsubscribing from or Releasing a Yearly/Monthly Cluster

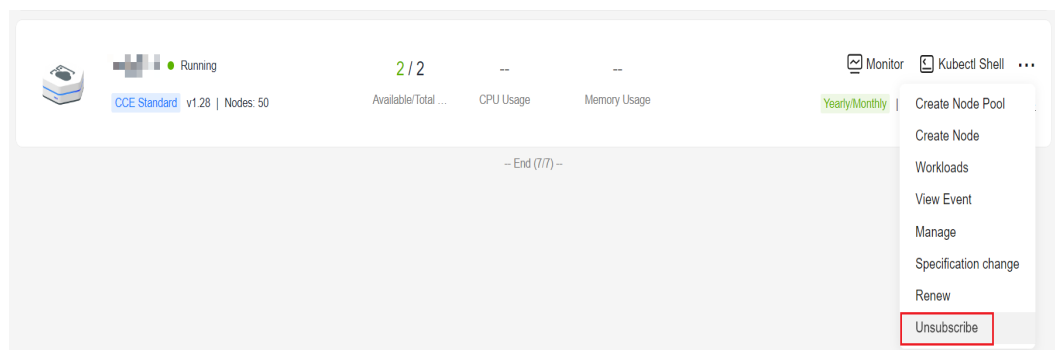
### NOTICE

- When you unsubscribe from or release a cluster, only the resources associated with the order are unsubscribed from. Non-associated resources are retained and their billing keeps going.
- For a yearly/monthly cluster, if the retention expires, the cluster will be automatically released. For the nodes in the cluster, if they expire at the same time, they will also be released. If not, CCE will not perform any operation on your nodes. Node data is retained and the billing keeps going. Pay attention to the expired clusters under your account and renew them in a timely manner to prevent data loss caused by node reinstallation.
- If an order contains resources in a primary-secondary relationship, unsubscribe from the resources separately.
- For details about unsubscription rules, see [Unsubscription Rules](#).
  - If you are unsubscribing from a resource that is being used, review the resource information and refund information carefully. The resource cannot be restored after unsubscription. If you want to reserve your resources and unsubscribe from only the unused renewal periods, log in to the Enterprise Projects console and [unsubscribe from renewal period as prompted](#).
  - Unsubscribing from a resource associated with other yearly/monthly-billed resources may affect the normal use of those resources.  
Unsubscribing from a resource associated with other pay-per-use resources will not affect the normal use of those resources. They will be billed normally.
  - If your operation is not a five-day unconditional unsubscription, you will be charged for the handling fee and the used amount. Used cash coupons and discount coupons will not be refunded.

**Step 1** Log in to the CCE console. In the navigation pane, choose **Clusters**.

**Step 2** Locate the cluster to be unsubscribed from, click ... to view more operations on the cluster, and choose **Unsubscribe** or **Release**.

**Figure 2-7** Unsubscribing from a cluster



**Step 3** On the displayed page, select the resources to be released.

- When you delete nodes from a cluster, the following options are available:
  - **Retain:** The node as well as the data on the system disk and data disks will be retained.
  - **Delete:** The node will be deleted with the cluster. This option is available only for pay-per-use nodes. To delete a yearly/monthly node, manually unsubscribe from the node.
  - **Reset:** The node will be retained and reset. The data on the system disk and data disks will not be retained.
- Delete cloud storage resources associated with workloads in the cluster.

 **NOTE**

When deleting underlying cloud storage resources bound to storage volumes in a cluster, pay attention to following constraints:

- The underlying storage resources are deleted according to the reclamation policy you defined for the storage volumes. For example, if the reclamation policy of storage volumes is **Retain**, the underlying storage resources will be retained after the cluster is deleted.
- If there are more than 1000 files in the OBS bucket, manually clear the files and then delete the cluster.

If the storage volume's reclamation policy is set to **Retain**, the following table lists the method of deleting PVs based on their type.

PV Type	Associated Underlying Storage	Whether to Delete Underlying Storage
EVS	Disks	Yes
DSS	Disks	Yes
SFS	SFS Capacity-Oriented or SFS 3.0	Yes
OBS	OBS buckets or parallel file systems	Yes
SFS Turbo	SFS Turbo file systems	Yes
Local PV	Logical volumes mounted to the node	Determined based on the policy selected when the cluster is deleted. If you choose to retain the node, the logical volumes will not be deleted. If you choose to delete or reset the node, the logical volumes will be deleted.
SFS 3.0 subdirectory	A directory in SFS 3.0	No
SFS Turbo subdirectory	A directory in SFS Turbo	No

- Delete network resources such as load balancers in a cluster. (Only automatically created load balancers will be deleted).

**Step 4** Click **Yes**. The unsubscription or release takes 1 to 3 minutes to complete.

----End

## 2.4.6 Hibernating or Waking Up a Cluster

### Scenario

If a cluster is not needed temporarily, hibernate it to reduce costs.

After a cluster is hibernated, resources such as workloads cannot be created or managed in the cluster.

### Precautions

- During cluster wakeup, the master node may fail to start due to insufficient resources, which leads to a cluster wakeup failure. In this case, wait for a while and try again.
- After a cluster is woken up, it takes 3 to 5 minutes to initialize data. Deliver services after the cluster runs properly.

### Hibernating a Cluster

**Step 1** Log in to the CCE console. In the navigation pane, choose **Clusters**.

**Step 2** Locate the cluster to be hibernated, click ... to view more operations on the cluster, and choose **Hibernate**.

**Step 3** In the dialog box displayed, check the precautions and click **Yes**. Wait until the cluster is hibernated.

----End

### Waking Up a Cluster

**Step 1** Log in to the CCE console. In the navigation pane, choose **Clusters**.

**Step 2** Click **Wake Up** in the row of the target cluster.

**Step 3** When the cluster status changes from **Waking up** to **Running**, the cluster is woken up. It takes about 3 to 5 minutes to wake up the cluster.

----End

## 2.4.7 Renewing a Yearly/Monthly Cluster

You can renew a yearly/monthly-billed cluster.

### Procedure

This section describes how to renew a cluster billed on a **yearly/monthly** basis.



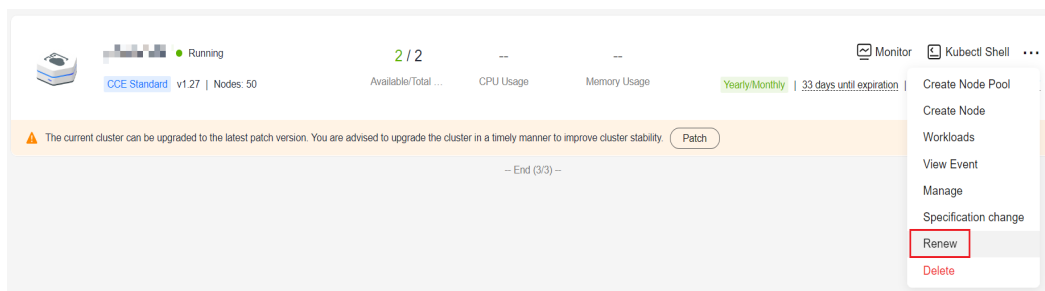
**NOTICE**

A yearly/monthly-billed cluster will be deleted if it is not renewed after expiration, and all nodes and the running services in the cluster will be destroyed. CCE strongly recommends that you renew the cluster before it expires or .

**Step 1** Log in to the CCE console. In the navigation pane, choose **Clusters**.

**Step 2** Locate the cluster to be renewed, click ... to view more operations on the cluster, and choose **Renew**.

**Figure 2-8** Renewing a cluster



**Step 3** On the displayed page, renew the service as prompted.

**NOTE**

- If the selected resource (highlighted) is associated with other resources, you can decide whether you want to perform the operation on all these resources at the same time.
- If you change the resource specifications before its renewal period takes effect, the renewal period cannot be unsubscribed from.
- Renewed resources are not eligible of a 5-day unconditional unsubscription.

**Step 4** Click **Pay**. On the page displayed, review the order amount, select a payment method, and click **Pay**.

**Step 5** After the payment is complete, you can go back to the **Orders** or **Renewals** page to view and manage your order.

----End

## 2.4.8 Changing the Billing Mode of a Cluster from Pay-per-Use to Yearly/Monthly

A cluster can be billed on a pay-per-use or yearly/monthly basis. The billing mode of a cluster can be changed from pay-per-use to yearly/monthly.

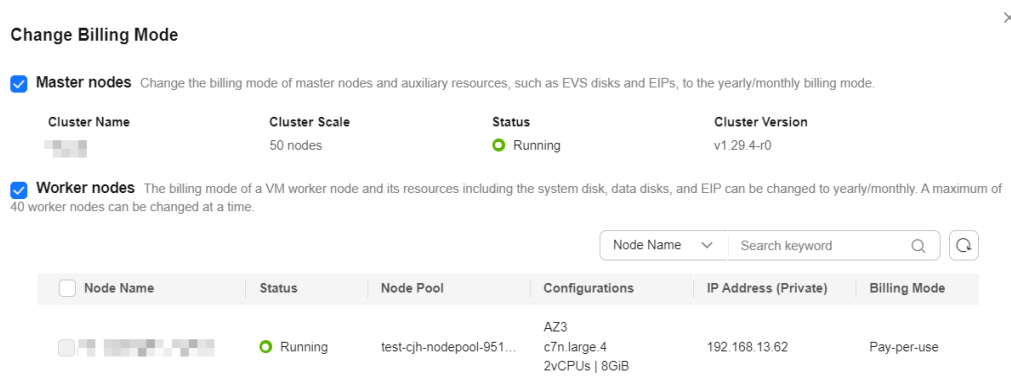
To change the billing mode of a node from pay-per-use to yearly/monthly, see [Changing the Billing Mode of a Node to Yearly/Monthly](#).

### Changing the Billing Mode of a Cluster

To change the billing mode of a cluster from pay-per-use to yearly/monthly, perform the following steps:

- Step 1** Log in to the CCE console. In the navigation pane, choose **Clusters**.
- Step 2** Locate the target cluster, click ... to view more operations on the cluster, and choose **Change Billing Mode**.
- Step 3** On the page displayed, select the target cluster. You can also select the nodes whose billing modes you want to change.

**Figure 2-9** Changing the billing mode of a cluster to yearly/monthly



- Step 4** Click **OK**. Wait until the order is processed and the payment is complete.

----End

## 2.5 Upgrading a Cluster

### 2.5.1 Process and Method of Upgrading a Cluster

CCE strictly complies with community consistency authentication. It releases three Kubernetes versions each year and offers a maintenance period of at least 24 months after each version is released. CCE ensures the stable running of Kubernetes versions during the maintenance period.

To ensure your service rights and benefits, upgrade your Kubernetes clusters before a maintenance period ends. You can check the Kubernetes version of your cluster on the cluster list page and check whether a new version is available. Proactive cluster upgrades help you:

- Reduce security and stability risks: During the iteration of Kubernetes versions, known security and stability vulnerabilities are continuously fixed. Long-term use of EOS clusters will result in security and stability risks to services.
- Experience the latest functions: During the iteration of Kubernetes versions, new functions and optimizations are continuously released. For details about the features of the latest version, see [Release Notes for CCE Cluster Versions](#).
- Minimize compatibility risks: During the iteration of Kubernetes versions, APIs are continuously modified and functions are deprecated. If a cluster has not been upgraded for a long time, more O&M assurance investment will be required when the cluster is upgraded. Periodic upgrades can effectively mitigate compatibility risks caused by accumulated version differences. It is a

good practice to upgrade a patch version every quarter and upgrade a major version to the latest version every year.

- Obtain more effective technical support: CCE does not provide security patches or issue fixing for EOS Kubernetes cluster versions, and does not ensure technical support for the EOS versions.

## Cluster Upgrade Path

CCE clusters evolve iteratively based on the community Kubernetes version. A CCE cluster version consists of the community Kubernetes version and the CCE patch version. Therefore, two cluster upgrade paths are provided.

- Upgrading a Kubernetes version

Source Kubernetes Version	Target Kubernetes Version
v1.13 or earlier	Not supported
v1.15	v1.19
v1.17	v1.19
v1.19	v1.21 or v1.23
v1.21	v1.23 or v1.25
v1.23	v1.25, v1.27, or v1.28
v1.25	v1.27 or v1.28
v1.27	v1.28
v1.28	v1.29
v1.29	/

### NOTE

- A version that has been end of maintenance cannot be directly upgraded to the latest version. You need to upgrade such a version for multiple times, for example, from v1.15 to v1.19, v1.23, and then to v1.27 or v1.28.
- A Kubernetes version can be upgraded only after the patch is upgraded to the latest version. CCE will automatically generate an optimal upgrade path on the console based on the current cluster version.

- Upgrading a patch version

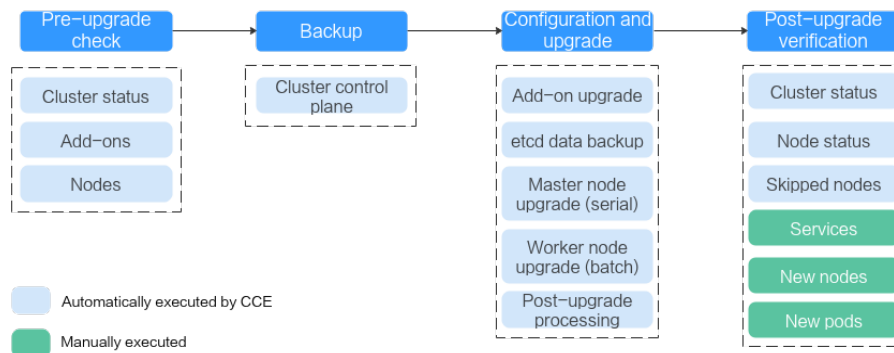
Patch version management is available for CCE clusters of v1.19 or later to provide new features and fix bugs and vulnerability for in-maintenance clusters without requiring a major version upgrade.

After a new patch version is released, you can directly upgrade any patch version to the latest patch version. For details about the release history of patch versions, see [Patch Version Release Notes](#).

## Cluster Upgrade Process

The cluster upgrade process involves pre-upgrade check, backup, upgrade, and post-upgrade verification.

**Figure 2-10** Process of upgrading a cluster



After determining the target version of the cluster, read the [precautions](#) carefully and prevent function incompatibility during the upgrade.

### 1. Pre-upgrade check

Before a cluster upgrade, CCE checks mandatory items such as the cluster status, add-ons, and nodes to ensure that the cluster meets the upgrade requirements. For more details, see [Pre-upgrade Check](#). If any check item is abnormal, rectify the fault as prompted on the console.

### 2. Backup

You can use disk snapshots to back up master node data, including CCE component images, component configurations, and etcd data. Back up data before an upgrade. If unexpected cases occur during an upgrade, you can use the backup to quickly restore the cluster.


Backup Type	Backup Object	Backup Mode	Backup Duration	Rollback Duration	Description
etcd data backup	etcd data	Automatic backup during an upgrade	1-5 minutes	2 hours	Mandatory. The data is automatically backed up during an upgrade.

### 3. Configuration and upgrade

Configure parameters before an upgrade. CCE has provided default settings, which can be modified as needed. After the configuration, upgrade add-ons, master nodes, and worker nodes in sequence.

- **Add-on Upgrade Configuration:** Add-ons that have been installed in your cluster are listed. During the cluster upgrade, CCE automatically upgrades the selected add-ons to be compatible with the target cluster version. You can click **Set** to re-define the add-on parameters.

 NOTE

If an add-on is marked with  on its right side, the add-on cannot be compatible with both the source and target versions of the cluster upgrade. In this case, CCE will upgrade the add-on after the cluster upgrade. The add-on may be unavailable during the cluster upgrade.

- **Node Upgrade Configuration**

- **Max. Nodes for Batch Upgrade:** You can configure the maximum number of nodes to be upgraded in a batch.  
Node pools will be upgraded in sequence. Nodes in node pools will be upgraded in batches. One node is upgraded in the first batch, two nodes in the second batch, and the number of nodes to be upgraded in each subsequent batch increases by a power of 2 until the maximum number of nodes to be upgraded in each batch is reached. The next cluster is upgraded after the previous one is upgraded. By default, 20 nodes are upgraded in a batch, and the number can be increased to the maximum of 60.
- **Node Priority:** You can customize node upgrade priorities. If the priorities are not specified, CCE will perform the upgrade based on the priorities generated by the default policy.
  - **Add Upgrade Priority:** You can custom the priorities for upgrading node pools. If the priorities are not specified, CCE will preferentially upgrade the node pool with the least number of nodes based on the default policy.
  - **Add Node Priority:** You can custom the priorities for upgrading nodes in a node pool. If the priorities are not specified, CCE will preferentially upgrade the node with lightest load (calculated based on the number of pods, resource request rate, and number of PVs) based on the default policy.
- **Scope of Node Upgrade Batches:** By default, this parameter is set to a cluster, but it can be customized.
  - If the scope is set to a cluster, the upgrade batch will remain unchanged throughout the entire upgrade process.
  - If the scope is set to node pools, the upgrade batch will be reset for each node pool separately.

4. **Post-upgrade verification**

After an upgrade, CCE will automatically check items including the cluster status and node status. You need to manually check services, new nodes, and new pods to ensure that the cluster functions properly after the upgrade. For details, see [Performing Post-Upgrade Verification](#).

## Upgrade Modes

Table 2-20 Upgrade modes

Upgrade Mode	Description	Upgrade Scope	Advantage	Constraint
In-place upgrade	<p>Kubernetes components, network components, and CCE management components are upgraded on nodes. During an upgrade, service pods and networks are not affected.</p> <p>Nodes are upgraded in batches. Only the nodes that have been upgraded can be used to schedule services.</p>	<ul style="list-style-type: none"> <li>Node OSs are not upgraded.</li> <li>The addons that are incompatible with the target cluster version will be automatically upgraded.</li> <li>Kubernetes components will be automatically upgraded.</li> </ul>	The one-click upgrade does not need to migrate services. This ensures service continuity.	In-place upgrade is supported only in clusters of v1.15 or later.

### 2.5.2 Before You Start

Before the upgrade, you can check whether your cluster can be upgraded and which versions are available on the CCE console. For details, see [Process and Method of Upgrading a Cluster](#).

#### Precautions

Before upgrading a cluster, pay attention to the following points:

- **Perform an upgrade during off-peak hours to minimize the impact on your services.**
- Before upgrading a cluster, learn about the features and differences of each cluster version in [Kubernetes Release Notes](#) to prevent exceptions due to the use of an incompatible cluster version. For example, check whether any APIs deprecated in the target version are used in the cluster. Otherwise, calling the APIs may fail after the upgrade. For details, see [Deprecated APIs](#).

During a cluster upgrade, pay attention to the following points that may affect your services:

- During a cluster upgrade, do not perform any operation on the cluster. If you **stop, restart, or delete nodes** while upgrading the cluster, the upgrade will fail.
- Before upgrading a cluster, **ensure no high-risk operations are performed in the cluster**. Otherwise, the cluster upgrade may fail or the configuration may be lost after the upgrade. Common high-risk operations include modifying cluster node configurations locally and modifying the configurations of the listeners managed by CCE on the ELB console. Instead, modify configurations on the CCE console so that the modifications can be automatically inherited during the upgrade.
- During a cluster upgrade, the running workloads will not be interrupted, but access to the API server will be temporarily interrupted.
- By default, application scheduling is not restricted during a cluster upgrade. During an upgrade of the following early cluster versions, the **node.kubernetes.io/upgrade** taint (equivalent to **NoSchedule**) will be added to the nodes in the cluster and removed after the cluster is upgraded:
  - All v1.15 clusters
  - All v1.17 clusters
  - v1.19 clusters with patch versions earlier than or equal to v1.19.16-r4
  - v1.21 clusters with patch versions earlier than or equal to v1.21.7-r0
  - v1.23 clusters with patch versions earlier than or equal to v1.23.5-r0

## Notes and Constraints

- If an error occurred during a cluster upgrade, the cluster can be rolled back using the backup data. If you perform other operations (for example, modifying cluster specifications) after a successful cluster upgrade, the cluster cannot be rolled back using the backup data.
- When clusters using the tunnel network model are upgraded to v1.19.16-r4, v1.21.7-r0, v1.23.5-r0, v1.25.1-r0, or later, the SNAT rule whose destination address is the container CIDR block but the source address is not the container CIDR block will be removed. If you have configured VPC routes to directly access all pods outside the cluster, only the pods on the corresponding nodes can be directly accessed after the upgrade.
- The new add-on versions (see [Change History](#)) of the NGINX Ingress Controller allow graceful exit and the grace period for deleting the ELB backend controller and support hitless upgrades. During the upgrade of the add-on versions listed in the following table, services may be unavailable for a short period of time. If the following add-on versions have been installed in the cluster, perform the upgrade during off-peak hours.

Add-on Version	Version Range
2.1.x	x < 32
2.2.x	x < 41
2.4.x	x < 4

- For more details, see [Version Differences](#).

## Version Differences

Upgrade Path	Version Difference	Self-Check
v1.23 or v1.25 Upgraded to v1.27	Docker is no longer recommended. Use containerd instead. For details, see <a href="#">Container Engines</a> .	This item has been included in the pre-upgrade check.
v1.21 or v1.19 Upgraded to v1.23	For the NGINX Ingress Controller of an earlier version (community version v0.49 or earlier, or CCE nginx-ingress version v1.x.x), the created ingresses can be managed by the NGINX Ingress Controller even if <b>kubernetes.io/ingress.class: nginx</b> is not set in the <b>ingress annotations</b> . However, for the NGINX Ingress Controller of a later version (community version v1.0.0 or later, or CCE nginx-ingress version v2.x.x), the ingresses created without specifying the Nginx type will not be managed by the NGINX Ingress Controller, and ingress rules will become invalid, which interrupts services.	This item has been included in the pre-upgrade check. You can also perform the self-check by referring to <a href="#">nginx-ingress</a> .
v1.19 to v1.21	The bug of <b>exec probe timeouts</b> is fixed in Kubernetes 1.21. Before this bug is fixed, the exec probe does not consider the <b>timeoutSeconds</b> field. Instead, the probe will run indefinitely, even beyond its configured deadline. It will stop until the result is returned. If this field is not specified, the default value <b>1</b> is used. This field takes effect after the upgrade. If the probe runs over 1 second, the application health check may fail and the application may restart frequently.	Before the upgrade, check whether the timeout is properly set for the exec probe.



Upgrade Path	Version Difference	Self-Check
	<p>kube-apiserver of CCE v1.19 or later requires that the Subject Alternative Names (SANs) field be configured for the certificate of your webhook server. Otherwise, kube-apiserver fails to call the webhook server after the upgrade, and containers cannot be started properly.</p> <p>Root cause: X.509 <b>CommonName</b> is discarded in Go v1.15. kube-apiserver of CCE v1.19 is compiled using Go v1.15. If your webhook certificate does not have SANs, kube-apiserver does not process the <b>CommonName</b> field of the X.509 certificate as the host name by default. As a result, the authentication fails.</p>	<p>Before the upgrade, check whether the SAN field is configured in the certificate of your webhook server.</p> <ul style="list-style-type: none"> <li>• If you do not have your own webhook server, you can skip this check.</li> <li>• If the field is not set, use the SAN field to specify the IP address and domain name supported by the certificate.</li> </ul>

**Table 2-21** QoS class changes before and after the upgrade

Init Container (Calculated Based on spec.initContainers)	Service Container (Calculated Based on spec.containers)	Pod (Calculated Based on spec.containers and spec.initContainers)	Impacted or Not
Guaranteed	Besteffort	Burstable	Yes
Guaranteed	Burstable	Burstable	No
Guaranteed	Guaranteed	Guaranteed	No
Besteffort	Besteffort	Besteffort	No
Besteffort	Burstable	Burstable	No
Besteffort	Guaranteed	Burstable	Yes
Burstable	Besteffort	Burstable	Yes
Burstable	Burstable	Burstable	No
Burstable	Guaranteed	Burstable	Yes

## Deprecated APIs

With the evolution of Kubernetes APIs, APIs are periodically reorganized or upgraded, and certain APIs are deprecated and finally deleted. The following tables list the deprecated APIs in each Kubernetes community version. For details about more deprecated APIs, see [Deprecated API Migration Guide](#).

- [APIs Deprecated in Kubernetes v1.29](#)
- No APIs deprecated in Kubernetes v1.28
- [APIs Deprecated in Kubernetes v1.27](#)
- [APIs Deprecated in Kubernetes v1.25](#)
- [APIs Deprecated in Kubernetes v1.22](#)
- [APIs Deprecated in Kubernetes v1.16](#)

 NOTE

When an API is deprecated, the existing resources are not affected. However, when you create or edit the resources, the API version will be intercepted.

**Table 2-22** APIs deprecated in Kubernetes v1.29

Resource	Deprecated API Version	Substitute API Version	Change Description
FlowSchema and PriorityLevelConfiguration	flowcontrol.apiserver.k8s.io/v1beta2	flowcontrol.apiserver.k8s.io/v1 (This API has been available since v1.29.) flowcontrol.apiserver.k8s.io/v1beta3 (This API has been available since v1.26.)	<ul style="list-style-type: none"> <li>• Significant changes in <b>flowcontrol.apiserver.k8s.io/v1</b>: <b>spec.limited.assuredConcurrencyShares</b> of <b>PriorityLevelConfiguration</b> has been renamed <b>spec.limited.nominalConcurrencyShares</b>. The default value is <b>30</b> only when it is not specified, and the explicit value <b>0</b> does not change to <b>30</b>.</li> <li>• Key changes in <b>flowcontrol.apiserver.k8s.io/v1beta3</b>: <b>spec.limited.assuredConcurrencyShares</b> of <b>PriorityLevelConfiguration</b> has been renamed <b>spec.limited.nominalConcurrencyShares</b>.</li> </ul>

**Table 2-23** APIs deprecated in Kubernetes v1.27

Resource	Deprecated API Version	Substitute API Version	Change Description
CSIStorageCapacity	storage.k8s.io/v1beta1	storage.k8s.io/v1 (This API has been available since v1.24.)	None
FlowSchema and PriorityLevelConfiguration	flowcontrol.apiserver.k8s.io/v1beta1	flowcontrol.apiserver.k8s.io/v1beta3 (This API has been available since v1.26.)	None
HorizontalPodAutoscaler	autoscaling/v2beta2	autoscaling/v2 (This API has been available since v1.23.)	None

**Table 2-24** APIs deprecated in Kubernetes v1.25

Resource	Deprecated API Version	Substitute API Version	Change Description
CronJob	batch/v1beta1	batch/v1 (This API has been available since v1.21.)	None
EndpointSlice	discovery.k8s.io/v1beta1	discovery.k8s.io/v1 (This API has been available since v1.21.)	Pay attention to the following changes: <ul style="list-style-type: none"> <li>• In each endpoint, the <b>topology["kubernetes.io/hostname"]</b> field has been deprecated. Replace it with the <b>nodeName</b> field.</li> <li>• In each endpoint, the <b>topology["kubernetes.io/zone"]</b> field has been deprecated. Replace it with the <b>zone</b> field.</li> <li>• The <b>topology</b> field is replaced with <b>deprecatedTopology</b> and cannot be written in v1.</li> </ul>

Resource	Deprecated API Version	Substitute API Version	Change Description
Event	events.k8s.io/v1beta1	events.k8s.io/v1  (This API has been available since v1.19.)	<p>Pay attention to the following changes:</p> <ul style="list-style-type: none"> <li>• The <b>type</b> field can only be set to <b>Normal</b> or <b>Warning</b>.</li> <li>• The <b>involvedObject</b> field is renamed <b>regarding</b>.</li> <li>• The <b>action</b>, <b>reason</b>, <b>reportingController</b>, and <b>reportingInstance</b> fields are mandatory for creating a new <b>events.k8s.io/v1</b> event.</li> <li>• Use <b>eventTime</b> instead of the deprecated <b>firstTimestamp</b> field (this field has been renamed <b>deprecatedFirstTimestamp</b> and is not allowed to appear in the new <b>events.k8s.io/v1</b> event object).</li> <li>• Use <b>series.lastObservedTime</b> instead of the deprecated <b>lastTimestamp</b> field (this field has been renamed <b>deprecatedLastTimestamp</b> and is not allowed to appear in the new <b>events.k8s.io/v1</b> event object).</li> <li>• Use <b>series.count</b> instead of the deprecated <b>count</b> field (this field has been renamed <b>deprecatedCount</b> and is not allowed to appear in the new <b>events.k8s.io/v1</b> event object).</li> <li>• Use <b>reportingController</b> instead of the deprecated <b>source.component</b> field (this field has been renamed <b>deprecatedSource.component</b> and is not allowed to appear in the new</li> </ul>

Resource	Deprecated API Version	Substitute API Version	Change Description
			<p><b>events.k8s.io/v1</b> event object).</p> <ul style="list-style-type: none"> <li>Use <b>reportingInstance</b> instead of the deprecated <b>source.host</b> field (this field has been renamed <b>deprecatedSource.host</b> and is not allowed to appear in the new <b>events.k8s.io/v1</b> event object).</li> </ul>
HorizontalPod Autoscaler	autoscaling/v2beta1	autoscaling/v2 (This API has been available since v1.23.)	None
PodDisruption Budget	policy/v1beta1	policy/v1 (This API has been available since v1.21.)	If <b>spec.selector</b> is set to null ( <b>{}</b> ) in <b>PodDisruptionBudget</b> of <b>policy/v1</b> , all pods in the namespace are selected. (In <b>policy/v1beta1</b> , an empty <b>spec.selector</b> means that no pod will be selected.) If <b>spec.selector</b> is not specified, pod will be selected in neither API version.
PodSecurityPolicy	policy/v1beta1	None	Since v1.25, the PodSecurityPolicy resource no longer provides APIs of the <b>policy/v1beta1</b> version, and the PodSecurityPolicy access controller is deleted. Use <b>Pod Security Admission</b> instead.
RuntimeClass	node.k8s.io/v1beta1	node.k8s.io/v1 (This API has been available since v1.20.)	None

**Table 2-25** APIs deprecated in Kubernetes v1.22

Resource	Deprecated API Version	Substitute API Version	Change Description
MutatingWebhookConfiguration ValidatingWebhookConfiguration	admissionregistration.k8s.io/v1beta1	admissionregistration.k8s.io/v1 (This API has been available since v1.16.)	<ul style="list-style-type: none"> <li>• The default value of <b>webhooks[*].failurePolicy</b> is changed from <b>Ignore</b> to <b>Fail</b> in v1.</li> <li>• The default value of <b>webhooks[*].matchPolicy</b> is changed from <b>Exact</b> to <b>Equivalent</b> in v1.</li> <li>• The default value of <b>webhooks[*].timeoutSeconds</b> is changed from <b>30s</b> to <b>10s</b> in v1.</li> <li>• The default value of <b>webhooks[*].sideEffects</b> is deleted, and this field must be specified. In v1, the value can only be <b>None</b> or <b>NoneOnDryRun</b>.</li> <li>• The default value of <b>webhooks[*].admissionReviewVersions</b> is deleted. In v1, this field must be specified. (<b>AdmissionReview</b> v1 and v1beta1 are supported.)</li> <li>• <b>webhooks[*].name</b> must be unique in the list of objects created through <b>admissionregistration.k8s.io/v1</b>.</li> </ul>

Resource	Deprecated API Version	Substitute API Version	Change Description
CustomResourceDefinition	apiextensions.k8s.io/v1beta1	apiextensions/v1 (This API has been available since v1.16.)	<ul style="list-style-type: none"> <li>• The default value of <b>spec.scope</b> is no longer <b>Namespaced</b>. This field must be explicitly specified.</li> <li>• <b>spec.version</b> is deleted from v1. Use <b>spec.versions</b> instead.</li> <li>• <b>spec.validation</b> is deleted from v1. Use <b>spec.versions[*].schema</b> instead.</li> <li>• <b>spec.subresources</b> is deleted from v1. Use <b>spec.versions[*].subresources</b> instead.</li> <li>• <b>spec.additionalPrinterColumns</b> is deleted from v1. Use <b>spec.versions[*].additionalPrinterColumns</b> instead.</li> <li>• <b>spec.conversion.webhookClientConfig</b> is moved to <b>spec.conversion.webhook.clientConfig</b> in v1.</li> <li>• <b>spec.conversion.conversionReviewVersions</b> is moved to <b>spec.conversion.webhook.conversionReviewVersions</b> in v1.</li> <li>• <b>spec.versions[*].schema.openAPIV3Schema</b> becomes a mandatory field when the <b>CustomResourceDefinition</b> object of the v1 version is created, and its value must be a <b>structural schema</b>.</li> <li>• <b>spec.preserveUnknownFields: true</b> cannot be specified when the <b>CustomResourceDefinition</b> object of the v1 version is created. This configuration must be specified using <b>x-kubernetes-preserve-</b></li> </ul>

Resource	Deprecated API Version	Substitute API Version	Change Description
			<p><b>unknown-fields: true</b> in the schema definition.</p> <ul style="list-style-type: none"> <li>In v1, the <b>JSONPath</b> field in the <b>additionalPrinterColumns</b> entry is renamed <b>jsonPath</b> (patch <a href="#">#66531</a>).</li> </ul>
APIService	apiregistration/v1beta1	apiregistration.k8s.io/v1 (This API has been available since v1.10.)	None
TokenReview	authentication.k8s.io/v1beta1	authentication.k8s.io/v1 (This API has been available since v1.6.)	None
LocalSubjectAccessReview SelfSubjectAccessReview SubjectAccessReview SelfSubjectRulesReview	authorization.k8s.io/v1beta1	authorization.k8s.io/v1 (This API has been available since v1.16.)	<b>spec.group</b> was renamed <b>spec.groups</b> in v1 (patch <a href="#">#32709</a> ).



Resource	Deprecated API Version	Substitute API Version	Change Description
CertificateSigningRequest	certificates.k8s.io/v1beta1	certificates.k8s.io/v1 (This API has been available since v1.19.)	<p>Pay attention to the following changes in <b>certificates.k8s.io/v1</b>:</p> <ul style="list-style-type: none"> <li>• For an API client that requests a certificate: <ul style="list-style-type: none"> <li>- <b>spec.signerName</b> becomes a mandatory field (see <a href="#">Known Kubernetes Signers</a>). In addition, the <b>certificates.k8s.io/v1</b> API cannot be used to create requests whose signer is <b>kubernetes.io/legacy-unknown</b>.</li> <li>- <b>spec.usages</b> now becomes a mandatory field, which cannot contain duplicate string values and can contain only known usage strings.</li> </ul> </li> <li>• For an API client that needs to approve or sign a certificate: <ul style="list-style-type: none"> <li>- <b>status.conditions</b> cannot contain duplicate types.</li> <li>- The <b>status.conditions[*].status</b> field is now mandatory.</li> <li>- The <b>status.certificate</b> must be PEM-encoded and can contain only the <b>CERTIFICATE</b> data block.</li> </ul> </li> </ul>
Lease	coordination.k8s.io/v1beta1	coordination.k8s.io/v1 (This API has been available since v1.14.)	None

Resource	Deprecated API Version	Substitute API Version	Change Description
Ingress	networking.k8s.io/v1beta1 extensions/v1beta1	networking.k8s.io/v1 (This API has been available since v1.19.)	<ul style="list-style-type: none"> <li>The <b>spec.backend</b> field is renamed <b>spec.defaultBackend</b>.</li> <li>The <b>serviceName</b> field of the backend is renamed <b>service.name</b>.</li> <li>The backend <b>servicePort</b> field represented by a number is renamed <b>service.port.number</b>.</li> <li>The backend <b>servicePort</b> field represented by a string is renamed <b>service.port.name</b>.</li> <li>The <b>pathType</b> field is mandatory for all paths to be specified. The options are <b>Prefix</b>, <b>Exact</b>, and <b>ImplementationSpecific</b>. To match the behavior of not defining the path type in v1beta1, use <b>ImplementationSpecific</b>.</li> </ul>
IngressClass	networking.k8s.io/v1beta1	networking.k8s.io/v1 (This API has been available since v1.19.)	None
ClusterRole ClusterRoleBinding Role RoleBinding	rbac.authorization.k8s.io/v1beta1	rbac.authorization.k8s.io/v1 (This API has been available since v1.8.)	None
PriorityClass	scheduling.k8s.io/v1beta1	scheduling.k8s.io/v1 (This API has been available since v1.14.)	None

Resource	Deprecated API Version	Substitute API Version	Change Description
CSIDriver CSINode StorageClass VolumeAttachment	storage.k8s.io/v1beta1	storage.k8s.io/v1	<ul style="list-style-type: none"> <li>CSIDriver is available in <b>storage.k8s.io/v1</b> since v1.19.</li> <li>CSINode is available in <b>storage.k8s.io/v1</b> since v1.17.</li> <li>StorageClass is available in <b>storage.k8s.io/v1</b> since v1.6.</li> <li>VolumeAttachment is available in <b>storage.k8s.io/v1</b> since v1.13.</li> </ul>

**Table 2-26** APIs deprecated in Kubernetes v1.16

Resource	Deprecated API Version	Substitute API Version	Change Description
NetworkPolicy	extensions/v1beta1	networking.k8s.io/v1  (This API has been available since v1.8.)	None
DaemonSet	extensions/v1beta1 apps/v1beta2	apps/v1  (This API has been available since v1.9.)	<ul style="list-style-type: none"> <li>The <b>spec.templateGeneration</b> field is deleted.</li> <li><b>spec.selector</b> is now a mandatory field and cannot be changed after the object is created. The label of an existing template can be used as a selector for seamless migration.</li> <li>The default value of <b>spec.updateStrategy.type</b> is changed to <b>RollingUpdate</b> (the default value in the <b>extensions/v1beta1</b> API version is <b>OnDelete</b>).</li> </ul>

Resource	Deprecated API Version	Substitute API Version	Change Description
Deployment	extensions/v1beta1 apps/v1beta1 apps/v1beta2	apps/v1 (This API has been available since v1.9.)	<ul style="list-style-type: none"> <li>The <b>spec.rollbackTo</b> field is deleted.</li> <li><b>spec.selector</b> is now a mandatory field and cannot be changed after the Deployment is created. The label of an existing template can be used as a selector for seamless migration.</li> <li>The default value of <b>spec.progressDeadlineSeconds</b> is changed to 600 seconds (the default value in <b>extensions/v1beta1</b> is unlimited).</li> <li>The default value of <b>spec.revisionHistoryLimit</b> is changed to 10. (In the <b>apps/v1beta1</b> API version, the default value of this field is 2. In the <b>extensions/v1beta1</b> API version, all historical records are retained by default.)</li> <li>The default values of <b>maxSurge</b> and <b>maxUnavailable</b> are changed to 25%. (In the <b>extensions/v1beta1</b> API version, these fields default to 1.)</li> </ul>
StatefulSet	apps/v1beta1 apps/v1beta2	apps/v1 (This API has been available since v1.9.)	<ul style="list-style-type: none"> <li><b>spec.selector</b> is now a mandatory field and cannot be changed after the StatefulSet is created. The label of an existing template can be used as a selector for seamless migration.</li> <li>The default value of <b>spec.updateStrategy.type</b> is changed to <b>RollingUpdate</b> (the default value in the <b>apps/v1beta1</b> API version is <b>OnDelete</b>).</li> </ul>

Resource	Deprecated API Version	Substitute API Version	Change Description
ReplicaSet	extensions/v1beta1 apps/v1beta1 apps/v1beta2	apps/v1 (This API has been available since v1.9.)	<b>spec.selector</b> is now a mandatory field and cannot be changed after the object is created. The label of an existing template can be used as a selector for seamless migration.
PodSecurityPolicy	extensions/v1beta1	policy/v1beta1 (This API has been available since v1.10.)	PodSecurityPolicy for the <b>policy/v1beta1</b> API version will be removed in v1.25.

## Upgrade Backup

The following table lists how to back up cluster data.

Backup Type	Backup Object	Backup Method	Backup Duration	Rollback Duration	Description
etcd data backup	etcd data	Automatic backup during an upgrade	1-5 minutes	2 hours	Mandatory. The data is automatically backed up during an upgrade.

## 2.5.3 Performing Post-Upgrade Verification

### 2.5.3.1 Cluster Status Check

#### Check Items

After a cluster is upgraded, check whether the cluster is in the **Running** state.

#### Procedure

CCE automatically checks your cluster status. Go to the cluster list page and confirm the cluster status based on the diagnosis result.

#### Solution

If your cluster malfunctions, contact technical support.

### 2.5.3.2 Node Status Check

#### Check Items

After a cluster is upgraded, check whether nodes in the cluster are in the **Running** state.

#### Procedure

CCE automatically checks your node statuses. Go to the node list page and confirm the node statuses based on the diagnosis result.

#### Solution

If a node malfunctions, [reset the node](#). If the fault persists, contact technical support.

### 2.5.3.3 Node Skipping Check

#### Check Items

After a cluster is upgraded, check whether there are any nodes that skip the upgrade in the cluster. These nodes may affect the proper running of the cluster.

#### Procedure

CCE automatically checks whether there are nodes that skip the upgrade in the cluster. Go to the node list page and confirm the nodes based on the diagnosis result. The skipped nodes are labeled with **upgrade.cce.io/skipped=true**.

#### Solution

The skipped nodes are displayed on the upgrade details page. Reset the skipped nodes after the upgrade is complete. For details about how to reset a node, see [Resetting a Node](#).

#### NOTE

Resetting a node will reset all node labels, which may affect workload scheduling. Before resetting a node, check and retain the labels that you have manually added to the node.

### 2.5.3.4 Service Check

#### Check Items

After a cluster is upgraded, check whether its services are running properly.

#### Procedure

Different services have different verification mode. Select a suitable one and verify the service before and after the upgrade.

You can verify the service from the following aspects:

- The service page is available.
- No alarm or event is generated on the normal platform.
- No error log is generated for key processes.
- The API dialing test is normal.

## Solution

If your online services malfunction after the cluster upgrade, contact technical support.

### 2.5.3.5 New Node Check

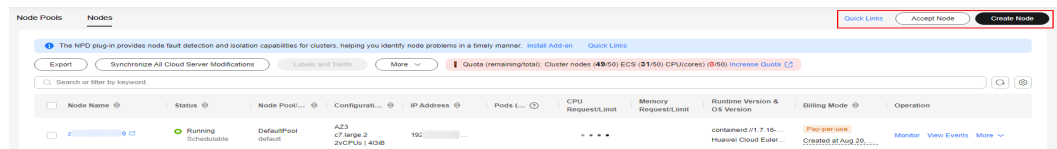
#### Check Items

Check whether nodes can be created in the cluster.

#### Procedure

- 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 **Nodes**. On the displayed page, click the **Nodes** tab and then **Create Node**. For details about the node configuration, see [Creating a Node](#).

Figure 2-11 Creating a node



----End

## Solution

If nodes cannot be created in your cluster after the cluster is upgraded, contact technical support.

### 2.5.3.6 New Pod Check

#### Check Items

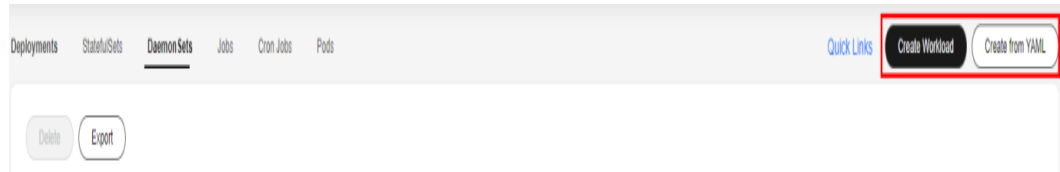
- Check whether pods can be created on the existing nodes after the cluster is upgraded.
- Check whether pods can be created on new nodes after the cluster is upgraded.

#### Procedure

After creating a node based on [New Node Check](#), create a DaemonSet workload to create pods on each node.

- 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 **Workloads**. On the displayed page, click **Create Workload** or **Create from YAML** in the upper right corner. For details about how to create a DaemonSet, see [Creating a DaemonSet](#).

**Figure 2-12** Creating a DaemonSet



It is a good practice to use the image for routine tests as the base image. You can deploy minimum pods for an application by referring to the following YAML file.

**NOTE**

In this test, YAML deploys DaemonSet in the default namespace, uses **nginx:perl** as the base image, requests 10m vCPUs and 10 MiB memory, and limits 100 MB CPU and 50 MiB memory.

```
apiVersion: apps/v1
kind: DaemonSet
metadata:
  name: post-upgrade-check
  namespace: default
spec:
  selector:
    matchLabels:
      app: post-upgrade-check
      version: v1
  template:
    metadata:
      labels:
        app: post-upgrade-check
        version: v1
    spec:
      containers:
        - name: container-1
          image: nginx:perl
          imagePullPolicy: IfNotPresent
      resources:
        requests:
          cpu: 10m
          memory: 10Mi
        limits:
          cpu: 100m
          memory: 50Mi
```

- Step 3** After the workload is created, check whether the pods of the workload are running properly.
- Step 4** After the check is complete, choose **Workloads** in the navigation pane. On the displayed page, click the **DaemonSets** tab, locate the **post-upgrade-check** workload, and choose **More > Delete** in the **Operation** column to delete the test workload.

----End



## Solution

If the pod cannot be created or the pod status is abnormal, contact technical support and specify whether the exception occurs on new nodes or existing nodes.

## 2.5.4 Migrating Services Across Clusters of Different Versions

### Application Scenarios

This section describes how to migrate services from a cluster of an earlier version to a cluster of a later version in CCE.

This operation is applicable when a cross-version cluster upgrade is required (for example, upgrade from v1.19.\* to v1.28.\*) and new clusters can be created for service migration.

### Prerequisites

**Table 2-27** Checklist before migration

Category	Description
Cluster	NodeIP-related: Check whether node IP addresses (including EIPs) of the cluster before the migration have been used in other configurations or whitelists.
Workloads	Record the number of workloads for post-migration check.
Storage	<ol style="list-style-type: none"> <li>1. Check whether the storage resources in use are provisioned by the cloud or by your organization.</li> <li>2. Change the automatically created storage to the existing storage in the new cluster.</li> </ol>
Network	<ol style="list-style-type: none"> <li>1. Pay special attention to the ELB and ingress.</li> <li>2. Clusters of an earlier version support only the classic load balancer. To migrate services to a new cluster, change load balancer type to shared load balancer. Then, the corresponding ELB service will be re-established.</li> </ol>
O&M	Private configuration: Check whether kernel parameters or system data have been configured on nodes in the cluster.

### Procedure

#### Step 1 Create a CCE cluster.

Create a cluster with the same specifications and configurations as the cluster of the earlier version. For details, see [Buying a CCE Standard/Turbo Cluster](#).

#### Step 2 Add a node.

Add a node with the same specifications and manual configuration items. For details, see [Creating a Node](#).

### Step 3 Create a storage volume in the new cluster.

Use the existing storage to create a PVC in the new cluster. The PVC name remains unchanged. For details, see [Using an Existing OBS Bucket Through a Static PV](#) or [Using an Existing SFS Turbo File System Through a Static PV](#).

#### NOTE

Storage switching supports only shared storage such as OBS and SFS Turbo. If non-shared storage is used, suspend the workloads in the old cluster to switch the storage resources. In this case, services will be unavailable.

### Step 4 Create a workload in the new cluster.

Create a workload in the new cluster. The name and specifications remain unchanged. For details, see [Creating a Deployment](#) or [Creating a StatefulSet](#).

### Step 5 Mount the storage again.

Remount the existing storage in the workload. For details, see [Using an Existing OBS Bucket Through a Static PV](#) or [Using an Existing SFS Turbo File System Through a Static PV](#).

### Step 6 Create a Service in the new cluster.

The Service name and specifications remain unchanged. For details about how to create a Service, see [Service](#).

### Step 7 Commission services.

After all resources are created, commission the containerized services. If the commissioning is successful, migrate the services to the new cluster.

### Step 8 Delete the old cluster.

When all functions of the new cluster are stable, delete the old cluster. For details about how to delete a cluster, see [Deleting a Cluster](#).

----End

## 2.5.5 Troubleshooting for Pre-upgrade Check Exceptions

### 2.5.5.1 Pre-upgrade Check

The system automatically checks a cluster before its upgrade. If the cluster does not meet the pre-upgrade check conditions, the upgrade cannot continue. To avoid risks, you can perform pre-upgrade check according to the check items and solutions described in this section.

**Table 2-28** Check items

No.	Check Item	Description
1	<b>Node Restrictions</b>	<ul style="list-style-type: none"> <li>• Check whether the node is available.</li> <li>• Check whether the node OS supports the upgrade.</li> <li>• Check whether the node is marked with unexpected node pool labels.</li> <li>• Check whether the Kubernetes node name is the same as the ECS name.</li> </ul>
2	<b>Upgrade Management</b>	Check whether the target cluster is under upgrade management.
3	<b>Add-ons</b>	<ul style="list-style-type: none"> <li>• Check whether the add-on status is normal.</li> <li>• Check whether the add-on supports the target version.</li> </ul>
4	<b>Helm Charts</b>	Check whether the current HelmRelease record contains discarded Kubernetes APIs that are not supported by the target cluster version. If yes, the Helm chart may be unavailable after the upgrade.
5	<b>SSH Connectivity of Master Nodes</b>	Check whether your master nodes can be accessed using SSH.
6	<b>Security Groups</b>	Check whether the <b>Protocol &amp; Port</b> of the worker node security groups is set to <b>ICMP: All</b> and whether the security group with the source IP address set to the master node security group is deleted.
7	<b>Arm Node Restrictions</b>	<ul style="list-style-type: none"> <li>• Check whether the cluster contains Arm nodes.</li> </ul>
8	<b>Residual Nodes</b>	Check whether nodes need to be migrated.
9	<b>Discarded Kubernetes Resources</b>	Check whether there are discarded resources in the clusters.
10	<b>Compatibility Risks</b>	Read the version compatibility differences and ensure that they are not affected. The patch upgrade does not involve version compatibility differences.
11	<b>CCE Agent Versions</b>	Check whether cce-agent on the current node is of the latest version.
12	<b>Node CPU Usage</b>	Check whether the node's CPU usage is above 90%.

No.	Check Item	Description
13	<b>CRDs</b>	<ul style="list-style-type: none"> <li>• Check whether the key CRD <b>packageversions.version.cce.io</b> of the cluster is deleted.</li> <li>• Check whether the cluster key CRD <b>network-attachment-definitions.k8s.cni.cncf.io</b> is deleted.</li> </ul>
14	<b>Node Disks</b>	<ul style="list-style-type: none"> <li>• Check whether the key data disks on the node meet the upgrade requirements.</li> <li>• Check whether the <b>/tmp</b> directory has 500 MB available space.</li> </ul>
15	<b>Node DNS</b>	<ul style="list-style-type: none"> <li>• Check whether the DNS configuration of the current node can resolve the OBS address.</li> <li>• Check whether the current node can access the OBS address of the storage upgrade component package.</li> </ul>
16	<b>Node Key Directory File Permissions</b>	Check whether the owner and owner group of the files in the <b>/var/paas</b> directory used by the CCE are both <b>paas</b> .
17	<b>kubelet</b>	Check whether the kubelet on the node is running properly.
18	<b>Node Memory</b>	Check whether the node's memory usage is above 90%.
19	<b>Node Clock Synchronization Server</b>	Check whether the clock synchronization server ntpd or chronyd of the node is running properly.
20	<b>Node OS</b>	Check whether the OS kernel version of the node is supported by CCE.
21	<b>Node CPU Cores</b>	Check and make sure that the master nodes in your cluster have more than 2 CPU cores.
22	<b>Node Python Commands</b>	Check whether the Python commands are available on a node.
23	<b>ASM Version</b>	<ul style="list-style-type: none"> <li>• Check whether ASM is used by the cluster.</li> <li>• Check whether the current ASM version supports the target cluster version.</li> </ul>
24	<b>Node Readiness</b>	Check whether the nodes in the cluster are ready.
25	<b>Node journald</b>	Check whether journald of a node is normal.
26	<b>containerd.sock</b>	Check whether the <b>containerd.sock</b> file is on the node. This file affects the startup of container runtime in the Euler OS.

No.	Check Item	Description
27	<b>Internal Error</b>	This check item is not typical and implies that an internal error was found during the pre-upgrade check.
28	<b>Node Mount Points</b>	Check whether there are inaccessible mount points on the node.
29	<b>Kubernetes Node Taints</b>	Check whether the taint needed for cluster upgrade exists on the node.
30	<b>Everest Restrictions</b>	Check whether there are any compatibility restrictions on the current Everest add-on.
31	<b>cce-hpa-controller Limitations</b>	Check whether there are compatibility limitations between the current and target cce-controller-hpa add-on versions.
32	<b>Enhanced CPU Policies</b>	Check whether the current cluster version and the target version support <b>enhanced CPU policy</b> .
33	<b>Health of Worker Node Components</b>	Check whether the container runtime and network components on the worker nodes are healthy.
34	<b>Health of Master Node Components</b>	Check whether cluster components such as the Kubernetes component, container runtime component, and network component are running properly before the upgrade.
35	<b>Memory Resource Limit of Kubernetes Components</b>	Check whether the resources of Kubernetes components, such as etcd and kube-controller-manager, exceed the upper limit.
36	<b>Discarded Kubernetes APIs</b>	<p>The system scans the audit logs of the past day to check whether the user calls the deprecated APIs of the target Kubernetes version.</p> <p><b>NOTE</b> Due to the limited time range of audit logs, this check item is only an auxiliary method. APIs to be deprecated may have been used in the cluster, but their usage is not included in the audit logs of the past day. Check the API usage carefully.</p>
37	<b>IPv6 Support in CCE Turbo Clusters</b>	If IPv6 is enabled for a CCE Turbo cluster, check whether the target cluster version supports IPv6.
38	<b>NetworkManager</b>	Check whether NetworkManager of a node is normal.
39	<b>Node ID File</b>	Check the ID file format.
40	<b>Node Configuration Consistency</b>	When you upgrade a cluster to v1.19 or later, the system checks whether the following configuration files have been modified on the backend:

No.	Check Item	Description
41	<a href="#">Node Configuration File</a>	Check whether the configuration files of key components exist on the node.
42	<a href="#">CoreDNS Configuration Consistency</a>	Check whether the current CoreDNS key configuration Corefile is different from the Helm release record. The difference may be overwritten during the add-on upgrade, <b>affecting domain name resolution in the cluster.</b>

## 2.5.5.2 Node Restrictions

### Check Items

Check the following items:

- Check whether the node is available.
- Check whether the node OS supports the upgrade.
- Check whether the node is marked with unexpected node pool labels.
- Check whether the Kubernetes node name is the same as the ECS name.

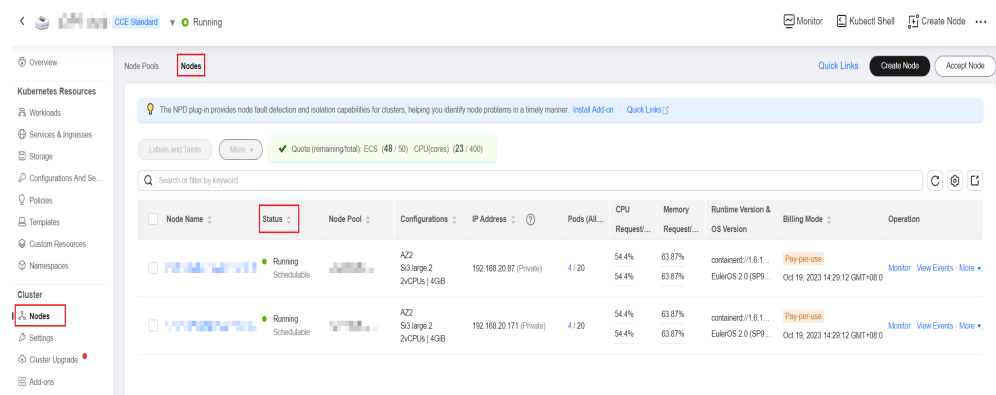
### Solution

#### 1. The node is unavailable. Preferentially recover the node.

If a node is unavailable, log in to the CCE console and click the cluster name to access the cluster console. Then, choose **Nodes** in the navigation pane. In the right pane, click the **Nodes** tab. Ensure that the node is in the **Running** state. A node in the **Installing** or **Deleting** state cannot be upgraded.

If a node is unavailable, recover the node and retry the check task. For details, see [What Should I Do If a Cluster Is Available But Some Nodes Are Unavailable?](#)

Figure 2-13 Checking node statuses



#### 2. The node OS does not support the upgrade.

The following table lists the node OSs that support the upgrade. You can reset the node OS to an available OS in the list.

**Table 2-29** OSs that support the upgrade

OS	Constraint
EulerOS 2.x	If the target version is earlier than v1.27, there are no constraints. If the target version is v1.27 or later, only EulerOS 2.9 and EulerOS 2.10 support the upgrade.
CentOS 7.x	None
Ubuntu	If the target version is v1.27 or later, only Ubuntu 22.04 supports the upgrade.
Huawei Cloud EulerOS	Unlimited

- The affected node belongs to the default node pool but it is configured with a non-default node pool label, which will affect the upgrade.**

If a node is migrated from a common node pool to the default node pool, the **cce.cloud.com/cce-nodepool** label will affect the cluster upgrade. Check whether load scheduling on the node depends on the label.

- If no, delete the label.
- If yes, modify the load balancing policy, remove the dependency, and then delete the label.

- The node is marked with a CNIPProblem taint. Preferentially recover the node.**

The node contains a taint whose key is **node.cloudprovider.kubernetes.io/cni-problem**, and the effect is **NoSchedule**. The taint is added by the NPD add-on. Upgrade the NPD add-on to the latest version and check again. If the problem persists, contact technical support.

- The resources for the affected node in Kubernetes are not available. It is possible that the node is being deleted. Try again later.**

Recheck after the node is deleted.

- The OS running on the master node is EulerOS 2.5, which does not support the cluster to be upgraded to v1.27.5-r0.**

You can upgrade the version to 1.25 or 1.28. If you choose to upgrade the version to 1.28, EulerOS 2.5 will be upgraded to HCE 2.0 during the upgrade. If you have other requirements, submit a service ticket.

### 2.5.5.3 Upgrade Management

#### Check Items

Check whether the target cluster is under upgrade management.

#### Solution

CCE may temporarily restrict the cluster upgrade due to the following reasons:

- The cluster is identified as the core production cluster.
- Other O&M tasks are being or will be performed, for example, 3-AZ reconstruction on master nodes.

To resolve this issue, contact technical support based on logs displayed on the console.

## 2.5.5.4 Add-ons

### Check Items

Check the following items:

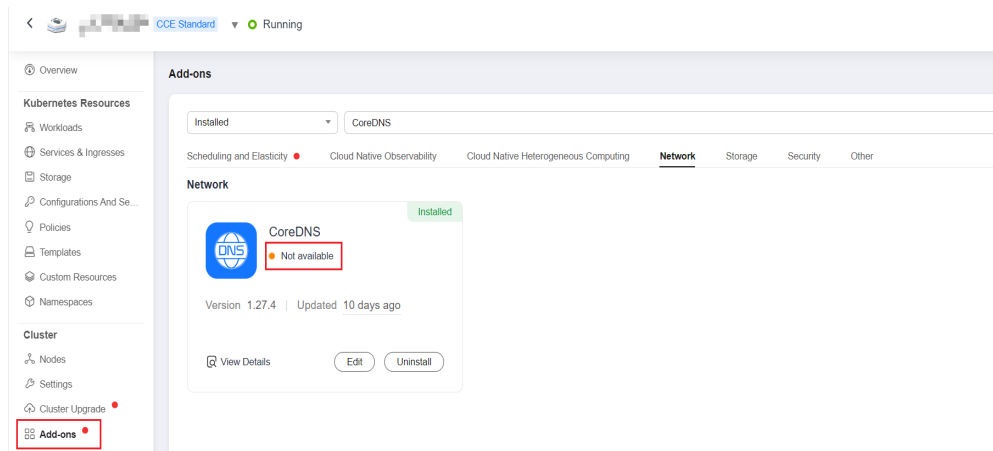
- Check whether the add-on status is normal.
- Check whether the add-on supports the target version.

### Solution

- **Scenario 1: The add-on malfunctions.**

Log in to the CCE console and click the cluster name to access the cluster console. Choose **Add-ons** in the navigation pane and obtain add-ons. Then, handle malfunctional add-ons.

**Figure 2-14** Checking add-on statuses



- **Scenario 2: The target version of the cluster upgrade does not support the add-on.**

The following error information is displayed during the pre-upgrade check:  
addon [\*\*\*] does not support cluster target version, check and try again

Log in to the CCE console and click the cluster name to access the cluster console. Choose **Add-ons** in the navigation pane and manually uninstall the add-on. For details about the supported add-on versions and substitutions, see the [Help](#) document.

- **Scenario 3: The add-on configuration does not meet the upgrade requirements. Upgrade the add-on and try again.**

The following error information is displayed during the pre-upgrade check:  
please upgrade addon [ ] in the page of addon managecheck and try again



In this case, log in to the CCE console and click the cluster name to access the cluster console. Choose **Add-ons** in the navigation pane and manually upgrade the add-on.

### 2.5.5.5 Helm Charts

#### Check Items

Check whether the current HelmRelease record contains discarded Kubernetes APIs that are not supported by the target cluster version. If yes, the Helm chart may be unavailable after the upgrade.

#### Solution

Convert the discarded Kubernetes APIs to APIs that are compatible with both the source and target versions.

#### NOTE

This item has been automatically processed in the upgrade process. You can ignore this item.

### 2.5.5.6 SSH Connectivity of Master Nodes

#### Check Items

Check whether your master nodes can be accessed using SSH.

#### Solution

There is a low probability that the SSH connectivity check fails due to network fluctuations. Perform the pre-upgrade check again.

If the check still fails, submit a service ticket to contact technical support.

### 2.5.5.7 Node Pools

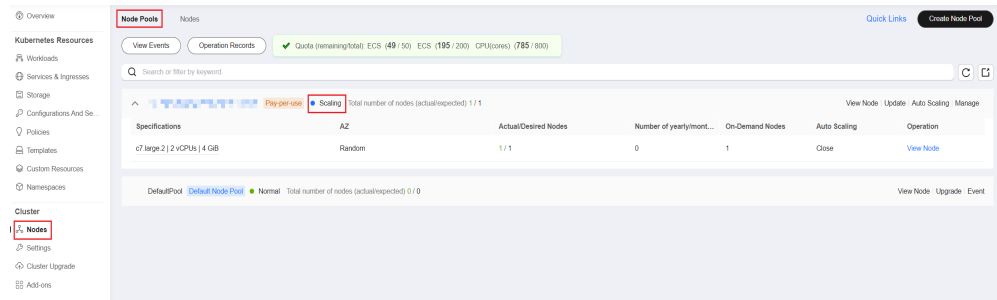
#### Check Items

- Check the node pool status.
- Check whether the node pool OS or container runtime is supported after the upgrade.

#### Solution

- **Scenario: The node pool malfunctions.**  
Log in to the CCE console and click the cluster name to access the cluster console. Choose **Nodes** in the navigation pane and check the status of the affected node pool on the **Node Pools** tab. If the node pool is being scaled, wait until the node pool scaling is complete.

Figure 2-15 Checking node pool statuses



• **Scenario: The node pool OS is not supported.**

The runtime and OS vary depending on the cluster version. This issue typically occurs when a cluster of an earlier version is upgraded to v1.27 or later. For details about the mapping between CCE cluster versions and OSs, see [Node OSs](#).

Log in to the CCE console and click the cluster name to access the cluster console. Choose **Nodes** in the navigation pane, view the status of the affected node pool on the **Node Pools** tab, and click **Upgrade**. Change the supported OSs based on the pre-upgrade check result, and click **OK**.

### 2.5.5.8 Security Groups

#### Check Items

Check whether the **Protocol & Port** of the worker node security groups is set to **ICMP: All** and whether the security group with the source IP address set to the master node security group is deleted.

**NOTE**

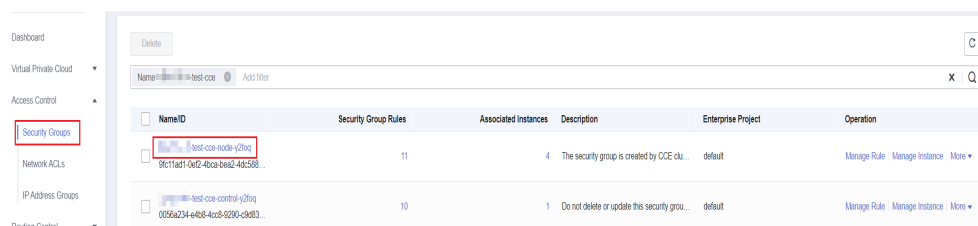
This check item is performed only for clusters using VPC networking. For clusters using other networking, skip this check item.

#### Solution

Log in to the VPC console, choose **Access Control > Security Groups**, and enter the target cluster name in the search box. Two security groups are expected to display:

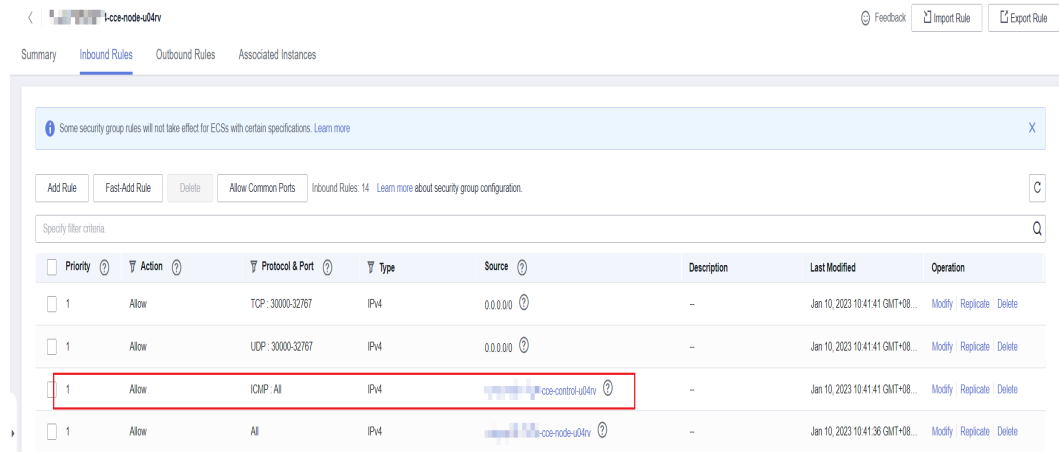
- The security group name is **cluster name-node-xxx**. This security group is associated with the worker nodes.
- The security group name is **cluster name-control-xxx**. This security group is associated with the master nodes.

Figure 2-16 Cluster security groups



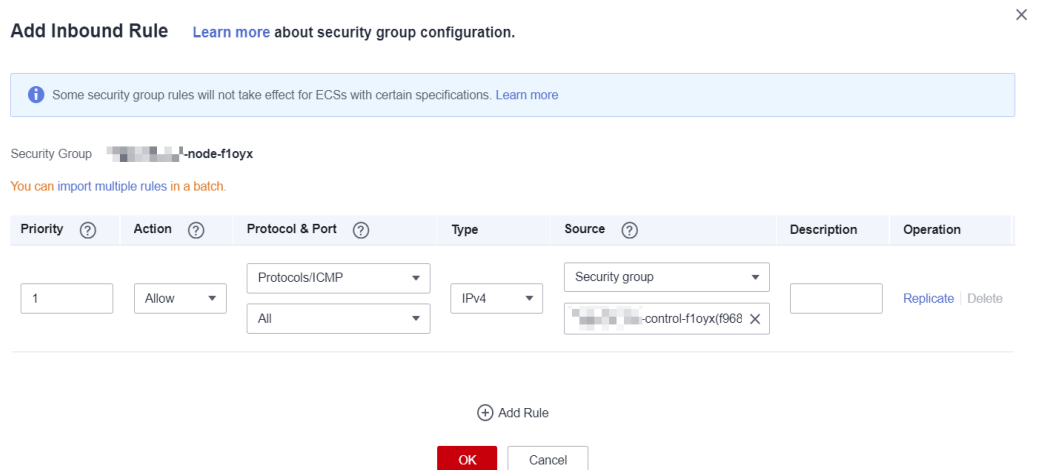
Click the node security group and ensure that the following rules are configured to allow the master node to access the node using **ICMP**.

**Figure 2-17** Node security group rules



If the preceding security group rule is unavailable, add the rule with the following configurations to the node security group: Set **Protocol & Port** to **Protocols/ICMP** and **All**, and **Source** to **Security group** and the master security group. Describe the rule as "Created by CCE, please don't modify! Used by the master node to access the worker node."

**Figure 2-18** Allowing ICMP for the master security group



## 2.5.5.9 Arm Node Restrictions

### Check Items

Check the following items:

- Check whether the cluster contains Arm nodes.

## Solution

- **Scenario 1: The cluster contains Arm nodes.**  
Delete Arm nodes.

### 2.5.5.10 Residual Nodes

## Check Items

Check whether nodes need to be migrated.

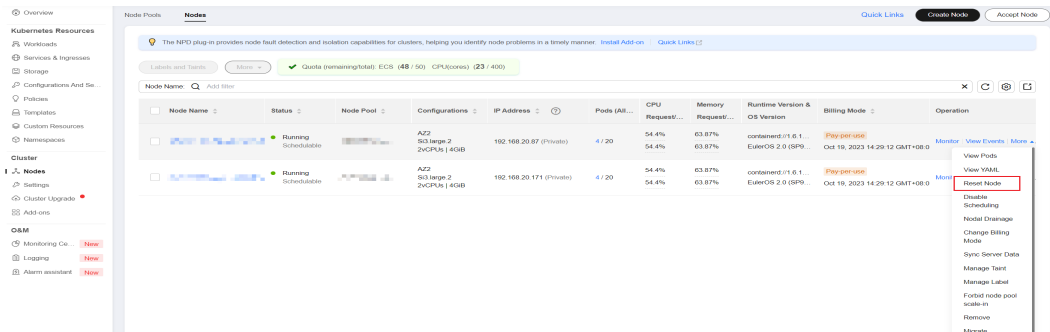
## Solution

This issue is caused by either an error in the node's package pull component or the absence of key system components on the node, which could be due to an upgrade from an earlier version.

### Solution 1

Log in to the CCE console and click the cluster name to access the cluster console. Choose **Nodes** in the navigation pane. Locate the row containing the target node and choose **More > Reset Node** in the **Operation** column. For details, see [Resetting a Node](#). After the node is reset, retry the check task.

**Figure 2-19** Resetting a node



### NOTE

Resetting a node will reset all node labels, which may affect workload scheduling. Before resetting a node, check and retain the labels that you have manually added to the node.

### Solution 2

After creating a node, delete the faulty node.

### 2.5.5.11 Discarded Kubernetes Resources

## Check Items

Check whether there are discarded resources in the clusters.

## Solution

- **Scenario 1: The Service in the clusters of v1.25 or later has discarded annotation tolerate-unready-endpoints.**

Error log:

```
some check failed in cluster upgrade: this cluster has deprecated service list: map[***] with deprecated annotation list [tolerate-unready-endpoints]
```

Check whether the Service provided in the log information contains the annotation **tolerate-unready-endpoint**. If so, replace the annotation with the following fields in Service **spec**:

```
publishNotReadyAddresses: true
```

- **Scenario 2: The Service in the clusters of v1.27 or later has discarded annotation service.kubernetes.io/topology-aware-hints.**

Error log:

```
some check failed in cluster upgrade: this cluster has deprecated service list: map[***] with deprecated annotation list [service.kubernetes.io/topology-aware-hints]
```

Check whether the Service provided in the log information contains the annotation **service.kubernetes.io/topology-aware-hints**. If so, replace it with the annotation **service.kubernetes.io/topology-mode** in the affected Service.

### 2.5.5.12 Compatibility Risks

#### Check Items

Read the version compatibility differences and ensure that they are not affected. The patch upgrade does not involve version compatibility differences.

#### Version compatibility

Upgrade Path	Version Difference	Self-Check
v1.23 or v1.25 Upgraded to v1.27	Docker is no longer recommended. Use containerd instead. For details, see <a href="#">Container Engines</a> .	This item has been included in the pre-upgrade check.

Upgrade Path	Version Difference	Self-Check
<p>v1.21 or v1.19 Upgraded to v1.23</p>	<p>For the NGINX Ingress Controller of an earlier version (community version v0.49 or earlier, or CCE nginx-ingress version v1.x.x), the created ingresses can be managed by the NGINX Ingress Controller even if <b>kubernetes.io/ingress.class: nginx</b> is not set in the ingress <b>annotations</b>. However, for the NGINX Ingress Controller of a later version (community version v1.0.0 or later, or CCE nginx-ingress version v2.x.x), the ingresses created without specifying the Nginx type will not be managed by the NGINX Ingress Controller, and ingress rules will become invalid, which interrupts services.</p>	<p>This item has been included in the pre-upgrade check. You can also perform the self-check by referring to <a href="#">nginx-ingress</a>.</p>
<p>v1.19 to v1.21</p>	<p>The bug of <b>exec probe timeouts</b> is fixed in Kubernetes 1.21. Before this bug is fixed, the exec probe does not consider the <b>timeoutSeconds</b> field. Instead, the probe will run indefinitely, even beyond its configured deadline. It will stop until the result is returned. If this field is not specified, the default value <b>1</b> is used. This field takes effect after the upgrade. If the probe runs over 1 second, the application health check may fail and the application may restart frequently.</p>	<p>Before the upgrade, check whether the timeout is properly set for the exec probe.</p>

Upgrade Path	Version Difference	Self-Check
	<p>kube-apiserver of CCE v1.19 or later requires that the Subject Alternative Names (SANs) field be configured for the certificate of your webhook server. Otherwise, kube-apiserver fails to call the webhook server after the upgrade, and containers cannot be started properly.</p> <p>Root cause: X.509 <b>CommonName</b> is discarded in Go v1.15. kube-apiserver of CCE v1.19 is compiled using Go v1.15. If your webhook certificate does not have SANs, kube-apiserver does not process the <b>CommonName</b> field of the X.509 certificate as the host name by default. As a result, the authentication fails.</p>	<p>Before the upgrade, check whether the SAN field is configured in the certificate of your webhook server.</p> <ul style="list-style-type: none"> <li>If you do not have your own webhook server, you can skip this check.</li> <li>If the field is not set, use the SAN field to specify the IP address and domain name supported by the certificate.</li> </ul>

**Table 2-30** QoS class changes before and after the upgrade

Init Container (Calculated Based on spec.initContainers)	Service Container (Calculated Based on spec.containers)	Pod (Calculated Based on spec.containers and spec.initContainers)	Impacted or Not
Guaranteed	Besteffort	Burstable	Yes
Guaranteed	Burstable	Burstable	No
Guaranteed	Guaranteed	Guaranteed	No
Besteffort	Besteffort	Besteffort	No
Besteffort	Burstable	Burstable	No
Besteffort	Guaranteed	Burstable	Yes
Burstable	Besteffort	Burstable	Yes
Burstable	Burstable	Burstable	No
Burstable	Guaranteed	Burstable	Yes

### 2.5.5.13 CCE Agent Versions

#### Check Items

Check whether cce-agent on the current node is of the latest version.

#### Solution

- **Scenario 1: The error message "you cce-agent no update, please restart it" is displayed.**

cce-agent does not need to be updated but is not restarted. In this case, log in to the node and manually restart cce-agent.

Solution: Log in to the node and run the following command:

```
systemctl restart cce-agent
```

Perform the pre-upgrade check again.

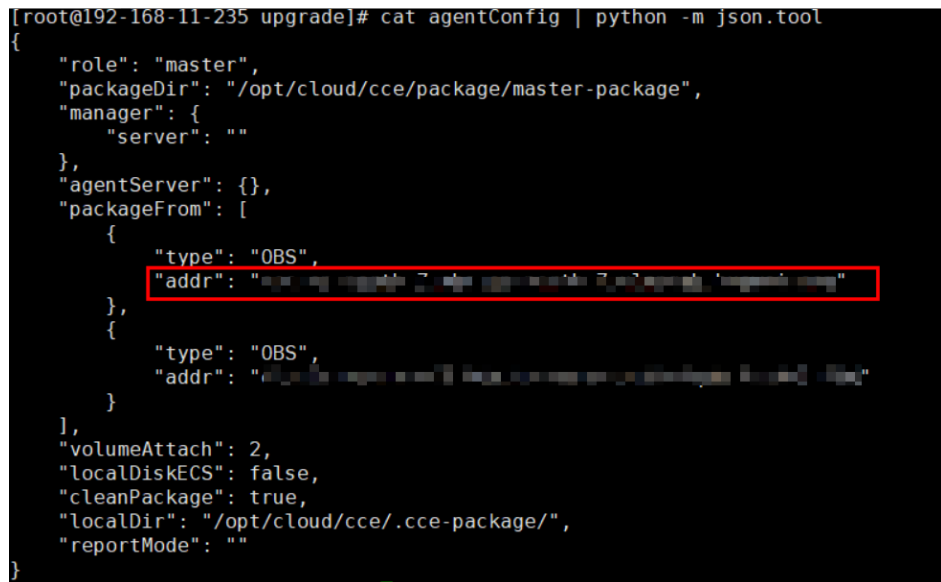
- **Scenario 2: The error message "your cce-agent is not the latest version" is displayed.**

cce-agent is not of the latest version, and the automatic update failed. This issue is typically caused by an invalid OBS path or the component version is outdated.

Solution

- a. Log in to the affected node and run the following command to obtain a valid OBS address:

```
cat /home/paas/upgrade/agentConfig | python -m json.tool
```



```
[root@192-168-11-235 upgrade]# cat agentConfig | python -m json.tool
{
  "role": "master",
  "packageDir": "/opt/cloud/cce/package/master-package",
  "manager": {
    "server": ""
  },
  "agentServer": {},
  "packageFrom": [
    {
      "type": "OBS",
      "addr": "https://obs.cn-north-1.myhuaweicloud.com"
    },
    {
      "type": "OBS",
      "addr": "https://obs.cn-north-1.myhuaweicloud.com"
    }
  ],
  "volumeAttach": 2,
  "localDiskECS": false,
  "cleanPackage": true,
  "localDir": "/opt/cloud/cce/.cce-package/",
  "reportMode": ""
}
```

- b. Log in to a where the check failed, obtain the OBS address again by referring to the previous step, and check whether the OBS addresses are the same. If they are different, change the OBS address of the abnormal node to the correct address.

- c. Run the following commands to download the latest binary file:

- x86  

```
curl -k "https://{OBS address you have obtained}/cluster-versions/base/cce-agent" > /tmp/cce-agent
```



- **Arm**  

```
curl -k "https://{OBS address you have obtained}/cluster-versions/base/cce-agent-arm"  
> /tmp/cce-agent-arm
```
  - d. Replace the original cce-agent binary file.
    - **x86**  

```
mv -f /tmp/cce-agent /usr/local/bin/cce-agent  
chmod 750 /usr/local/bin/cce-agent  
chown root:root /usr/local/bin/cce-agent
```
    - **Arm**  

```
mv -f /tmp/cce-agent-arm /usr/local/bin/cce-agent-arm  
chmod 750 /usr/local/bin/cce-agent-arm  
chown root:root /usr/local/bin/cce-agent-arm
```
  - e. Restart cce-agent.  

```
systemctl restart cce-agent
```
- If you have any questions about the preceding operations, contact technical support.

### 2.5.5.14 Node CPU Usage

#### Check Items

Check whether the node's CPU usage is above 90%.

#### Solution

- **Upgrade the cluster during off-peak hours.**
- Check whether too many pods are deployed on the node. If yes, reschedule pods to other idle nodes.

### 2.5.5.15 CRDs

#### Check Items

Check the following items:

- Check whether the key CRD **packageversions.version.cce.io** of the cluster is deleted.
- Check whether the cluster key CRD **network-attachment-definitions.k8s.cni.cncf.io** is deleted.

#### Solution

If check results are abnormal, contact technical support.

### 2.5.5.16 Node Disks

#### Check Items

Check the following items:

- Check whether the key data disks on the node meet the upgrade requirements.

- Check whether the **/tmp** directory has 500 MB available space.

## Solution

During the node upgrade, the key disks store the upgrade component package, and the **/tmp** directory stores temporary files.

- **Scenario 1: Master node disks fail to meet the upgrade requirements.**  
Contact technical support.
- **Scenario 2: Worker node disks fail to meet the upgrade requirements.**  
Check the usage of each key disk. After ensuring that the available space meets the requirements, check again.
  - Disk partition of Docker: at least 1 GB of available space  
`df -h /var/lib/docker`
  - Disk partition of containerd: at least 1 GB of available space  
`df -h /var/lib/containerd`
  - Disk partition of kubelet: at least 1 GB of available space  
`df -h /mnt/paas/kubernetes/kubelet`
  - System disk: at least 2 GB of available space  
`df -h /`
- **Scenario 3: The available space of the /tmp directory on worker nodes is insufficient.**  
Run the following command to check the usage of the file system where the **/tmp** directory is located. Ensure that the space is greater than 500 MB and check again.  
`df -h /tmp`

### 2.5.5.17 Node DNS

#### Check Items

Check the following items:

- Check whether the DNS configuration of the current node can resolve the OBS address.
- Check whether the current node can access the OBS address of the storage upgrade component package.

#### Solution

During the node upgrade, obtain the upgrade component package from OBS. If this check fails, contact technical support.

### 2.5.5.18 Node Key Directory File Permissions

#### Check Items

Check whether the owner and owner group of the files in the **/var/paas** directory used by the CCE are both **paas**.

## Solution

- **Scenario 1: The error message "xx file permission has been changed!" is displayed.**

Solution: Enable CCE to use the `/var/paas` directory to manage nodes and store file data whose owner and owner group are both `paas`.

During the current cluster upgrade, the owner and owner group of the files in the `/var/paas` directory are reset to `paas`.

Check whether file data in the current service pod is stored in the `/var/paas` directory. If yes, do not use this directory, remove abnormal files from this directory, and check again. After the check is passed, proceed with the upgrade.

```
find /var/paas -not \( -user paas -o -user root \) -print
```

- **Scenario 2: The error message "user paas must have at least read and execute permissions on the root directory" is displayed.**

Solution: Change the permission on the root directory to the default permission 555. If the permission on the root directory of the node is modified, user `paas` does not have the read permission on the root directory. As a result, restarting the component failed during the upgrade.

### 2.5.5.19 kubelet

#### Check Items

Check whether the kubelet on the node is running properly.

#### Solution

- **Scenario 1: The kubelet status is abnormal.**  
If the kubelet malfunctions, the node is unavailable. Restore the node and check again. For details, see [What Should I Do If a Cluster Is Available But Some Nodes Are Unavailable?](#)
- **Scenario 2: The cce-pause version is incorrect.**  
The version of the pause container image on which kubelet depends is not `cce-pause:3.1`. If you continue the upgrade, pods will restart in batches. Currently, the upgrade is not supported. Contact technical support.

### 2.5.5.20 Node Memory

#### Check Items

Check whether the node's memory usage is above 90%.

#### Solution

- **Upgrade the cluster during off-peak hours.**
- Check whether too many pods are deployed on the node. If yes, reschedule pods to other idle nodes.

## 2.5.5.21 Node Clock Synchronization Server

### Check Items

Check whether the clock synchronization server ntpd or chronyd of the node is running properly.

### Solution

- **Scenario 1: ntpd is running abnormally.**

Log in to the node and run the **systemctl status ntpd** command to obtain the running status of ntpd. If the command output is abnormal, run the **systemctl restart ntpd** command and obtain the status again.

The normal command output is as follows:

Figure 2-20 Running status of ntpd

```
[root@paas]# systemctl status ntpd
● ntpd.service - Network Time Service
   Loaded: loaded (/usr/lib/systemd/system/ntpd.service; enabled; vendor preset: disabled)
   Active: active (running) since Tue 2022-12-06 14:52:30 CST; 4 days ago
     Main PID: 8587 (ntpd)
        Tasks: 2
       Memory: 1.6M
      CGroup: /system.slice/ntpd.service
             └─8587 /usr/sbin/ntpd -u ntp:ntp -g -x
```

If the problem persists after ntpd is restarted, contact technical support.

- **Scenario 2: chronyd is running abnormally.**

Log in to the node and run the **systemctl status chronyd** command to obtain the running status of chronyd. If the command output is abnormal, run the **systemctl restart chronyd** command and obtain the status again.

The normal command output is as follows:

Figure 2-21 Running status of chronyd

```
root@paas]# systemctl status chronyd
● chrony.service - chrony, an NTP client/server
   Loaded: loaded (/lib/systemd/system/chrony.service; enabled; vendor preset: enabled)
   Active: active (running) since Wed 2022-08-24 16:33:28 CST; 3 months 16 days ago
     Docs: man:chronyd(8)
           man:chronyc(1)
           man:chrony.conf(5)
   Process: 6492 ExecStartPost=/usr/lib/chrony/chrony-helper update-daemon (code=exited, status=0/SUCCESS)
   Process: 6461 ExecStart=/usr/lib/systemd/scripts/chronyd-starter.sh $DAEMON_OPTS (code=exited, status=0/SUCCESS)
     Main PID: 6488 (chronyd)
        Tasks: 1 (limit: 4915)
      CGroup: /system.slice/chrony.service
             └─6488 /usr/sbin/chronyd
```

If the problem persists after chronyd is restarted, contact technical support.

## 2.5.5.22 Node OS

### Check Items

Check whether the OS kernel version of the node is supported by CCE.

## Solution

- **Case 1: The node image is not a standard CCE image.**

CCE nodes run depending on the initial standard kernel version specified when they are created. CCE has performed comprehensive compatibility tests based on this kernel version. A non-standard kernel version may lead to unexpected compatibility issues during a node upgrade and the upgrade may fail. For details, see [High-Risk Operations and Solutions](#).

Do not directly upgrade this type of nodes. Instead, [reset the nodes](#) to a standard kernel version and then upgrade the nodes.
- **Case 2: An image of a special version is defective.**

A EulerOS release 2.8 (Arm) image of v1.17 is used in the source version. Such an image is defective because the **docker exec** command will be affected after Docker is restarted. When the cluster version is upgraded, the Docker version will be updated and Docker will be restarted. To resolve this issue, do as follows:

  - a. Empty and isolate the affected nodes before upgrading the cluster.
  - b. Upgrade the version to v1.19 or later and reset the nodes to replace the image with one of a later version, for example, EulerOS release 2.9.

### 2.5.5.23 Node CPU Cores

#### Check Items

Check and make sure that the master nodes in your cluster have more than 2 CPU cores.

#### Solution

The number of CPU cores on the master nodes is 2, which may lead to a cluster upgrade failure.

Contact technical support to expand the number of CPU cores to four or more.

### 2.5.5.24 Node Python Commands

#### Check Items

Check whether the Python commands are available on a node.

#### Check Method

```
/usr/bin/python --version  
echo $?
```

If the command output is not 0, the check fails.

#### Solution

Reset the node or manually install Python before attempting the upgrade again.

## 2.5.5.25 ASM Version

### Check Items

Check the following items:

- Check whether ASM is used by the cluster.
- Check whether the current ASM version supports the target cluster version.

### Solution

- Upgrade ASM and then upgrade the cluster. The adaptation rules between ASM and cluster versions are as follows:

**Table 2-31** Adaptation rules between ASM and cluster versions

ASM Version	Cluster Version
1.3	v1.13, v1.15, v1.17, or v1.19
1.6	v1.15 or v1.17
1.8	v1.15, v1.17, v1.19, or v1.21
1.13	v1.21 or v1.23
1.15	v1.21, v1.23, or v1.25

- If ASM is not required, delete it before the upgrade. After the upgrade, the cluster cannot be bound to ASM that does not match the table. For example, a cluster of v1.21 and ASM of v1.8 are used. If you want to upgrade the cluster to v1.25, upgrade ASM to v1.15 first.

## 2.5.5.26 Node Readiness

### Check Items

Check whether the nodes in the cluster are ready.

### Solution

- **Scenario 1: The nodes are in the unavailable status.**  
Log in to the CCE console and click the cluster name to access the cluster console. Choose **Nodes** in the navigation pane and filter out unavailable nodes, rectify the faulty nodes by referring to the suggestions provided by the console, and check again.
- **Scenario 2: The displayed node status is inconsistent with the actual status.**  
The possible causes are as follows:
  - a. The node status is normal on the nodes page, but the check result shows that the node is not ready. Check again.

- b. The node is not found on the nodes page, but the check result shows that the node is in the cluster. Contact technical support.

### 2.5.5.27 Node journald

#### Check Items

Check whether journald of a node is normal.

#### Solution

Log in to the node and run the **systemctl is-active systemd-journald** command to obtain the running status of journald. If the command output is abnormal, run the **systemctl restart systemd-journald** command and obtain the status again.

The normal command output is as follows:

**Figure 2-22** Running status of journald

```
[root@xxxxxxxxxxxxx paas]# systemctl is-active systemd-journald
active
```

If the problem persists after journald is restarted, contact technical support.

### 2.5.5.28 containerd.sock

#### Check Items

Check whether the **containerd.sock** file is on the node. This file affects the startup of container runtime in the Euler OS.

#### Solution

**Scenario: The Docker used by the node is the customized Euler-docker.**

- Step 1** Log in to the node.
- Step 2** Run the **rpm -qa | grep docker | grep euleros** command. If the command output is not empty, the Docker used on the node is Euler-docker.
- Step 3** Run the **stat /run/containerd/containerd.sock** command. If the file exists, Docker startup will fail.
- Step 4** Run the **rm -rf /run/containerd/containerd.sock** command and perform the cluster upgrade check again.

----End

### 2.5.5.29 Internal Error

#### Check Items

This check item is not typical and implies that an internal error was found during the pre-upgrade check.

## Solution

Perform the pre-upgrade check again.

If it fails again, submit a service ticket to contact technical support.

### 2.5.5.30 Node Mount Points

#### Check Items

Check whether there are inaccessible mount points on the node.

#### Solution

**Scenario: There are inaccessible mount points on the node.**

If NFS (such as obsfs or SFS) is used by the node and the node is disconnected from the NFS server, the mount point would be inaccessible and all processes that access this mount point are in D state.

**Step 1** Log in to the node.

**Step 2** Create a script file, for example, `/tmp/check_hang_mount.sh` on the node. The content of the script file is as follows:

```
for mount_path in `cat /proc/self/mountinfo | awk '{print $5}' | grep -v netns`
do
    timeout 10 sh -c "cd $mount_path"
    if [ $? == 124 ];then
        echo "$mount_path hang mount"
    fi
done
```

**Step 3** Run the saved script and check the output.

```
[root@test-02-upgrade-v115-v119-vpc-06491 ~]# bash test.sh
/root/foo hang mount
/root/bar hang mount
[root@test-02-upgrade-v115-v119-vpc-06491 ~]#
```

The mount points of the `/root/foo` and `/root/bar` folders are incorrect.

**Step 4** Run the following command to check the suspended mount points:

```
mount -n | grep /root/foo
```

```
/root/foo hang mount
[root@test-02-upgrade-v115-v119-vpc-06491 ~]# mount -n | grep /root/foo
localhost:/tmp/nfs on /root/foo type nfs (rw,relatime,vers=3,rsize=524288,wsize=524288,namlen=255,hard,proto=tcp,timeo=600,retrans=2,sec=sys,mountaddr=127.0.0.1,mountvers=3,mountport=20040,mountproto=udp,local_lock=none,addr=127.0.0.1)
[root@test-02-upgrade-v115-v119-vpc-06491 ~]#
```

When a mount point is suspended, it typically indicates that services are not using it anymore. After confirming that the mount point is no longer required, run the following command to unmount it and then re-execute the previously mentioned script:

```
umount -l -f localhost:/tmp/nfs
```

```
[root@test-02-upgrade-v115-v119-vpc-06491 ~]# umount -l -f localhost:/tmp/nfs
[root@test-02-upgrade-v115-v119-vpc-06491 ~]# bash test.sh
[root@test-02-upgrade-v115-v119-vpc-06491 ~]#
[root@test-02-upgrade-v115-v119-vpc-06491 ~]#
[root@test-02-upgrade-v115-v119-vpc-06491 ~]#
```



After the execution, perform the check again on the pre-upgrade check page.

----End

### 2.5.5.31 Kubernetes Node Taints

#### Check Items

Check whether the taint needed for cluster upgrade exists on the node.

**Table 2-32** Taint checklist

Taint Name	Impact
node.kubernetes.io/upgrade	NoSchedule

#### Solution

Scenario 1: The node is skipped during the cluster upgrade.

- Step 1** Configure the `kubectl` command. For details, see [Connecting to a Cluster Using kubectl](#).
- Step 2** Check the kubelet version of the corresponding node. The following information is expected:

**Figure 2-23** kubelet version

```
[root@10-3-120-59 paas]# kubectl get node
NAME                STATUS    ROLES    AGE    VERSION
10.3.5.100          Ready    <none>   28h    v1.19.16-r4-CCE22.11.1
10.3.5.101          Ready    <none>   28h    v1.19.16-r4-CCE22.11.1
```

If the version of the node is different from that of other nodes, the node is skipped during the upgrade. Reset the node and upgrade the cluster again. For details about how to reset a node, see [Resetting a Node](#).

**NOTE**

Resetting a node will reset all node labels, which may affect workload scheduling. Before resetting a node, check and retain the labels that you have manually added to the node.

----End

### 2.5.5.32 Everest Restrictions

#### Check Items

Check whether there are any compatibility restrictions on the current Everest add-on.

**Table 2-33** List of Everest add-on versions with compatibility restrictions

Add-on Name	Versions Involved
everest	v1.0.2-v1.0.7 v1.1.1-v1.1.5

## Solution

There are compatibility restrictions on the current Everest add-on and it cannot be upgraded with the cluster upgrade. Contact technical support.

### 2.5.5.33 cce-hpa-controller Limitations

## Check Items

Check whether there are compatibility limitations between the current and target cce-controller-hpa add-on versions.

## Solution

There are compatibility limitations between the current and target versions of the cce-controller-hpa add-on. To address this, install an add-on that provides metrics APIs, like metrics-server, in the cluster.

Install the metrics add-on in the cluster and try again.

### 2.5.5.34 Enhanced CPU Policies

## Check Items

Check whether the current cluster version and the target version support [enhanced CPU policy](#).

## Solution

**Scenario:** Only the current cluster version supports the enhanced CPU policy function. The target version does not support the enhanced CPU policy function.

Upgrade to a cluster version that supports the enhanced CPU policy function. The following table lists the cluster versions that support the enhanced CPU policy function.

**Table 2-34** List of cluster versions that support the enhanced CPU policy function

Cluster Version	Enhanced CPU Policy
Clusters of v1.17 or earlier	Not supported
Clusters of v1.19	Not supported

Cluster Version	Enhanced CPU Policy
Clusters of v1.21	Not supported
Clusters of v1.23 or later	Supported

### 2.5.5.35 Health of Worker Node Components

#### Check Items

Check whether the container runtime and network components on the worker nodes are healthy.

#### Solution

- Issue 1: CNI Agent is not active.  
If your cluster version is earlier than v1.17.17, or you are using the `underlay_ipvlan` network model and your cluster version is later than v1.17.17, log in to the node and run the **`systemctl status canal`** command to check the status of canal. If you encounter an error, run the **`systemctl restart canal`** command and check the status again.  
If you are using the VPC or Cloud Native 2.0 network model and your cluster version is later than v1.17.17, log in to the node and run the **`systemctl status yangtse`** command to check the status of Yangtse. If you encounter an error, run the **`systemctl restart yangtse`** command and check the status again.
- Issue 2: kube-proxy is not active.  
Log in to the node and run the **`systemctl is-active kube-proxy`** command to check the status of kube-proxy. If you encounter an error, run the **`systemctl restart kube-proxy`** command and check the status again.  
If the fault persists, reset the node. For details, see [Resetting a Node](#).

### 2.5.5.36 Health of Master Node Components

#### Check Items

Check whether cluster components such as the Kubernetes component, container runtime component, and network component are running properly before the upgrade.

#### Solution

Perform the pre-upgrade check again.

If it fails again, submit a service ticket to contact technical support.

### 2.5.5.37 Memory Resource Limit of Kubernetes Components

#### Check Items

Check whether the resources of Kubernetes components, such as etcd and kube-controller-manager, exceed the upper limit.

#### Solution

- Solution 1: Reduce Kubernetes resources that are needed.
- Solution 2: Modify cluster specifications. For details, see [Changing Cluster Scale](#).

### 2.5.5.38 Discarded Kubernetes APIs

#### Check Items

The system scans the audit logs of the past day to check whether the user calls the deprecated APIs of the target Kubernetes version.

#### NOTE

Due to the limited time range of audit logs, this check item is only an auxiliary method. APIs to be deprecated may have been used in the cluster, but their usage is not included in the audit logs of the past day. Check the API usage carefully.

#### Solution

##### Check Description

Based on the check result, it is detected that your cluster calls a deprecated API of the target cluster version using kubectl or other applications. You can rectify the fault before the upgrade. Otherwise, the API will be intercepted by kube-apiserver after the upgrade. For details about each deprecated API, see [Deprecated APIs](#).

##### Case Study

Ingresses of the extensions/v1beta1 and networking.k8s.io/v1beta1 APIs are deprecated in Kubernetes v1.22. If you upgrade a cluster from v1.19 or v1.21 to v1.23, existing resources are not affected, but the v1beta1 API may be intercepted in the creation and editing scenarios.

For details about the YAML configuration structure changes, see [Using kubectl to Create a LoadBalancer Ingress](#).

### 2.5.5.39 IPv6 Support in CCE Turbo Clusters

#### Check Items

If IPv6 is enabled for a CCE Turbo cluster, check whether the target cluster version supports IPv6.

## Solution

CCE Turbo clusters support IPv6 since v1.23. This feature is available in the following versions:

- v1.23: 1.23.8-r0 or later
- v1.25: 1.25.3-r0 or later
- v1.25 or later

If IPv6 has been enabled in the cluster before the upgrade, the target cluster version must also support IPv6. Select a proper cluster version.

### 2.5.5.40 NetworkManager

#### Check Items

Check whether NetworkManager of a node is normal.

#### Solution

Log in to the node and run the **systemctl is-active NetworkManager** command to obtain the running status of NetworkManager. If the command output is abnormal, run the **systemctl restart NetworkManager** command and obtain the status again.

If the fault persists, reset the node. For details, see [Resetting a Node](#). Alternatively, contact technical support to restore the file and then perform the upgrade.

### 2.5.5.41 Node ID File

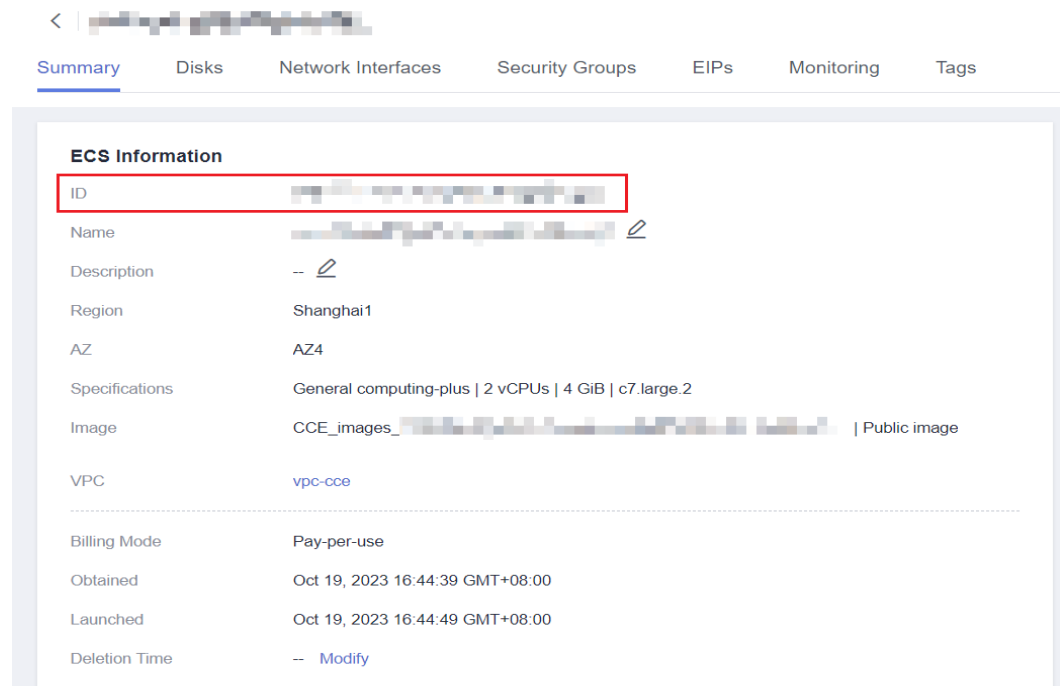
#### Check Items

Check the ID file format.

#### Solution

**Step 1** On the **Nodes** page of the CCE console, click the name of the abnormal node to go to the ECS page.

**Step 2** Copy the node ID and save it to the local host.

**Figure 2-24** Copying a node ID

**Step 3** Log in to the abnormal node and back up files.

```
cp /var/lib/cloud/data/instance-id /tmp/instance-id
cp /var/paas/conf/server.conf /tmp/server.conf
```

**Step 4** Log in to the abnormal node and write the obtained node ID to the file.

```
echo "Node ID" > /var/lib/cloud/data/instance-id
echo "Node ID" > /var/paas/conf/server.conf
```

----End

## 2.5.5.42 Node Configuration Consistency

### Check Items

When you upgrade a cluster to v1.19 or later, the system checks whether the following configuration files have been modified on the backend:

- /opt/cloud/cce/kubernetes/kubelet/kubelet
- /opt/cloud/cce/kubernetes/kubelet/kubelet\_config.yaml
- /opt/cloud/cce/kubernetes/kube-proxy/kube-proxy
- /etc/containerd/default\_runtime\_spec.json
- /etc/sysconfig/docker
- /etc/default/docker
- /etc/docker/daemon.json

If you modify some parameters in these files, the cluster upgrade may fail or services may be abnormal after the upgrade. If you confirm that the modification does not affect services, continue the upgrade.

 NOTE

CCE uses the standard image script to check node configuration consistency. If you use other custom images, the check may fail.

The expected modification will not be intercepted. The following table lists the parameters that can be modified.

**Table 2-35** Parameters that can be modified

Component	Configuration File	Parameter	Upgrade Version
kubelet	/opt/cloud/cce/kubernetes/kubelet/kubelet_config.yaml	cpuManagerPolicy	Later than v1.19
kubelet	/opt/cloud/cce/kubernetes/kubelet/kubelet_config.yaml	maxPods	Later than v1.19
kubelet	/opt/cloud/cce/kubernetes/kubelet/kubelet_config.yaml	kubeAPIQPS	Later than v1.19
kubelet	/opt/cloud/cce/kubernetes/kubelet/kubelet_config.yaml	kubeAPIBurst	Later than v1.19
kubelet	/opt/cloud/cce/kubernetes/kubelet/kubelet_config.yaml	podPidsLimit	Later than v1.19
kubelet	/opt/cloud/cce/kubernetes/kubelet/kubelet_config.yaml	topologyManager-Policy	Later than v1.19
kubelet	/opt/cloud/cce/kubernetes/kubelet/kubelet_config.yaml	resolvConf	Later than v1.19
kubelet	/opt/cloud/cce/kubernetes/kubelet/kubelet_config.yaml	eventRecordQPS	Later than v1.21
kubelet	/opt/cloud/cce/kubernetes/kubelet/kubelet_config.yaml	topologyManager-Scope	Later than v1.21
kubelet	/opt/cloud/cce/kubernetes/kubelet/kubelet_config.yaml	allowedUnsafeSysctls	Later than v1.19
Docker	/etc/docker/daemon.json	dm.basesize	Later than v1.19

## Solution

If you modify some parameters in these files, exceptions may occur after the upgrade. If you are not sure whether the modified parameters will affect the upgrade, contact technical support.

### 2.5.5.43 Node Configuration File

#### Check Items

Check whether the configuration files of key components exist on the node.

The following table lists the files to be checked.

File Name	File Content	Remarks
/opt/cloud/cce/kubernetes/kubelet/kubelet	kubelet command line startup parameters	None
/opt/cloud/cce/kubernetes/kubelet/kubelet_config.yaml	kubelet startup parameters	None
/opt/cloud/cce/kubernetes/kube-proxy/kube-proxy	kube-proxy command line startup parameters	None
/etc/sysconfig/docker	Docker configuration file	Not checked when containerd or the Debain-Group machine is used.
/etc/default/docker	Docker configuration file	Not checked when containerd or the Centos-Group machine is used.

## Solution

Reset the node. For details, see [Resetting a Node](#). Alternatively, contact technical support to restore the file and then perform the upgrade.

### 2.5.5.44 CoreDNS Configuration Consistency

#### Check Items

Check whether the current CoreDNS key configuration Corefile is different from the Helm release record. The difference may be overwritten during the add-on upgrade, **affecting domain name resolution in the cluster**.



## Solution

You can upgrade CoreDNS separately after confirming the configuration differences.

**Step 1** Configure the `kubecttl` command. For details, see [Connecting to a Cluster Using kubecttl](#).

**Step 2** Obtain the Corefile that takes effect currently.

```
kubecttl get cm -nkube-system coredns -o jsonpath='{.data.Corefile}' > corefile_now.txt
cat corefile_now.txt
```

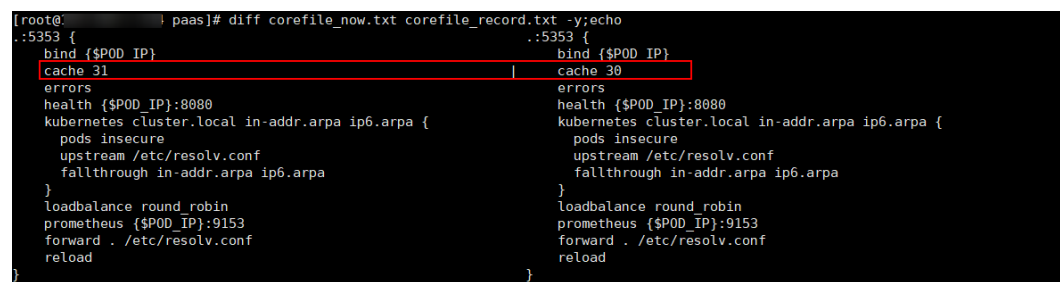
**Step 3** Obtain the Corefile in the Helm Release records.

```
latest_release=`kubecttl get secret -nkube-system -l owner=helm -l name=cceaddon-coredns --sort-by=.metadata.creationTimestamp | awk 'END{print $1}'`
kubecttl get secret -nkube-system $latest_release -o jsonpath='{.data.release}' | base64 -d | base64 -d | gzip -d | python -m json.tool | python -c "
from __future__ import print_function
import json,sys,re,yaml;
manifests = json.load(sys.stdin)['manifest']
files = re.split('(?:^|\\s*\\n)---\\s*',manifests)
for file in files:
    if 'coredns/templates/configmap.yaml' in file and 'Corefile' in file:
        corefile = yaml.safe_load(file)['data']['Corefile']
        print(corefile,end="")
        exit(0);
print('error')
exit(1);
" > corefile_record.txt
cat corefile_record.txt
```

**Step 4** Compare the output differences between [Step 2](#) and [Step 3](#).

```
diff corefile_now.txt corefile_record.txt -y;
```

**Figure 2-25** Viewing output differences



**Step 5** Return to the CCE console and click the cluster name to access the cluster console. Choose **Add-ons** in the navigation pane, select CoreDNS, and click **Upgrade**.

To retain custom configurations, use either of the following methods:

- (Recommended) Set `parameterSyncStrategy` to **inherit**. In this case, custom settings are automatically inherited. The system automatically parses, identifies, and inherits custom parameters.
- Set `parameterSyncStrategy` to **force**. Manually enter the differential configuration. For details, see [CoreDNS](#).

**Step 6** Click **OK**. After the add-on upgrade is complete, check whether all CoreDNS instances are available and whether Corefile meets the expectation.

```
kubecttl get cm -nkube-system coredns -o jsonpath='{.data.Corefile}'
```

- Step 7** Change the value of **parameterSyncStrategy** to **ensureConsistent** to enable configuration consistency verification.

In addition, it is a good practice to use the parameter configuration function of CCE add-ons to modify the Corefile configuration for consistency.

----End

## 2.5.5.45 sudo

### Check Items

Check whether the sudo commands and sudo-related files of the node are working.

### Solution

- Scenario 1: The sudo command fails to be executed.  
During the in-place cluster upgrade, the sudo command must be available. Log in to the node and run the following command to check whether the sudo command is available:  

```
sudo echo hello
```

  
If the sudo command is unavailable, copy this command from another node to the target node.
- Scenario 2: Key files cannot be modified.  
During the in-place cluster upgrade, the **/etc/sudoers** and **/etc/sudoers.d/sudoerspaas** files are modified to obtain the sudo permission and update the components (such as Docker and kubelet) whose owner and owner group are **root** and related configuration files on the node. Log in to the node and run the following command to check whether the file can be modified:  

```
lsattr -l /etc/sudoers.d/sudoerspaas /etc/sudoers
```

  
If **immutable** is displayed in the command output, the file is locked by the **i** lock and cannot be modified. You are advised to remove the **i** lock.  

```
chattr -i /etc/sudoers.d/sudoerspaas /etc/sudoers
```

## 2.5.5.46 Key Node Commands

### Check Items

Whether some key commands that the node upgrade depends on are working

### Solution

- Scenario 1: Executing the package manager command failed.  
Executing the **rpm** or **dpkg** command failed. In this case, log in to the affected node and check whether the following commands are available:  

```
rpm -qa
```

  
If the preceding command is unavailable, run the following command:  

```
rpm --rebuilddb
```
- Scenario 2: The **systemctl status** command fails to be executed.

If the **systemctl status** command on a node is unavailable, many check items will be affected. Log in to the node and check the availability of the following commands:

```
systemctl status kubelet
```

If the fault persists, reset the node. For details, see [Resetting a Node](#).

## 2.5.5.47 Mounting of a Sock File on a Node

### Check Items

Check whether the **docker/containerd.sock** file is directly mounted to the pods on a node. During an upgrade, Docker or containerd restarts and the sock file on the host changes, but the sock file mounted to pods does not change accordingly. As a result, your services cannot access Docker or containerd due to sock file inconsistency. After the pods are rebuilt, the sock file is mounted to the pods again, and the issue is resolved accordingly.

Kubernetes cluster users typically use sock files in the following scenarios:

1. Monitoring applications deployed as DaemonSets use a sock file to access Docker or containerd to obtain pod statuses on a node.
2. Compilation platform applications use a sock file to access Docker or containerd to obtain containers for compiling programs.

### Solution

- Scenario 1: This issue occurred on an application, and operations need to be taken to resolve this issue.

Mount the sock file by mounting a directory. For example, if the sock file is stored in **/var/run/docker.sock** on the host, perform the following operations to resolve this issue (the following modifications will lead to the rebuilding of pods):

```
kind: Deployment
apiVersion: apps/v1
metadata:
  name: test
spec:
  replicas: 1
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      app: nginx
    spec:
      containers:
        - name: container-1
          image: 'nginx'
          imagePullPolicy: IfNotPresent
          volumeMounts:
            - name: sock-dir
              mountPath: /var/run
      imagePullSecrets:
        - name: default-secret
      volumes:
        - name: sock-dir
          hostPath:
            path: /var/run
```

- Scenario 2: This issue occurred on an application, and the risk that sock cannot be accessed for a short time is acceptable.  
Skip this check item and perform the check again. After the cluster is upgraded, delete the existing pods to trigger pod rebuilding. Then, the access to sock will be recovered.
- Scenario 3: This issue occurred on some CCE add-ons of earlier versions.  
Upgrade the CCE add-ons to the latest version. For example, if this issue occurred on the Dolphin add-on of versions earlier than 1.2.2, upgrade the add-on to 1.2.2 or later.
- Scenario 4: The "failed to execute docker ps -aq" error is displayed in the log analysis.  
This error is usually caused by a container engine exception. Submit a service ticket and contact O&M personnel.

## 2.5.5.48 HTTPS Load Balancer Certificate Consistency

### Check Items

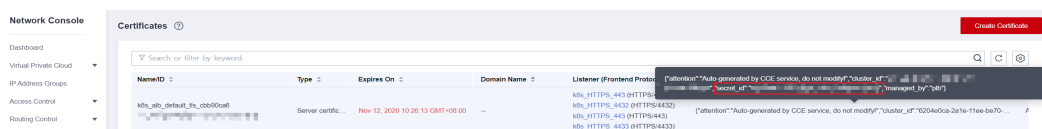
Check whether the certificate used by an HTTPS load balancer has been modified on ELB.

### Solution

The certificate referenced by an HTTPS ingress created on CCE is modified on the ELB console. This leads to inconsistent certificate content in the CCE cluster and that required by the load balancer. After the CCE cluster is upgraded, the load balancer's certificate is overwritten.

- Step 1** Log in to the ELB console, choose **Elastic Load Balance > Certificates**, locate the certificate, and find the **secret\_id** in the certificate description.

Figure 2-26 Viewing a certificate



The **secret\_id** is the **metadata.uid** of the secret in the cluster. Use this UID to obtain the secret name in the cluster.

Run the following kubectl command to obtain the secret name (replace *<secret\_id>* with the actual value):

```
kubectl get secret --all-namespaces -o jsonpath="{range .items[*]}{\"uid:\"}{.metadata.uid}{\" namespace:\"}{.metadata.namespace}{\" name:\"}{.metadata.name}{\"\\n\"}{end}\" | grep <secret_id>
```

- Step 2** Only clusters of v1.19.16-r2, v1.21.5-r0, v1.23.3-r0, and later versions support certificates required by load balancers. For clusters of the earlier versions, see [Solution 1](#). For clusters of other versions, see [Solution 2](#).
- Solution 1: Replace the certificate used by an ingress with the one used by the load balancer. Then, you can create or edit the certificate on the ELB console.
    - a. Log in to the CCE console and click the cluster name to access the cluster console. Choose **Services & Ingresses** in the navigation pane, click the

**Ingresses** tab, locate the row containing the ingress that uses the certificate, and choose **More** > **Update** in the **Operation** column. If multiple ingresses are using this certificate, update the certificate for all of these ingresses. To check which ingresses are using a certificate, use the **secretName** parameter in **spec.tls** of the ingress YAML files.

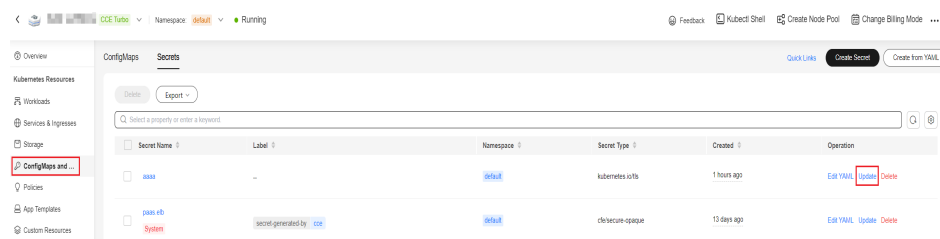
Run the following kubectl command to obtain the ingresses using a certificate (replace `<secret_name>` with the actual value):

```
kubectl get ingress --all-namespaces -o jsonpath='{range .items[*]}{"namespace:"}{.metadata.namespace}{" name:"}{.metadata.name}{" tls:"}{.spec.tls[*]}{"\n"}{end}' | grep <secret_name>
```

- b. When configuring a listener, select **ELB server certificate** for **Certificate Source** and click **OK**. In this way, the certificate can be created or edited on the ELB console.
  - c. On the **ConfigMaps and Secrets** page, delete the target secret. Before the deletion, back up data.
- Solution 2: Overwrite the certificate used by an ingress with the corresponding secret resource of the cluster to prevent the certificate being updated on the ELB console during the cluster upgrade.

Log in to the CCE console and click the cluster name to access the cluster console. Choose **ConfigMaps and Secrets** from the navigation pane, click the **Secrets** tab, locate the row containing the target secret, click **Update** in the **Operation** column, and enter the certificate you are using.

Figure 2-27 Modifying a secret



----End

## 2.5.5.49 Node Mounting

### Check Items

Check whether the default mount directory and soft link on the node have been manually mounted or modified.

- Non-shared disk
  - By default, `/var/lib/docker`, `containerd`, or `/mnt/paas/kubernetes/kubelet` is mounted to CCE nodes. Check whether `/var`, `/var/lib`, `/mnt`, `/mnt/paas`, and `/mnt/paas/kubernetes` have been manually mounted.
  - CCE creates the `/var/lib/kubelet` to `/mnt/paas/kubernetes/kubelet` link by default. Check whether the link has been manually modified.
- Shared disk
  - By default, `/mnt/paas/` is mounted to CCE nodes. Check whether `/mnt` has been manually mounted.

- CCE creates the `/var/lib/kubelet` to `/mnt/paas/kubernetes/kubelet` and `/var/lib/docker` or `/var/lib/containerd` to `/mnt/paas/runtime` soft links by default. Check whether the links have been manually modified.

## Solution

### How Do I Check Whether a Disk Is Shared?

- Step 1** Log in to the target node based on the check information.
- Step 2** Run the `lsblk` command to check whether `vgpaas-share` is mounted to `/mnt/paas`. If yes, a shared disk is used.

Figure 2-28 Checking whether a shared disk is used

```
[root@test-os-upgrade-35777 ~]# lsblk
NAME                MAJ:MIN RM  SIZE RO TYPE MOUNTPOINT
vda                  253:0    0   50G  0 disk
└─vda1                253:1    0   50G  0 part /
vdb                  253:16   0  100G  0 disk
└─vgpaas-share       252:0    0  100G  0 lvm  /mnt/paas
```

----End

### What Can I Do If an Error Occurred in a Node Mounting Check?

1. Cancel the manually modified mount point.
2. Cancel the modification on the default soft links.

## 2.5.5.50 Login Permissions of User `paas` on a Node

### Check Items

Check whether user `paas` is allowed to log in to a node.

### Solution

Run the following command to check whether user `paas` is allowed to log in to a node:

```
sudo grep "paas" /etc/passwd
```

If the permissions assigned to user `paas` contain `nologin` or `false`, the user does not have the login permission. In this case, restore the login permission of user `paas`.

Run the following command to restore the login permission of user `paas`:

```
usermod -s /bin/bash paas
```

## 2.5.5.51 Private IPv4 Addresses of Load Balancers

### Check Items

Check whether the load balancer associated with a Service is allocated with a private IPv4 address.

### Solution

**Solution 1:** Delete the Service that is associated with a load balancer without a private IPv4 address.

**Solution 2:** Bind a private IP address to the load balancer without a private IPv4 address. The procedure is as follows:

**Step 1** Obtain the load balancer associated with the target Service.

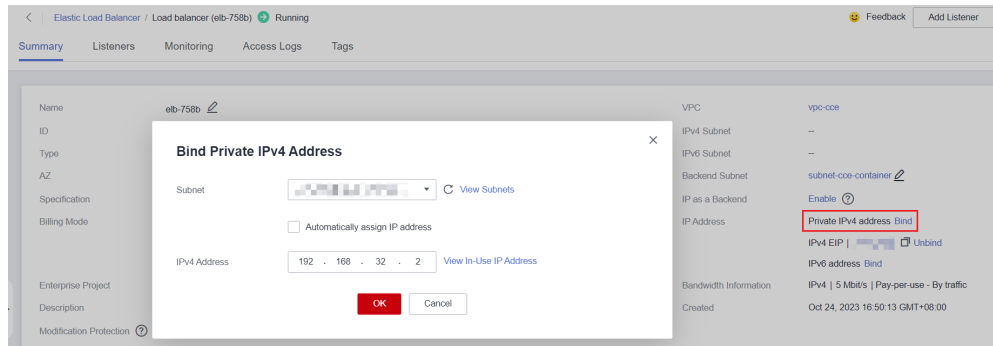
- Method 1: Obtain the load balancer ID based on the pre-upgrade check log. Go to the ELB console and filter load balancers by load balancer ID. elbs (ids: [\*\*\*\*]) without ipv4 private ip, please bind private ip to these elbs and try again
- Method 2: Log in to the CCE console and click the cluster name to access the cluster console. Then, choose **Services & Ingresses** in the navigation pane. In the right pane, click the name of the target load balancer to go to the ELB page.

**Step 2** Check whether the load balancer has a private IPv4 address.

Name/ID	Moni...	Status	Type	Specific...	IP Address and Network
[Redacted]	[Icon]	Running	Dedicated	Network loa...	[Redacted] (IPv4 EIP) vpc-cce (VPC)
[Redacted]	[Icon]	Running	Shared	--	192.168.0.133 (Private IPv4 address) [Redacted] (IPv4 EIP) vpc-cce (VPC)

**Step 3** Bind a private IP address to the load balancer without a private IPv4 address.

1. Log in to the CCE console and click the name of the target load balancer.
2. On the **Summary** tab, click **Bind** next to **Private IPv4 address**.
3. Configure the subnet and IPv4 address, and click **OK**.



----End

## 2.5.5.52 Historical Upgrade Records

### Check Items

Check the historical upgrade records of the cluster and confirm that the current version of the cluster meets the requirements for upgrading to the target version.

### Solution

Upgrading your cluster from an earlier version can be risky and may result in this issue. To avoid this, it is recommended that you migrate the cluster beforehand.

If you still want to proceed with the cluster upgrade, submit a service ticket to contact technical support for evaluation.

## 2.5.5.53 CIDR Block of the Cluster Management Plane

### Check Items

Check whether the CIDR block of the cluster management plane is the same as that configured on the backbone network.

### Solution

The CIDR block of the management plane has been modified in your region. As a result, the CIDR block of the management plane and that of the backbone network are inconsistent.

Submit a service ticket to contact technical support to modify the settings, and recheck the configurations.

## 2.5.5.54 GPU Add-on

### Check Items

The GPU add-on is involved in the upgrade, which may affect the GPU driver installation during the creation of a GPU node.



## Solution

The GPU add-on driver needs to be configured by yourself. Check the compatibility between the GPU add-on and the GPU driver. It is a good practice to verify the upgrade of the GPU driver to the target version in the test environment, configure the current GPU driver, and check whether the created GPU node can run properly.

Perform the following operations to check the upgrade of the GPU driver to the target version and current driver configuration of GPU add-on:

- Step 1** Log in to the CCE console. In the navigation pane, choose **Add-ons**. Then, check the CCE AI Suite (NVIDIA GPU) add-on.
- Step 2** Click **Upgrade** of the add-on and check the target version and driver version of the add-on.
- Step 3** Verify the upgrade of the GPU driver to the target version in the test environment, configure the current GPU driver, and check whether the created GPU node can run properly.

If the GPU add-on and the GPU driver are incompatible, install the driver of a later version. If necessary, contact technical support.

----End

### 2.5.5.55 Nodes' System Parameters

#### Check Items

Check whether the default system parameter settings on your nodes are modified.

#### Solution

If the MTU value of the bond0 network on your BMS node is not the default value 1500, this check item failed.

Non-default parameter settings may lead to service packet loss. Change them back to the default values.

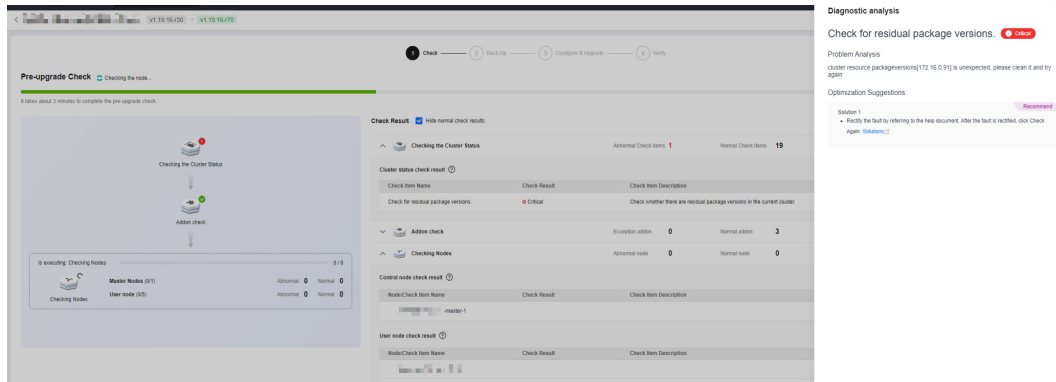
### 2.5.5.56 Residual Package Version Data

#### Check Items

Check whether there are residual package version data in the current cluster.

#### Solution

A message is displayed indicating that there are residual 10.12.1.109 CRD resources in your cluster. This issue occurs because CRD resources are not cleared after nodes in earlier CCE versions are deleted.



Manually perform the following operations to clear the residual resources:

**Step 1** Back up the residual CRD resources. Take CRD resource 10.12.1.109 as an example. Replace it with the resource displayed in the error message.

```
kubectl get packageversion 10.12.1.109 -oyaml > /tmp/packageversion-109.bak
```

**Step 2** Clear the residual CRD resources.

```
kubectl delete packageversion 10.12.1.109
```

**Step 3** Check residual package versions again.

----End

## 2.5.5.57 Node Commands

### Check Items

Check whether the commands required for the upgrade are available on the node.

### Solution

The cluster upgrade failure is typically caused by the lack of key node commands that are required in the cluster upgrade.

Error messages:

```
__error_code#ErrorCommandNotExist#chage command is not exists#__
__error_code#ErrorCommandNotExist#chown command is not exists#__
__error_code#ErrorCommandNotExist#chmod command is not exists#__
__error_code#ErrorCommandNotExist#mkdir command is not exists#__
__error_code#ErrorCommandNotExist#in command is not exists#__
__error_code#ErrorCommandNotExist#touch command is not exists#__
__error_code#ErrorCommandNotExist#pidof command is not exists#__
```

The preceding error messages indicate the lack of node commands such as **chage**, **chown**, and **chmod**. Add these commands and check the node commands again.

## 2.5.5.58 Node Swap

### Check Items

Check whether swap has been enabled on CCE nodes.

## Solution

By default, swap is disabled on CCE nodes. Check the necessity of enabling swap manually and determine the impact of disabling this function.

If there is no impact, run the **swapoff -a** command to disable swap and perform the check again.

### 2.5.5.59 nginx-ingress Upgrade

#### Check Items

- Check item 1: Check whether there is an Nginx Ingress route whose ingress type is not specified (**kubernetes.io/ingress.class: nginx** is not added to **annotations**) in the cluster.
- Check item 2: Check whether the DefaultBackend Service specified by the NGINX Ingress Controller backend is available.

#### Fault Locating

##### For Check Item 1

For Nginx Ingress, check the YAML. If the ingress type is not specified in the YAML file and the ingress is managed by the NGINX Ingress Controller, the ingress is at risk.

##### Step 1 Check the ingress type.

Run the following command:

```
kubectl get ingress <ingress-name> -oyaml | grep -E 'kubernetes.io/ingress.class: | ingressClassName:'
```

- Fault scenario: If the command output is empty, the ingress type is not specified.
- Normal scenario: The command output is not empty, indicating that the ingress type has been specified by **annotations** or **ingressClassName**.

```
[root@192-168-0-31 paas]# kubectl get ingress test -oyaml | grep -E 'kubernetes.io/ingress.class: | ingressClassName:' -B 1
Warning: extensions/v1beta1 Ingress is deprecated in v1.14+, unavailable in v1.22+; use networking.k8s.io/v1 Ingress
annotations:
  kubernetes.io/ingress.class: nginx
spec:
  ingressClassName: nginx
```

##### Step 2 Ensure that the ingress is managed by the Nginx ingress Controller. The LoadBalancer Ingresses are not affected by this issue.

- For clusters of v1.19, confirm this issue using **managedFields**.

```
kubectl get ingress <ingress-name> -oyaml | grep 'manager: nginx-ingress-controller'
```

```
[root@192-168-0-31 paas]# kubectl get ingress test -oyaml | grep 'manager: nginx-ingress-controller'
Warning: extensions/v1beta1 Ingress is deprecated in v1.14+, unavailable in v1.22+; use networking.k8s.io/v1 Ingress
manager: nginx-ingress-controller
```

- For clusters of other versions, check the logs of the NGINX Ingress Controller pod.

```
kubectl logs -nkube-system cceaddon-nginx-ingress-controller-545db6b4f7-bv74t | grep 'updating Ingress status'
```

```
[root@192-168-0-31 paas]# kubectl logs -nkube-system cceaddon-nginx-ingress-controller-545db6b4f7-bv74t | grep 'updating Ingress status'
8 status.go:281] "updating Ingress status" namespace="default" ingress="test" currentValue=[] newValue=[{IP: Hostname: Ports:[]}] {IP: Hostname: Ports:[]}]
```

If the fault persists, contact technical support.

----End

### For Check Item 2

- Step 1** View the DefaultBackend Service in the namespace where the NGINX Ingress Controller is deployed.

```
kubectl get pod cceaddon-nginx-ingress-<controller-name>-controller-*** -n <namespace> -oyaml | grep 'default-backend'
```

In the preceding command, *cceaddon-nginx-ingress-*<controller-name>*-controller-\*\*\** is the controller pod name, *<controller-name>* is the controller name specified during add-on installation, and *<namespace>* is the namespace where the controller is deployed.

Command output:

```
- '--default-backend-service=<namespace>/<backend-svc-name>'
```

In the preceding command, *<backend-svc-name>* is the name of the DefaultBackend Service for the NGINX Ingress Controller.

- Step 2** Check whether the DefaultBackend Service of the NGINX Ingress Controller is available.

```
kubectl get svc <backend-svc-name> -n <namespace>
```

If the Service is unavailable, this check item failed.

----End

## Solution

### For Check Item 1

Add an annotation to the Nginx ingresses as follows:

```
kubectl annotate ingress <ingress-name> kubernetes.io/ingress.class=nginx
```

#### NOTICE

There is no need to add this annotation to LoadBalancer ingresses. **Verify** that these ingresses are managed by NGINX Ingress Controller.

### For Check Item 2

Create the DefaultBackend Service again.

- If a custom DefaultBackend Service has been specified in the default 404 service configuration during add-on installation, create the same Service.
- If the default DefaultBackend Service is used during add-on installation, the re-created YAML example is as follows:

```
apiVersion: v1
kind: Service
metadata:
  name: cceaddon-nginx-ingress-<controller-name>-default-backend # <controller-name> is the
  controller name.
  namespace: kube-system
labels:
  app: nginx-ingress-<controller-name>
  app.kubernetes.io/managed-by: Helm
  chart: nginx-ingress-<version> # <version> is the add-on version.
  component: default-backend
  heritage: Helm
```

```
release: cceaddon-nginx-ingress- <controller-name>
annotations:
  meta.helm.sh/release-name: cceaddon-nginx-ingress- <controller-name>
  meta.helm.sh/release-namespace: kube-system # Namespace where the add-on is installed
spec:
  ports:
    - name: http
      protocol: TCP
      port: 80
      targetPort: http
  selector:
    app: nginx-ingress- <controller-name>
    component: default-backend
    release: cceaddon-nginx-ingress- <controller-name>
  type: ClusterIP
  sessionAffinity: None
  ipFamilies:
    - IPv4
  ipFamilyPolicy: SingleStack
  internalTrafficPolicy: Cluster
```

### 2.5.5.60 ELB Listener Access Control

#### Check Items

Check whether ELB listener access control has been configured for the Services in the current cluster using annotations.

If so, check whether their configurations are correct.

#### Solution

In case of an incorrect configuration, contact O&M personnel.

### 2.5.5.61 Master Node Flavor

#### Check Items

Check whether the flavor of the master nodes in the cluster is the same as the actual flavor of these nodes.

#### Solution

This issue is typically caused by modifications made to the master node. This upgrade may reset the node.

If you are unsure about the impact, submit a service ticket to contact O&M personnel.

### 2.5.5.62 Subnet Quota of Master Nodes

#### Check Items

Check whether the number of available IP addresses in the cluster subnet supports rolling upgrade.

## Solution

Rolling upgrade is not supported if there are not enough IP addresses in the selected cluster subnet.

Move nodes out of the target subnet and check again. If you are unsure about the impact of migration, submit a service ticket to contact O&M personnel.

### 2.5.5.63 Node Runtime

#### Check Items

Check whether an alarm is generated when a cluster is upgraded to v1.27 or later. Do not use Docker in clusters of versions later than 1.27 because CCE is going to stop the support for Docker.

#### Solution

If you plan to create and use Docker nodes in a cluster later than v1.27, ignore this alarm. However, you are advised to switch to containerd as soon as possible for a better user experience and more powerful functions.

### 2.5.5.64 Node Pool Runtime

#### Check Items

Check whether an alarm is generated when a cluster is upgraded to v1.27 or later. Do not use Docker in clusters of versions later than 1.27 because CCE is going to stop the support for Docker.

#### Solution

If you plan to create and use Docker node pools in a cluster later than v1.27, ignore this alarm. However, you are advised to switch to containerd as soon as possible for a better user experience and more powerful functions.

### 2.5.5.65 Number of Node Images

#### Check Items

Check the number of images on your node. If there are more than 1000 images, it takes a long time for Docker to start, affecting the standard Docker output and functions such as Nginx.

#### Solution

Manually delete residual images.

Perform the pre-upgrade check again.

## 2.5.5.66 OpenKruise Compatibility Check

### Check Items

Check whether the OpenKruise add-on is compatible before upgrading a cluster.

### Solution

Since version 1.24, the Kubernetes community no longer supports Dockershim. CCE uses cri-dockerd as an alternative to Dockershim in clusters of v1.25 or later to accommodate users' Docker habits. However, the OpenKruise community does not support cri-dockerd. For details, see [issue](#).

If you install OpenKruise v1.0.3 in a cluster of v1.25 or later, kruise-daemon cannot run on a Docker node. In this case, use a containerd node.

Use either of the following methods to resolve this issue:

- Method 1: Disable OpenKruise kruise-daemon and upgrade the cluster.
- Method 2: Change the node runtime from Docker to containerd. For details, see [Migrating Nodes from Docker to containerd](#).

## 2.5.5.67 Compatibility Check of Secret Encryption

### Check Items

Check whether the target version supports secret encryption. If it does not, clusters that have this feature enabled cannot be upgraded to the target version.

### Solution

Secret encryption is supported in CCE clusters of v1.27 or later versions. The feature is available in the following versions:

- v1.27: v1.27.10-r0 or later
- v1.28: v1.28.8-r0 or later
- v1.29: v1.29.4-r0 or later
- Other clusters of later versions

If secret encryption has been enabled in the cluster before the upgrade, the target cluster version must also support this feature. Select a target version that supports secret encryption for the upgrade.

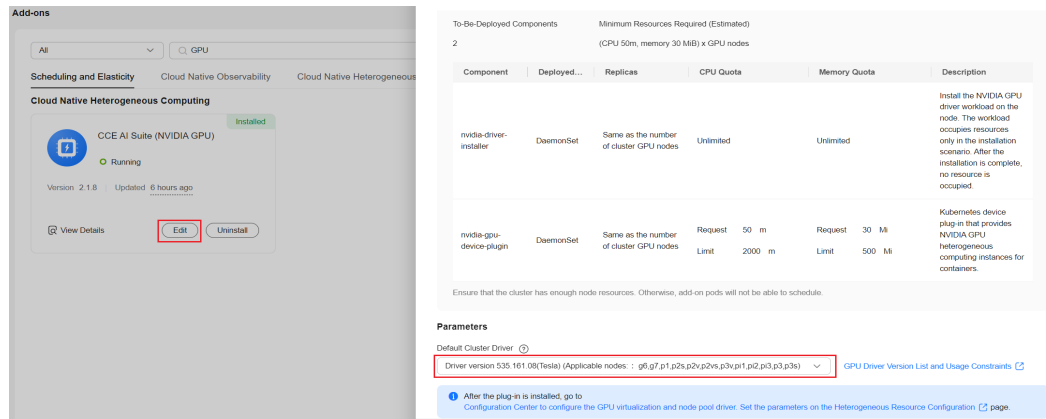
## 2.5.5.68 Compatibility Between the Ubuntu Kernel and GPU Driver

### Check Items

Make sure that the GPU add-on and Ubuntu nodes are compatible before using them in a cluster. If the Ubuntu kernel is 5.15.0-113-generic, the driver of the GPU add-on must be 535.161.08 or later.

## Solution

This issue may occur when you create or reset an Ubuntu node after an upgrade. In this case, upgrade the GPU add-on driver to version 535.161.08 or later, and restart the node.



### 2.5.5.69 Drainage Tasks

## Check Items

An unfinished drainage task is detected in the cluster, which may resume after the upgrade. If this happens, running pods will be evicted, which could impact your services.

## Solution

- Step 1** Configure the `kubectl` command. For details, see [Connecting to a Cluster Using kubectl](#).
- Step 2** Check if there are any drainage tasks in progress.

```
kubectl get drainage
```

**Figure 2-29** An in-progress drainage task

```
[root@10-12 | # kubectl get drainage
NAME                               AGE
10.12 | 1725432592786              4s
10.12 | 1725432506531              90s
```

Delete the drainage resources and perform the pre-upgrade check again.

- Step 3** Run the following command to delete a drainage task:

```
kubectl delete drainage {Name of the drainage task}
```

----End



## 2.5.5.70 Image Layers on a Node

### Check Items

Check the number of image layers on your node. If there are more than 5000 layers, it will take a long time for Docker or containerd to start, affecting the stdout of Docker or containerd.

If Nginx is used in your cluster, data forwarding may be slow.

### Solution

Log in to the node and manually delete unnecessary images.

## 2.5.5.71 Cluster Rolling Upgrade

### Check Items

Check whether your cluster is eligible for a rolling upgrade. The result shows that the rolling upgrade is not supported.

### Solution

Rolling upgrades cannot be performed if the tenant's resource quotas are insufficient.

Contact O&M personnel to expand resources and perform the check again.

## 2.5.5.72 Rotation Certificates

### Check Items

Check whether the number of certificates on your node is greater than 1000. During an upgrade, certificate files will be processed in batches. An excessive number of certificate files will lead to a slow node upgrade and result in pod eviction from the node.

### Solution

Solution 1 (preferred): Reset the node. For details, see [Resetting a Node](#).

Solution 2: Fix the certificate rotation issue on the node.

1. Go to the `/opt/cloud/cce/kubernetes/kubelet/pki/` directory on the node.
2. Back up certificate files `kubelet-server-current.pem` and `kubelet-client-current.pem` on the node.
3. Delete the residual `kubelet-server-*` certificate files from the node.  

```
link_target="$(basename $(readlink kubelet-server-current.pem))" && find -maxdepth 1 -type f -name 'kubelet-server-*.pem' ! -name "$link_target" -delete
```
4. Delete soft links for the certificates.  

```
find -maxdepth 1 -type f -name 'kubelet-server-current.pem' -delete
```

## 2.5.5.73 Ingress and ELB Configuration Consistency

### Check Items

Check whether any modifications have been made to the listener, forwarding policy, forwarding rule, backend cloud server group, backend cloud server, or certificate configurations that were automatically generated by the ingress on the ELB console.

Any modifications made to the ELB will be overwritten during the upgrade process. Verify the configuration consistency before upgrading the cluster.

### Solution

Based on the diagnostic analysis logs, determine which configurations require correction. Typically, this issue occurs when different forwarding policies are configured for the listener associated with the target ingress. This causes a mismatch with the forwarding policies configured for the cluster ingress. To resolve this, either transfer the additional forwarding policies to a different listener or add them to the ingress.

# 3 Nodes

---

## 3.1 Node Overview

### Introduction

A container cluster consists of a set of worker machines, called nodes, that run containerized applications. A node can be a virtual machine (VM) or a physical machine (PM), depending on your service requirements. The components on a node include kubelet, container runtime, and kube-proxy.

#### NOTE

A Kubernetes cluster consists of master nodes and worker nodes. The nodes described in this section refer to **worker nodes**, which are computing nodes of a cluster that run containerized applications.

CCE uses high-performance Elastic Cloud Servers (ECSs) as nodes to build highly available Kubernetes clusters.

### Supported Node Specifications

Different regions support different node flavors, and node flavors may be changed. Log in to the CCE console and check whether the required node flavors are supported on the page for creating nodes.

### Underlying File Storage System of Containers

#### Docker

- In clusters of v1.15.6 or earlier, the underlying Docker file storage system is in XFS format.
- In clusters of v1.15.11 or later, after a node is created or reset, the underlying Docker file storage system changes to the ext4 format.

For containerized applications that use the XFS format, pay attention to the impact of the underlying file storage format change. (The sequence of files in different file systems is different. For example, some Java applications reference a JAR package, but the directory contains multiple versions of the JAR package. If

the version is not specified, the actual referenced package is determined by the system file.)

Run the **docker info | grep "Backing Filesystem"** command to check the format of the underlying Docker storage file used by the current node.

### containerd

Nodes running on containerd use the ext4 file storage system.

## paas User and User Group

When you create a node in a cluster, the paas user or a user group will be created on the node by default. CCE components and CCE add-ons on a node run as a non-root user (user **paas** or a user group) to minimize the running permission. If the paas user or user group is modified, CCE components and pods may fail to run properly.

### NOTICE

The normal running of CCE components depends on the paas user or user group. Pay attention to the following requirements:

- Do not modify the directory permission and container directory permission on a node.
- Do not change the GID and UID of the paas user or user group.
- Do not directly use the paas user or user group to set the user and group to which the service file belongs.

## Node Lifecycle

A lifecycle indicates the node statuses recorded from the time when the node is created through the time when the node is deleted or released.

**Table 3-1** Node statuses

Status	Status Attribute	Description
Running	Stable state	The node is running properly and is connected to the cluster. Nodes in this state can provide services.
Unavailable	Stable state	The node is not functioning correctly, which includes being in a stopped state. Instances in this state cannot provide services.
Creating	Intermediate state	The node has been created but is not running.
Installing	Intermediate state	The Kubernetes software is being installed on the node.

Status	Status Attribute	Description
Upgrading	Intermediate state	The node is being upgraded.
Deleting	Intermediate state	The node is being deleted. If this state stays for a long time, an exception occurred.
Error	Stable state	The node is abnormal. Instances in this state no longer provide services. In this case, perform the operations in <a href="#">Resetting a Node</a> .

## 3.2 Container Engines

### Introduction to Container Engines

Container engines, one of the most important components of Kubernetes, manage the lifecycle of images and containers. The kubelet interacts with a container runtime through the Container Runtime Interface (CRI).

CCE supports containerd and Docker. **containerd is recommended for its shorter traces, fewer components, higher stability, and less consumption of node resources.**

**Table 3-2** Comparison between container engines

Item	containerd	Docker
Tracing	kubelet --> CRI plugin (in the containerd process) --> containerd	<ul style="list-style-type: none"> <li>Docker (Kubernetes v1.23 and earlier versions): kubelet --&gt; dockershim (in the kubelet process) --&gt; docker --&gt; containerd</li> <li>Docker (community solution for Kubernetes v1.24 or later): kubelet --&gt; cri-dockerd (kubelet uses CRI to connect to cri-dockerd) --&gt; docker--&gt; containerd</li> </ul>
Command	crictl/ctr	docker
Kubernetes CRI	Native support	Support through dockershim or cri-dockerd
Pod delayed startup	Minor	High

Item	containerd	Docker
kubelet CPU/ memory usage	Minor	High
Runtime's CPU/memory usage	Minor	High

## Mapping Between Node OSs and Container Engines

### NOTE

VPC network clusters of v1.23 or later versions support containerd. Tunnel network clusters of v1.23.2-r0 or later versions support containerd.

**Table 3-3** Node OSs and container engines in CCE clusters

OS	Kernel Version	Container Engine	Container Storage Rootfs	Container Runtime
CentOS 7.6	3.x	Docker Clusters of v1.23 and later support containerd.	Clusters of v1.19.16 and earlier use Device Mapper. Clusters of v1.19.16 and later use OverlayFS.	runC
EulerOS 2.9	4.x	Docker Clusters of v1.23 and later support containerd.	OverlayFS	runC

## Common Commands of containerd and Docker

containerd does not support Docker APIs and Docker CLI, but you can run crictl commands to implement similar functions.

**Table 3-4** Image-related commands

Operation	Docker Command	containerd Command	
	docker	crictl	ctr
List local images.	docker images	crictl images	ctr -n k8s.io i ls
Pull images.	docker pull	crictl pull	ctr -n k8s.io i pull

Operation	Docker Command	containerd Command	
	docker	crictl	ctr
Push images.	docker push	None	ctr -n k8s.io i push
Delete a local image.	docker rmi	crictl rmi	ctr -n k8s.io i rm
Check images.	docker inspect	crictl inspecti	None

**Table 3-5** Container-related commands

Operation	Docker Command	containerd Command	
	docker	crictl	ctr
List containers.	docker ps	crictl ps	ctr -n k8s.io c ls
Create a container.	docker create	crictl create	ctr -n k8s.io c create
Start a container.	docker start	crictl start	ctr -n k8s.io run
Stop a container.	docker stop	crictl stop	None
Delete a container.	docker rm	crictl rm	ctr -n k8s.io c del
Connect to a container.	docker attach	crictl attach	None
Access the container.	docker exec	crictl exec	None
Query container details.	docker inspect	crictl inspect	ctr -n k8s.io c info
View container logs.	docker logs	crictl logs	None
Check the resource usage of the container.	docker stats	crictl stats	None
Update container resource limits.	docker update	crictl update	None

**Table 3-6** Pod-related commands

Operation	Docker Command	containerd Command	
	docker	crictl	ctr
List pods.	None	crictl pods	None
View pod details.	None	crictl inspectp	None
Start a pod.	None	crictl start	None
Run a pod.	None	crictl runp	None
Stop a pod.	None	crictl stopp	None
Delete a pod.	None	crictl rmp	None

 **NOTE**

Containers created and started by containerd are immediately deleted by kubelet. containerd does not support suspending, resuming, restarting, renaming, and waiting for containers, nor Docker image build, import, export, comparison, push, search, and labeling. containerd does not support file copy. You can log in to the image repository by modifying the configuration file of containerd.

### Differences in Tracing

- Docker (Kubernetes v1.23 and earlier versions):  
kubelet --> dockershim (in the kubelet process) --> docker --> containerd
- Docker (community solution for Kubernetes v1.24 or later):  
kubelet --> cri-dockerd (kubelet uses CRI to connect to cri-dockerd) --> docker--> containerd
- containerd:  
kubelet --> CRI plugin (in the containerd process) --> containerd

Although Docker has added functions such as swarm cluster, docker build, and Docker APIs, it also introduces bugs. Compared with containerd, Docker has one more layer of calling. **Therefore, containerd is more resource-saving and secure.**

### Container Engine Versions

- Docker
  - EulerOS/CentOS: docker-engine 18.9.0, a Docker version customized for CCE. Security vulnerabilities will be fixed promptly.
- containerd: 1.6.14

## 3.3 Node OSs

This section describes the mappings between released cluster versions and OS versions.



## ECS (VM)

Table 3-7 ECS (VM) OS

OS	Cluster Version	CCE Standard Cluster		CCE Turbo Cluster	Latest Kernel
		VPC Network	Tunnel Network	Cloud Native 2.0 Network	
EulerOS release 2.9	v1.29	√	√	√	4.18.0-147.5.1.6.h1152.eulerosv2r9.x86_64
	v1.28	√	√	√	4.18.0-147.5.1.6.h1152.eulerosv2r9.x86_64
	v1.27	√	√	√	4.18.0-147.5.1.6.h1152.eulerosv2r9.x86_64
	v1.25	√	√	√	4.18.0-147.5.1.6.h1152.eulerosv2r9.x86_64
	v1.23	√	√	√	4.18.0-147.5.1.6.h1152.eulerosv2r9.x86_64
	v1.21 (end of maintenance)	√	√	√	4.18.0-147.5.1.6.h1071.eulerosv2r9.x86_64
	v1.19 (end of maintenance)	√	√	√	4.18.0-147.5.1.6.h1071.eulerosv2r9.x86_64
EulerOS release 2.9 (Arm)	v1.29	√	√	√	4.19.90-vhulk2103.1.0.h1144.eulerosv2r9.aarch64
	v1.28	√	√	√	4.19.90-vhulk2103.1.0.h1144.eulerosv2r9.aarch64

OS	Cluster Version	CCE Standard Cluster		CCE Turbo Cluster	Latest Kernel
		VPC Network	Tunnel Network	Cloud Native 2.0 Network	
	v1.27	√	√	√	4.19.90-vhulk2103.1.0.h1144.eulerosv2r9.aarch64
	v1.25	√	√	√	4.19.90-vhulk2103.1.0.h1144.eulerosv2r9.aarch64
	v1.23	√	√	√	4.19.90-vhulk2103.1.0.h1144.eulerosv2r9.aarch64
	v1.21 (end of maintenance)	√	√	√	4.19.90-vhulk2103.1.0.h1060.eulerosv2r9.aarch64
	v1.19 (end of maintenance)	√	√	√	4.19.90-vhulk2103.1.0.h1060.eulerosv2r9.aarch64
CentOS Linux release 7.6	v1.29	√	√	√	3.10.0-1160.108.1.el7.x86_64
	v1.28	√	√	√	3.10.0-1160.108.1.el7.x86_64
	v1.27	√	√	√	3.10.0-1160.108.1.el7.x86_64
	v1.25	√	√	√	3.10.0-1160.108.1.el7.x86_64
	v1.23	√	√	√	3.10.0-1160.108.1.el7.x86_64
	v1.21 (end of maintenance)	√	√	√	3.10.0-1160.92.1.el7.x86_64

OS	Cluster Version	CCE Standard Cluster		CCE Turbo Cluster	Latest Kernel
		VPC Network	Tunnel Network	Cloud Native 2.0 Network	
	v1.19.16 (end of maintenance)	√	√	√	3.10.0-1160.92.1.el7.x86_64
Ubuntu 22.04	v1.29	√	x	√	5.15.0-92-generic
	v1.28	√	x	√	5.15.0-92-generic
	v1.27	√	x	√	5.15.0-92-generic
	v1.25	√	x	√	5.15.0-92-generic
	v1.23	√	x	√	5.15.0-92-generic

## ECS (PM)

Table 3-8 ECS (PM) OS

OS	Cluster Version	CCE Standard Cluster		CCE Turbo Cluster	Latest Kernel
		VPC Network	Tunnel Network	Cloud Native 2.0 Network	
EulerOS release 2.10	v1.29	√	√	√	4.18.0-147.5.2.15.h1109.eulerosv2r10.x86_64
	v1.28	√	√	√	4.18.0-147.5.2.15.h1109.eulerosv2r10.x86_64
	v1.27	√	√	√	4.18.0-147.5.2.15.h1109.eulerosv2r10.x86_64
	v1.25	√	√	√	4.18.0-147.5.2.15.h1109.eulerosv2r10.x86_64

OS	Cluster Version	CCE Standard Cluster		CCE Turbo Cluster	Latest Kernel
		VPC Network	Tunnel Network	Cloud Native 2.0 Network	
	v1.23	√	√	√	4.18.0-147.5.2.15.h1109.eulerosv2r10.x86_64
	v1.21 (end of maintenance)	√	√	√	4.18.0-147.5.2.15.h1109.eulerosv2r10.x86_64
	v1.19.16 (end of maintenance)	√	√	√	4.18.0-147.5.2.15.h1109.eulerosv2r10.x86_64

## BMS

Table 3-9 BMS OS

OS	Cluster Version	CCE Standard Cluster		CCE Turbo Cluster	Latest Kernel
		VPC Network	Tunnel Network	Cloud Native 2.0 Network	
EulerOS release 2.9 (restricted use. Submit a service ticket to apply for it.)	v1.29	√	√	x	4.18.0-147.5.1.6.h841.eulerosv2r9.x86_64
	v1.28	√	√	x	4.18.0-147.5.1.6.h841.eulerosv2r9.x86_64
	v1.27	√	√	x	4.18.0-147.5.1.6.h841.eulerosv2r9.x86_64
	v1.25	√	√	x	4.18.0-147.5.1.6.h841.eulerosv2r9.x86_64

OS	Cluster Version	CCE Standard Cluster		CCE Turbo Cluster	Latest Kernel
		VPC Network	Tunnel Network	Cloud Native 2.0 Network	
	v1.23	√	√	x	4.18.0-147.5.1.6.h841.eulerov2r9.x86_64
	v1.21 (end of maintenance)	√	√	x	4.18.0-147.5.1.6.h841.eulerov2r9.x86_64
	v1.19 (end of maintenance)	√	√	x	4.18.0-147.5.1.6.h841.eulerov2r9.x86_64
EulerOS release 2.3 (end of maintenance)	v1.27 or later	x	x	x	None
	v1.25 (end of maintenance)	√	√	x	3.10.0-514.41.4.28.h62.x86_64
	v1.23 (end of maintenance)	√	√	x	3.10.0-514.41.4.28.h62.x86_64
	v1.21 (end of maintenance)	√	√	x	3.10.0-514.41.4.28.h62.x86_64
	v1.19 (end of maintenance)	√	√	x	3.10.0-514.41.4.28.h62.x86_64
	v1.17 (end of maintenance)	√	√	x	3.10.0-514.41.4.28.h62.x86_64
	v1.15.11 (end of maintenance)	√	√	x	3.10.0-514.41.4.28.h62.x86_64

## 3.4 Creating a Node

### Prerequisites

- At least one cluster has been created.

### Notes and Constraints

- The DNS configuration of a subnet where a node is located cannot be modified because OBS and other dependent services are necessary for creating nodes.

### Precautions

- To maintain node stability, Kubernetes components such as kubelet, kube-proxy, and docker, as well as the Kubernetes system, will be allocated a specific number of node resources based on the node flavor. Therefore, the total number of node resources and the number of allocatable node resources for your cluster are different. The larger the node specifications, the more the containers deployed on the node. Therefore, more node resources need to be reserved to run Kubernetes components. For details, see [Node Resource Reservation Policy](#).
- Networks including VM networks and container networks of nodes are all managed by CCE. Do not add or delete ENIs, or change routes and IP addresses. Otherwise, services may be unavailable. For example, the ENI named **gw\_11cbf51a@eth0** on the node is the container network gateway and cannot be modified.

### Procedure

After a cluster is created, you can create nodes for the cluster.

**Step 1** Log in to the [CCE console](#).

**Step 2** In the navigation pane of the CCE console, choose **Clusters**. Click the target cluster name to access its details page.

**Step 3** In the navigation pane, choose **Nodes**. On the page displayed, click the **Nodes** tab and then **Create Node** in the upper right corner. Configure node parameters.

#### Configurations

You can configure the flavor and OS of a cloud server, on which your containerized applications run.

**Table 3-10** Node configuration parameters

Parameter	Description
Billing Mode	<p>The following billing modes are supported:</p> <ul style="list-style-type: none"> <li>• Pay-per-use Resources will be billed based on usage duration. You can provision or delete resources at any time.</li> </ul>
AZ	<p>AZ where the node is located. Nodes in a cluster can be created in different AZs for higher reliability. The value cannot be changed after the node is created.</p> <p>Select <b>Random</b> to deploy your node in a random AZ based on the selected node flavor.</p> <p>An AZ is a physical region where resources use independent power supply and networks. AZs are physically isolated but interconnected through an internal network. To enhance workload availability, create nodes in different AZs.</p>
Node Type	<p>Select a node type based on service requirements. Then, the available node flavors will be automatically displayed in the <b>Specifications</b> area for you to select.</p> <p>CCE standard clusters support the following node types:</p> <ul style="list-style-type: none"> <li>• ECS (VM): A virtualized ECS is used as a cluster node.</li> </ul> <p>CCE Turbo clusters support the following node types:</p> <ul style="list-style-type: none"> <li>• ECS (VM): A virtualized ECS is used as a cluster node. A CCE Turbo cluster supports only the cloud servers that allow multiple ENIs. Select a server type displayed on the CCE console.</li> <li>• ECS (physical machine): A QingTian-backed bare metal server is used as a cluster node.</li> </ul>
Specifications	<p>Select node flavors as needed. A node needs at least two vCPU cores and 4 GiB of memory.</p> <p>The available node flavors vary depending on AZs. Obtain the flavors displayed on the console.</p>
Container Engine	<p>The container engines supported by CCE include Docker and containerd, which may vary depending on cluster types, cluster versions, and OSs. Select a container engine based on the information displayed on the CCE console. For details, see <a href="#">Mapping Between Node OSs and Container Engines</a>.</p>
OS	<p>Select an OS type. Different types of nodes support different OSs.</p> <ul style="list-style-type: none"> <li>• <b>Public image:</b> Select a public image for the node.</li> </ul> <p><b>NOTE</b> 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.</p>

Parameter	Description
Node Name	<p>Name of the node. When nodes (ECSs) are created in batches, the value of this parameter is used as the name prefix for each ECS.</p> <p>The system generates a default name for you, which can be modified.</p> <p>Enter 1 to 56 characters. Only lowercase letters, digits, hyphens (-), and periods (.) are allowed. The name must start with a lowercase letter and cannot end with a hyphen (-). Only lowercase letters or digits are allowed before and after periods (.).</p>
Login Mode	

### Storage Settings

Configure storage resources on a node for the containers running on it. Select a disk type and configure its size based on service requirements. For details about EVS disks, see [Disk Types and Performance](#).

**Table 3-11** Configuration parameters

Parameter	Description
System Disk	System disk used by the node OS. The value ranges from 40 GiB to 1024 GiB. The default value is 50 GiB.



Parameter	Description
Data Disk	<p><b>At least one data disk is required</b> for the container runtime and kubelet components. <b>This data disk cannot be deleted or detached. Otherwise, the node will be unavailable.</b></p> <ul style="list-style-type: none"> <li>• Default data disk: used for container runtime and kubelet components. The disk size ranges from 20 GiB to 32768 GiB. The default value is 100 GiB.</li> <li>• Other common data disks: You can set the data disk size to a value ranging from 10 GiB to 32768 GiB. The default value is 100 GiB.</li> </ul> <p><b>NOTE</b></p> <ul style="list-style-type: none"> <li>• If the node flavor is disk-intensive or ultra-high I/O, one data disk can be a local disk.</li> <li>• Local disks may break down and do not ensure data reliability. Store your service data in EVS disks, which are more reliable than local disks.</li> </ul> <p><b>Advanced Settings</b></p> <p>Expand the area and configure the following parameters:</p> <ul style="list-style-type: none"> <li>• <b>Data Disk Space Allocation:</b> allocates space for container engines, images, and ephemeral storage for them to run properly. For details about how to allocate data disk space, see <a href="#">Space Allocation of a Data Disk</a>.</li> </ul> <p><b>Adding data disks</b></p> <p>A maximum of 16 data disks can be attached to an ECS and 10 to a BMS. By default, a raw disk is created without any processing. You can also click <b>Expand</b> and select any of the following options:</p> <ul style="list-style-type: none"> <li>• <b>Default:</b> By default, a raw disk is created without any processing.</li> <li>• <b>Mount Disk:</b> The data disk is attached to a specified directory.</li> </ul>

### Network Settings

Configure networking resources to allow node and containerized application access.

**Table 3-12** Configuration parameters

Parameter	Description
VPC/Node Subnet	The node subnet selected during cluster creation is used by default. You can choose another subnet instead.
Node IP Address	IP address of the specified node. By default, the value is randomly allocated.

Parameter	Description
EIP	An ECS without a bound EIP cannot access the Internet or be accessed by public networks. The default value is <b>Do not use</b> . <b>Use existing</b> and <b>Auto create</b> are supported.

### Advanced Settings

Configure advanced node capabilities such as labels, taints, and startup command.

**Table 3-13** Advanced configuration parameters

Parameter	Description
Resource Tag	You can add resource tags to classify resources. A maximum of eight resource tags can be added. You can create <b>predefined tags</b> on the TMS console. The predefined tags are available to all resources that support tags. You can use these tags to improve the tag creation and resource migration efficiency. CCE will automatically create the "CCE-Dynamic-Provisioning-Node= <i>Node ID</i> " tag.
Kubernetes Label	A key-value pair added to a Kubernetes object (such as a pod). After specifying a label, click <b>Add Label</b> for more. A maximum of 20 labels can be added. Labels can be used to distinguish nodes. With workload affinity settings, container pods can be scheduled to a specified node. For more information, see <a href="#">Labels and Selectors</a> .

Parameter	Description
Taint	<p>This parameter is left blank by default. You can add taints to configure anti-affinity for the node. A maximum of 20 taints are allowed for each node. Each taint contains the following parameters:</p> <ul style="list-style-type: none"> <li>• <b>Key:</b> A key must contain 1 to 63 characters, starting with a letter or digit. Only letters, digits, hyphens (-), underscores (_), and periods (.) are allowed. A DNS subdomain name can be used as the prefix of a key.</li> <li>• <b>Value:</b> A value must contain 1 to 63 characters, starting with a letter or digit. Only letters, digits, hyphens (-), underscores (_), and periods (.) are allowed.</li> <li>• <b>Effect:</b> Available options are <b>NoSchedule</b>, <b>PreferNoSchedule</b>, and <b>NoExecute</b>.</li> </ul> <p>For details, see <a href="#">Managing Node Taints</a>.</p> <p><b>NOTE</b> For a cluster of v1.19 or earlier, the workload may have been scheduled to a node before the taint is added. To avoid such a situation, select a cluster of v1.19 or later.</p>
Max. Pods	<p>Maximum number of pods that can run on the node, including the default system pods.</p> <p>This limit prevents the node from being overloaded with pods.</p> <p>This number is also decided by other factors. For details, see <a href="#">Maximum Number of Pods That Can Be Created on a Node</a>.</p>
ECS Group	<p>An ECS group logically groups ECSs. The ECSs in the same ECS group comply with the same policy associated with the ECS group.</p> <p><b>Anti-affinity:</b> ECSs in an ECS group are deployed on different physical hosts to improve service reliability.</p> <p>Select an existing ECS group, or click <b>Add ECS Group</b> to create one. After the ECS group is created, click the refresh icon.</p>
Pre-installation Command	<p>Installation script command, in which Chinese characters are not allowed. The script command will be Base64-transcoded.</p> <p>The script will be executed before Kubernetes software is installed. Note that if the script is incorrect, Kubernetes software may fail to be installed.</p>

Parameter	Description
Post-installation Command	<p>Installation script command, in which Chinese characters are not allowed. The script command will be Base64-transcoded. The script will be executed after Kubernetes software is installed, which does not affect the installation.</p> <p><b>NOTE</b> Do not run the <b>reboot</b> command in the post-installation script to restart the system immediately. To restart the system, run the <b>shutdown -r 1</b> command to restart with a delay of one minute.</p>
Agency	<p>An agency is created by the tenant administrator on the IAM console. Using an agency, you can share your cloud server resources with another account, or entrust a more professional person or team to manage your resources.</p> <p>If no agency is available, click <b>Create Agency</b> on the right to create one.</p>

**Step 4** Configure the number of nodes to be purchased. Then, click **Next: Confirm**. Confirm the configured parameters and specifications.

**Step 5** Click **Submit**.

The node list page is displayed. If the node status is **Running**, the node is created successfully. It takes about 6 to 10 minutes to create a node.

**Step 6** Click **Back to Node List**. The node is created successfully if it changes to the **Running** state.

----End

## 3.5 Accepting Nodes for Management

### Scenario

In CCE, you can create a node ([Creating a Node](#)) or add existing nodes (ECSs) to your cluster for management.

#### NOTICE

- When accepting an ECS, you can reset the ECS OS to a standard public image offered by CCE. If you choose to do so, you need to reset the password or key pair, and the previous password or key pair will become invalid.
- LVM information, including volume groups (VGs), logical volumes (LVs), and physical volumes (PVs), will be deleted from the system disks and data disks attached to the selected ECSs during acceptance. Ensure that the information has been backed up.
- During the acceptance of an ECS, do not perform any operation on the ECS through the ECS console.

## Notes and Constraints

- BMSs and ECSs can be managed.

## Prerequisites

The cloud servers to be managed must meet the following requirements:

- The node to be accepted must be in the **Running** state and not used by other clusters. In addition, the node to be accepted does not carry the CCE-Dynamic-Provisioning-Node tag.
- The node to be accepted and the cluster must be in the same VPC. (If the cluster version is earlier than v1.13.10, the node to be accepted and the CCE cluster must be in the same subnet.)
- Data disks must be attached to the nodes to be managed. A local disk (disk-intensive disk) or a data disk of at least 20 GiB can be attached to the node, and any data disks already attached cannot be smaller than 10 GiB.
- The node to be accepted has 2-core or higher CPU, 4 GiB or larger memory, and only one NIC.
- If an enterprise project is used, the node to be accepted and the cluster must be in the same enterprise project. Otherwise, resources cannot be identified during management. As a result, the node cannot be accepted.
- Only cloud servers with the same data disk configuration can be accepted in batches for management.
- Nodes in a CCE Turbo cluster must support sub-ENIs or be bound to at least 16 ENIs. For details about the node flavors, see the node flavors that can be selected on the console when you create a node.
- Data disks that have been partitioned will be ignored during node management. Ensure that there is at least one unpartitioned data disk meeting the specifications is attached to the node.

## Procedure

- Step 1** Log in to the CCE console and go to the cluster where the node to be accepted resides.
- Step 2** In the navigation pane, choose **Nodes**. On the displayed page, click the **Nodes** tab and then **Accept Node** in the upper right corner.
- Step 3** Specify node parameters.

### Configurations

**Table 3-14** Node configuration parameters

Parameter	Description
Node Pool	<ul style="list-style-type: none"> <li>Default node pool: You can add nodes that meet the management requirements to the default node pool of the cluster.</li> <li>Custom node pool: You only need to select the nodes that meet the management requirements. These nodes will use the basic, network, and advanced settings of the custom node pool. You do not need to configure other parameters. For details, see <a href="#">Accepting Nodes in a Node Pool</a>.</li> </ul>
Specifications	<p>Click <b>Select Cloud Server</b> and select the servers to be accepted. You can select multiple cloud servers for batch management. However, only the cloud servers with the same specifications, AZ, and data disk configuration can be added in batches. If a cloud server contains multiple data disks, select one of them for the container runtime and kubelet.</p>
Container Engine	<p>The container engines supported by CCE include Docker and containerd, which may vary depending on cluster types, cluster versions, and OSs. Select a container engine based on the information displayed on the CCE console. For details, see <a href="#">Mapping Between Node OSs and Container Engines</a>.</p>
OS	<p>Select an OS type. Different types of nodes support different OSs.</p> <ul style="list-style-type: none"> <li><b>Public image:</b> Select a public image for the node.</li> </ul> <p><b>NOTE</b> 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.</p>
Login Mode	

### Storage Settings

Configure storage resources on a node for the containers running on it.

**Table 3-15** Configuration parameters

Parameter	Description
System Disk	Directly use the system disk of the cloud server.

Parameter	Description
Data Disk	<p><b>At least one data disk is required</b> for the container runtime and kubelet components. <b>This data disk cannot be deleted or detached. Otherwise, the node will be unavailable.</b></p> <p>Click <b>Expand</b> to configure <b>Data Disk Space Allocation</b>, which is used to allocate space for container engines, images, and ephemeral storage for them to run properly. For details about how to allocate data disk space, see <a href="#">Space Allocation of a Data Disk</a>.</p> <p>For other data disks, a raw disk is created without any processing by default. You can also click <b>Expand</b> and select <b>Mount Disk</b> to mount the data disk to a specified directory.</p>

### Advanced Settings

**Table 3-16** Advanced configuration parameters

Parameter	Description
Resource Tag	<p>You can add resource tags to classify resources. A maximum of eight resource tags can be added.</p> <p>You can create <b>predefined tags</b> on the TMS console. The predefined tags are available to all resources that support tags. You can use these tags to improve the tag creation and resource migration efficiency.</p> <p>CCE will automatically create the "CCE-Dynamic-Provisioning-Node=<i>Node ID</i>" tag.</p>
Kubernetes Label	<p>Click <b>Add Label</b> to set the key-value pair attached to the Kubernetes objects (such as pods). A maximum of 20 labels can be added.</p> <p>Labels can be used to distinguish nodes. With workload affinity settings, container pods can be scheduled to a specified node. For more information, see <a href="#">Labels and Selectors</a>.</p>

Parameter	Description
Taint	<p>This parameter is left blank by default. You can add taints to configure anti-affinity for the node. A maximum of 20 taints are allowed for each node. Each taint contains the following parameters:</p> <ul style="list-style-type: none"> <li>• <b>Key:</b> A key must contain 1 to 63 characters, starting with a letter or digit. Only letters, digits, hyphens (-), underscores (_), and periods (.) are allowed. A DNS subdomain name can be used as the prefix of a key.</li> <li>• <b>Value:</b> A value must contain 1 to 63 characters, starting with a letter or digit. Only letters, digits, hyphens (-), underscores (_), and periods (.) are allowed.</li> <li>• <b>Effect:</b> Available options are <b>NoSchedule</b>, <b>PreferNoSchedule</b>, and <b>NoExecute</b>.</li> </ul> <p><b>NOTICE</b></p> <ul style="list-style-type: none"> <li>• If taints are used, you must configure tolerations of pods. Otherwise, a scale-out may fail or pods cannot be scheduled onto the added nodes.</li> <li>• After a node pool is created, you can click <b>Edit</b> to modify its configuration. The modification will be synchronized to all nodes in the node pool.</li> </ul>
Max. Pods	<p>Maximum number of pods that can run on the node, including the default system pods.</p> <p>This limit prevents the node from being overloaded with pods.</p>
Pre-installation Command	<p>Installation script command, in which Chinese characters are not allowed. The script command will be Base64-transcoded.</p> <p>The script will be executed before Kubernetes software is installed. Note that if the script is incorrect, Kubernetes software may fail to be installed.</p>
Post-installation Command	<p>Installation script command, in which Chinese characters are not allowed. The script command will be Base64-transcoded.</p> <p>The script will be executed after Kubernetes software is installed, which does not affect the installation.</p>

**Step 4** Click **Next: Confirm**. Click **Submit**.

----End

## 3.6 Management Nodes

### 3.6.1 Managing Node Labels

You can add different labels to nodes and define different attributes for labels. By using these node labels, you can quickly understand the characteristics of each node.



## Node Label Usage Scenario

Node labels are mainly used in the following scenarios:

- Node management: Node labels are used to classify nodes.
- Node affinity or anti-affinity for workloads: By adding labels to nodes, you can schedule pods to specific nodes through node affinity or prevent pods from being scheduled to specific nodes through node anti-affinity. For details, see [Configuring Node Affinity Scheduling \(nodeAffinity\)](#).

## Inherent Label of a Node

After a node is created, some fixed labels exist and cannot be deleted. For details about these labels, see [Table 3-17](#).

### NOTE

Do not manually change the inherent labels that are automatically added to a node. If the manually changed value conflicts with the system value, the system value is used.

**Table 3-17** Inherent labels of a node

Key	Description
New: topology.kubernetes.io/region Old: failure-domain.beta.kubernetes.io/region	Region where the node is located
New: topology.kubernetes.io/zone Old: failure-domain.beta.kubernetes.io/zone	AZ where the node is located
New: node.kubernetes.io/baremetal Old: failure-domain.beta.kubernetes.io/is-baremetal	Whether the node is a bare metal node <b>false</b> indicates that the node is not a bare metal node.
node.kubernetes.io/container-engine	Container engine Example: <b>docker</b> or <b>containerd</b>
node.kubernetes.io/instance-type	Node specifications
kubernetes.io/arch	Node processor architecture
kubernetes.io/hostname	Node name
kubernetes.io/os	Node OS type
node.kubernetes.io/subnetid	ID of the subnet where the node is located
os.architecture	Node processor architecture For example, <b>amd64</b> indicates a AMD64-bit processor.

Key	Description
os.name	Node OS name
os.version	Node OS kernel version
accelerator/huawei-npu	NPU node labels
accelerator	GPU node labels
cce.cloud.com/cce-nodepool	The dedicated label of a node in a node pool

## Adding or Deleting a Node Label

- 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 **Nodes**. On the displayed page, click the **Nodes** tab, select the target node and click **Labels and Taints** in the upper left corner.
- Step 3** In the displayed dialog box, click **Add operation** under **Batch Operation**, and then choose **Add/Update** or **Delete**.
- Enter the key and value of the label to be added or deleted, and click **OK**.
- For example, the key is **deploy\_qa** and the value is **true**, indicating that the node is used to deploy the QA (test) environment.
- Step 4** After the label is added, check the added label in node data.
- End

## 3.6.2 Managing Node Taints

Taints enable a node to repel specific pods to prevent these pods from being scheduled to the node.

### Procedure for Operations Performed on the Console

On the CCE console, you can also batch manage nodes' taints.

- 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 **Nodes**. On the displayed page, click the **Nodes** tab, select the target node and click **Labels and Taints** in the upper left corner.
- Step 3** In the displayed dialog box, click **Add Operation** under **Batch Operation**, and then choose **Add/Update** or **Delete** as well as **Taint**.
- Enter the key and value of the taint to be operated, choose a taint effect, and click **OK**.
- Step 4** After the taint is added, check the added taint in node data.
- End

## Procedure for Operations Performed Through kubectl

A taint is a key-value pair associated with an effect. The following effects are available:

- **NoSchedule**: No pod will be scheduled onto the node unless it has a matching toleration. Existing pods will not be evicted from the node.
- **PreferNoSchedule**: Kubernetes prevents pods that cannot tolerate this taint from being scheduled onto the node.
- **NoExecute**: If the pod has been running on a node, the pod will be evicted from the node. If the pod has not been running on a node, the pod will not be scheduled onto the node.

To add a taint to a node, run the **kubectl taint node *nodename*** command as follows:

```
$ kubectl get node
NAME          STATUS    ROLES    AGE   VERSION
192.168.10.170 Ready    <none>   73d   v1.19.8-r1-CCE21.4.1.B003
192.168.10.240 Ready    <none>   4h8m  v1.19.8-r1-CCE21.6.1.2.B001
$ kubectl taint node 192.168.10.240 key1=value1:NoSchedule
node/192.168.10.240 tainted
```

To view the taint configuration, run the **describe** and **get** commands as follows:

```
$ kubectl describe node 192.168.10.240
Name:          192.168.10.240
...
Taints:        key1=value1:NoSchedule
...
$ kubectl get node 192.168.10.240 -oyaml
apiVersion: v1
...
spec:
  providerID: 06a5ea3a-0482-11ec-8e1a-0255ac101dc2
  taints:
  - effect: NoSchedule
    key: key1
    value: value1
...
```

To remove a taint, add a hyphen (-) at the end of the command for adding a taint, as shown in the following example:

```
$ kubectl taint node 192.168.10.240 key1=value1:NoSchedule-
node/192.168.10.240 untainted
$ kubectl describe node 192.168.10.240
Name:          192.168.10.240
...
Taints:        <none>
...
```

## Configuring a Node Scheduling Policy in One-Click Mode

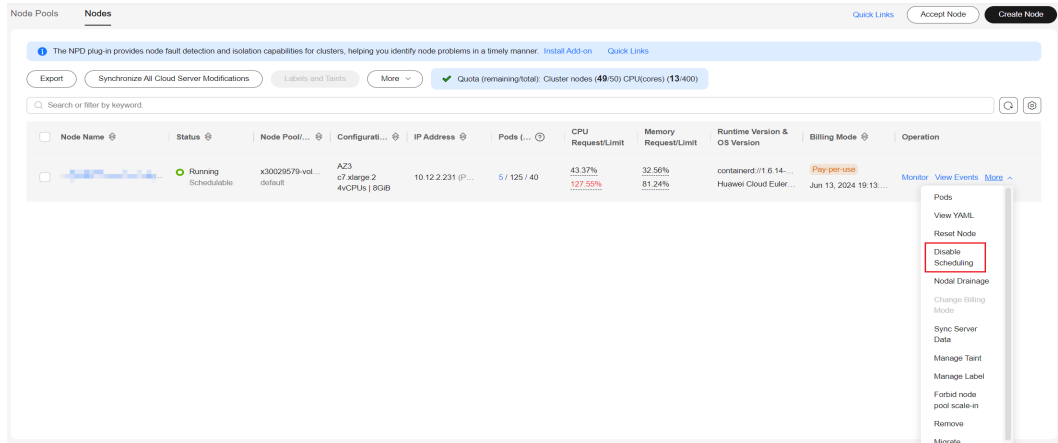
You can configure a node to be unschedulable on the console. Then, CCE will add a taint with key **node.kubernetes.io/unschedulable** and the **NoSchedule** setting to the node. After a node is set to be unschedulable, new pods cannot be scheduled to this node, but pods running on the node are not affected.

**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 **Nodes**. On the displayed page, click the **Nodes** tab.

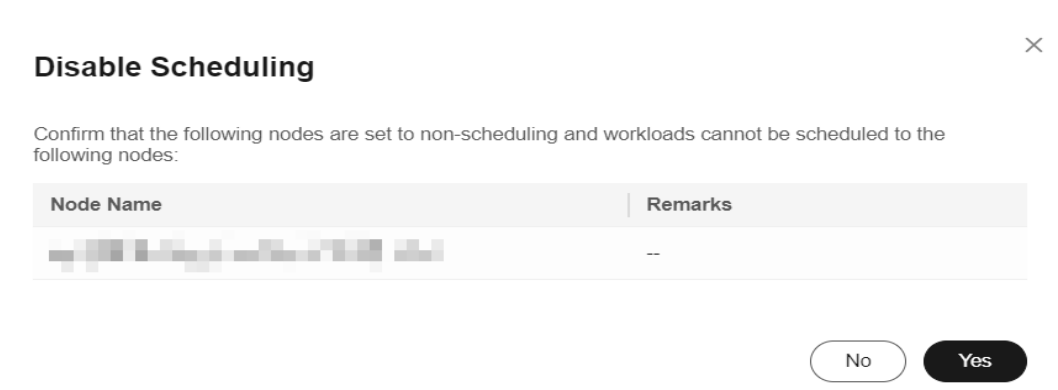
**Step 3** In the node list, locate the target node and choose **More > Disable Scheduling** in the **Operation** column.

**Figure 3-1** Disable Scheduling



**Step 4** In the dialog box that is displayed, click **Yes** to configure the node to be unschedulable.

**Figure 3-2** Disable Scheduling



This operation will add a taint to the node. You can use `kubectl` to view the content of the taint.

```
$ kubectl describe node 192.168.10.240
...
Taints:      node.kubernetes.io/unschedulable:NoSchedule
...
```

**Step 5** Go back to the node list, locate the target node, and choose **More > Enable Scheduling**. Then, the node changes to be schedulable.

----End

## System Taints

When some issues occurred on a node, Kubernetes automatically adds a taint to the node. The built-in taints are as follows:

- `node.kubernetes.io/not-ready`: The node is not ready. The node **Ready** value is **False**.
- `node.kubernetes.io/unreachable`: The node controller cannot access the node. The node **Ready** value is **Unknown**.
- `node.kubernetes.io/memory-pressure`: The node memory is approaching the upper limit.
- `node.kubernetes.io/disk-pressure`: The node disk space is approaching the upper limit.
- `node.kubernetes.io/pid-pressure`: The node PIDs are approaching the upper limit.
- `node.kubernetes.io/network-unavailable`: The node network is unavailable.
- `node.kubernetes.io/unschedulable`: The node cannot be scheduled.
- `node.cloudprovider.kubernetes.io/uninitialized`: If an external cloud platform driver is specified when kubelet is started, kubelet adds a taint to the current node and marks it as unavailable. After a controller of **cloud-controller-manager** initializes the node, kubelet will delete the taint.

## Related Operations (Tolerations)

Tolerations are applied to pods, and allow (but do not require) the pods to schedule onto nodes with matching taints.

Taints and tolerations work together to ensure that pods are not scheduled onto inappropriate nodes. One or more taints are applied to a node. This marks that the node should not accept any pods that do not tolerate the taints.

Example:

```
apiVersion: v1
kind: Pod
metadata:
  name: nginx
  labels:
    env: test
spec:
  containers:
  - name: nginx
    image: nginx
    imagePullPolicy: IfNotPresent
  tolerations:
  - key: "key1"
    operator: "Equal"
    value: "value1"
    effect: "NoSchedule"
```

In the preceding example, the toleration label of the pod is `key1=value1` and the taint effect is `NoSchedule`. Therefore, the pod can be scheduled onto the corresponding node.

You can also configure tolerations similar to the following information, which indicates that the pod can be scheduled onto a node when the node has the taint `key1`:

```
tolerations:
- key: "key1"
  operator: "Exists"
  effect: "NoSchedule"
```

## 3.6.3 Resetting a Node

### Scenario

You can reset a node to modify the node configuration, such as the node OS and login mode.

Resetting a node will reinstall the node OS and the Kubernetes software on the node. If a node is unavailable because you modify the node configuration, you can reset the node to rectify the fault.

### Notes and Constraints

- To enable node resetting in CCE standard clusters, the version must be v1.13 or later.

### Precautions

- Only worker nodes can be reset. If the node is still unavailable after the resetting, delete the node and create a new one.
- **After a node is reset, the node OS will be reinstalled. Before resetting a node, drain the node to gracefully evict the pods running on the node to other available nodes. Perform this operation during off-peak hours.**
- **After a node is reset, its system disk and data disks will be cleared. Back up important data before resetting a node.**
- **If you reset a worker node that has an additional data disk attached on the ECS console, the attachment will be removed. To keep the data, you need to reattach the disk.**
- The IP addresses of the workload pods on the node will change, but the container network access is not affected.
- There is remaining EVS disk quota.
- When a node is reset, the backend will make it unschedulable.
- Resetting a node will clear the Kubernetes labels and taints you added (those added by editing a node pool will not be lost). As a result, node-specific resources (such as local storage and workloads scheduled to this node) may be unavailable.

### Resetting Nodes in the Default Pool

- 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 **Nodes**. On the displayed page, click the **Nodes** tab.
- Step 3** In the node list of the default pool, select one or more nodes to be reset and choose **More > Reset Node** in the **Operation** column.
- Step 4** In the displayed dialog box, click **Next**.
- Step 5** Specify node parameters.

#### Compute Settings

**Table 3-18** Configuration parameters

Parameter	Description
Specifications	Specifications cannot be modified when you reset a node.
Container Engine	The container engines supported by CCE include Docker and containerd, which may vary depending on cluster types, cluster versions, and OSs. Select a container engine based on the information displayed on the CCE console. For details, see <a href="#">Mapping Between Node OSs and Container Engines</a> .
OS	Select an OS type. Different types of nodes support different OSs. <ul style="list-style-type: none"> <li>• <b>Public image:</b> Select a public image for the node.</li> </ul> <p><b>NOTE</b> 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.</p>
Login Mode	

### Storage Settings

Configure storage resources on a node for the containers running on it.

**Table 3-19** Configuration parameters

Parameter	Description
System Disk	Directly use the system disk of the cloud server.
Data Disk	<p><b>At least one data disk is required</b> for the container runtime and kubelet components. <b>This data disk cannot be deleted or detached. Otherwise, the node will be unavailable.</b></p> <p>Click <b>Expand</b> to configure <b>Data Disk Space Allocation</b>, which is used to allocate space for container engines, images, and ephemeral storage for them to run properly. For details about how to allocate data disk space, see <a href="#">Space Allocation of a Data Disk</a>.</p> <p>For other data disks, a raw disk is created without any processing by default. You can also click <b>Expand</b> and select <b>Mount Disk</b> to mount the data disk to a specified directory.</p>

### Advanced Settings

**Table 3-20** Advanced configuration parameters

Parameter	Description
Resource Tag	<p>You can add resource tags to classify resources. A maximum of eight resource tags can be added.</p> <p>You can create predefined tags on the TMS console. The predefined tags are available to all resources that support tags. You can use these tags to improve the tag creation and resource migration efficiency.</p> <p>CCE will automatically create the <b>CCE-Dynamic-Provisioning-Node=Node ID</b> tag.</p>
Kubernetes Label	<p>Click <b>Add Label</b> to set the key-value pair attached to the Kubernetes objects (such as pods). A maximum of 20 labels can be added.</p> <p>Labels can be used to distinguish nodes. With workload affinity settings, container pods can be scheduled to a specified node. For more information, see <a href="#">Labels and Selectors</a>.</p>
Taint	<p>This parameter is left blank by default. You can add taints to configure anti-affinity for the node. A maximum of 20 taints are allowed for each node. Each taint contains the following parameters:</p> <ul style="list-style-type: none"> <li>• <b>Key:</b> A key must contain 1 to 63 characters, starting with a letter or digit. Only letters, digits, hyphens (-), underscores (_), and periods (.) are allowed. A DNS subdomain name can be used as the prefix of a key.</li> <li>• <b>Value:</b> A value must contain 1 to 63 characters, starting with a letter or digit. Only letters, digits, hyphens (-), underscores (_), and periods (.) are allowed.</li> <li>• <b>Effect:</b> Available options are <b>NoSchedule</b>, <b>PreferNoSchedule</b>, and <b>NoExecute</b>.</li> </ul> <p><b>NOTICE</b></p> <ul style="list-style-type: none"> <li>• If taints are used, you must configure tolerations of pods. Otherwise, a scale-out may fail or pods cannot be scheduled onto the added nodes.</li> <li>• After a node pool is created, you can click <b>Edit</b> to modify its configuration. The modification will be synchronized to all nodes in the node pool.</li> </ul>
Max. Pods	<p>Maximum number of pods that can run on the node, including the default system pods.</p> <p>This limit prevents the node from being overloaded with pods.</p>
Pre-installation Command	<p>Installation script command, in which Chinese characters are not allowed. The script command will be Base64-transcoded.</p> <p>The script will be executed before Kubernetes software is installed. Note that if the script is incorrect, Kubernetes software may fail to be installed.</p>



Parameter	Description
Post-installation Command	Installation script command, in which Chinese characters are not allowed. The script command will be Base64-transcoded. The script will be executed after Kubernetes software is installed, which does not affect the installation.

**Step 6** Click **Next: Confirm**.

**Step 7** Click **Submit**.

----End

## 3.6.4 Removing a Node

### Scenario

Removing a node from a cluster will re-install the node OS and clear CCE components on the node.

Removing a node will not delete the server corresponding to the node. You are advised to remove nodes at off-peak hours to avoid impacts on your services.

After a node is removed from the cluster, the node is still running.

### Notes and Constraints

- If the OS fails to be re-installed after the node is removed, manually re-install the OS. After the re-installation, log in to the node and run the clearance script to clear CCE components. For details, see [Handling Failed OS Reinstallation](#).

### Precautions

- When you remove a node, the pods running on it will need to migrate to another node, which could potentially impact your services. To avoid any disruptions, drain the node so that pods can be gracefully evicted to other available nodes. Additionally, perform this task during off-peak hours.
- Unexpected risks may occur during the operation. Back up data beforehand.
- When a node is removed, the backend will make it unschedulable.
- After you remove the node and re-install the OS, the original LVM partitions will be cleared and the data managed by LVM will be cleared. Therefore, back up data in advance.

### Procedure

- 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 **Nodes**. On the displayed page, click the **Nodes** tab.
- Step 3** Locate the target node and choose **More > Remove** in the **Operation** column.
- Step 4** In the dialog box displayed, configure the login information required for re-installing the OS and click **Yes**. Wait until the node is removed.

After the node is removed, workload pods on the node are automatically migrated to other available nodes.

----End

## Handling Failed OS Reinstallation

You can perform the following steps to re-install the OS and clear the CCE components on the node if previous attempts fail:

**Step 1** Log in to the management console of the server and re-install the OS.

**Step 2** Log in to the server and run the following commands to clear the CCE components and LVM data:

Write the following scripts to the **clean.sh** file:

```
lsblk
vgs --noheadings | awk '{print $1}' | xargs vgremove -f
pvs --noheadings | awk '{print $1}' | xargs pvremove -f
lvs --noheadings | awk '{print $1}' | xargs -i lvremove -f --select {}
function init_data_disk() {
    all_devices=$(lsblk -o KNAME,TYPE | grep disk | grep -v nvme | awk '{print $1}' | awk '{ print "/dev/"$1}')
    for device in ${all_devices[@]}; do
        isRootDisk=$(lsblk -o KNAME,MOUNTPOINT $device 2>/dev/null | grep -E '[:,space:]]/$' | wc -l )
        if [[ ${isRootDisk} != 0 ]]; then
            continue
        fi
        dd if=/dev/urandom of=${device} bs=512 count=64
    done
    return
}
init_data_disk
lsblk
```

Run the following command:

```
bash clean.sh
```

----End

## 3.6.5 Synchronizing the Data of Cloud Servers

### Scenario

Each node in a cluster is a cloud server or physical machine. After a cluster node is created, you can change the cloud server name or specifications as required. Modifying node specifications will affect services. Perform the operation on nodes one by one.

Some information of CCE nodes is maintained independently from the ECS console. After you change the name, EIP, or specifications of an ECS on the ECS console, synchronize the ECS with the target node on the CCE console. After the synchronization, information on both consoles is consistent.

### Notes and Constraints

- Data, including the VM status, ECS names, number of CPUs, size of memory, ECS specifications, and public IP addresses, can be synchronized.

- The following data cannot be synchronized: OS, image ID, and disk configuration.

## Synchronizing the Data of a Cloud 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 **Nodes**. On the displayed page, click the **Nodes** tab.
- Step 3** Locate the target node and choose **More > Sync Server Data** in the **Operation** column.

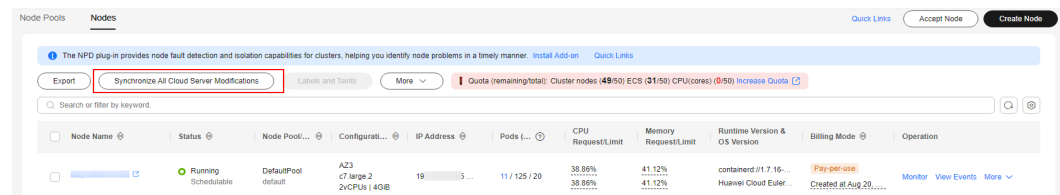
After the synchronization is complete, the **ECS data synchronization requested** message is displayed in the upper right corner.

----End

## Synchronizing the Data of All Cloud Servers

- 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 **Nodes**. On the displayed page, click the **Nodes** tab.
- Step 3** Click **Synchronize All Cloud Server Modifications** and click **OK**.

**Figure 3-3** Synchronize All Cloud Server Modifications



After the synchronization is complete, the **ECS data synchronization requested** message is displayed in the upper right corner.

----End

## 3.6.6 Draining a Node

### Scenario

After you enable nodal drainage on the console, CCE configures the node to be non-schedulable and securely evicts all pods that comply with **Rules for Draining Nodes** on the node. Subsequent new pods will not be scheduled to this node.

When a node becomes faulty, nodal drainage quickly isolates the faulty node. The pods evicted from the faulty node will be scheduled by the workload controller to other nodes that are running properly.

**NOTICE**

To ensure service availability during drainage, [specify a disruption budget](#) for your application. Otherwise, the application may become unavailable during pod rescheduling.

**Prerequisites**

- A cluster is available and the cluster version meets the following requirements:
  - v1.21: v1.21.10-r0 or later
  - v1.23: v1.23.8-r0 or later
  - v1.25: v1.25.3-r0 or later
  - Versions later than v1.25
- To drain a node as an IAM user, you must have at least one of the following permissions (for details, see [Namespace Permissions \(Kubernetes RBAC-based\)](#)):
  - cluster-admin (administrator): read and write permissions on all resources in all namespaces.
  - drainage-editor: drain a node.
  - drainage-viewer: view the nodal drainage status but cannot drain a node.

**Rules for Draining Nodes**

When a node is drained, all pods on the node will be safely evicted. However, CCE will take specific actions for pods that meet certain filtering criteria.

Filter Criterion	Forced Drainage Enabled	Forced Drainage Disabled
The <a href="#">status.phase</a> field of the pod is <b>Succeeded</b> or <b>Failed</b> .	Deletion	Deletion
The pod is not managed by the workload controller.	Deletion	Drainage cancellation
The pod is managed by DaemonSet.	None	Drainage cancellation
A volume of the emptyDir type is mounted to the pod.	Eviction	Drainage cancellation
The pod is a <a href="#">static pod</a> directly managed by kubelet	None	None

 NOTE

The following operations may be performed on pods during node drainage:

- Deletion: The pod is deleted from the current node and will not be scheduled to other nodes.
- Eviction: The pod is deleted from the current node and rescheduled to another node.
- None: The pod will not be evicted or deleted.
- Drainage cancellation: If a pod on a node cancels drainage, the drainage process of the node is terminated and no pod is evicted or deleted.

## Procedure

This section describes how to drain nodes.

## Drainage Through the Console

- 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 **Nodes**. On the displayed page, click the **Nodes** tab.
- Step 3** Locate the target node and choose **More > Nodal Drainage** in the **Operation** column.
- Step 4** In the **Nodal Drainage** window displayed, set parameters.
  - **Timeout (s)**: Node drain tasks automatically fail after the specified timeout expires. A value of 0 indicates that task will not time out.
  - **Forced Drainage**: If this function is enabled, pods managed by DaemonSet will be ignored, and pods with emptyDir volumes and pods not managed by controllers will be deleted. For details, see [Rules for Draining Nodes](#).
- Step 5** Click **OK** and wait until the node drainage is complete.

----End

## Drainage Using kubectl

- Step 1** Use kubectl to access the cluster. For details, see [Connecting to a Cluster Using kubectl](#).
- Step 2** Edit the YAML file for drainage.

The following is an example of **Drainage-test.yaml**:

```
apiVersion: node.cce.io/v1
kind: Drainage
metadata:
  name: 192.168.1.67-1721616409999 # Drainage resource name
spec:
  nodeName: 192.168.1.67 # Kubernetes name of the node to be drained, which can be obtained by
  force: true
  timeout: 0
```

- **nodeName**: node to be drained. The parameter value is the node name in Kubernetes, not the name displayed on the console.

You can run the **kubectl get node** command to obtain a node name in Kubernetes.

- **force**: whether to forcibly drain a node. Value **true** means that the drainage is forced, while **false** means it is not.
- **timeout**: timeout measured in seconds. Node drain tasks automatically fail after the specified timeout expires. A value of 0 indicates that task will not time out.

### Step 3 Create drainage resources.

```
kubectl create -f Drainage-test.yaml
```

If information similar to the following is displayed, the drainage resources have been created:

```
drainage.node.cce.io/192.168.1.67-1721616409999 created
```

### Step 4 Check the result.

```
kubectl get drainages 192.168.1.67-1721616409999 -o yaml
```

If **phase** is **Succeeded**, the operation is successful.

```
apiVersion: node.cce.io/v1
kind: Drainage
metadata:
  creationTimestamp: "2024-07-22T03:12:56Z"
  generation: 1
  name: 192.168.1.67-1721616409999
  resourceVersion: "2683143"
  uid: 3ec131e4-0505-4c88-8255-ef9d0eb02712
spec:
  force: true
  nodeName: 192.168.1.67
  timeout: 0
status:
  conditions:
  - lastTransitionTime: "2024-07-22T03:12:56Z"
    message: start to drain node
    reason: Started
    status: "True"
    type: Started
  - lastTransitionTime: "2024-07-22T03:13:26Z"
    message: node has been drained
    reason: Succeeded
    status: "True"
    type: Finished
  phase: Succeeded
```

----End

## Canceling Node Drainage

This section describes how to stop a node drainage that is currently in progress.

### NOTE

In clusters of v1.23.16-r0, v1.25.11-r0, v1.27.8-r0, v1.28.6-r0, v1.29.2-r0, or later versions, node drainage can be canceled.

This operation will abort drainage on nodes, but workloads that have been evicted from these nodes will not be automatically migrated back.

## Cancellation Through the Console

**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 **Nodes**. On the displayed page, click the **Nodes** tab.

**Step 3** Locate the node that is being drained and click **Cancel Drainage**.

**Step 4** In the displayed dialog box, click **OK**. The node status changes to **Drainage cancelled**. You can click **Enable Scheduling** to restore the node to the schedulable state.

----End

## Cancellation Using kubectl

**Step 1** Use kubectl to access the cluster. For details, see [Connecting to a Cluster Using kubectl](#).

**Step 2** Check drainage resources.

```
kubectl get drainages
```

Command output:

```
NAME                               AGE
192.168.1.67-1721616409999        13s
```

**Step 3** Cancel drainage.

```
kubectl annotate drainages 192.168.1.67-1721616409999 node.cce.io/drainage-disable=true
```

**Step 4** Check the result.

```
kubectl get drainages 192.168.1.67-1721616409999 -o yaml
```

If the command output, **phase** is changed to **Cancelled**.

```
apiVersion: node.cce.io/v1
kind: Drainage
metadata:
  annotations:
    node.cce.io/drainage-disable: "true"
    creationTimestamp: "2024-07-22T03:12:56Z"
    generation: 1
    name: 192.168.1.67-1721616409999
    resourceVersion: "2689858"
    uid: 3ec131e4-0505-4c88-8255-ef9d0eb02712
spec:
  force: true
  nodeName: 192.168.1.67
  timeout: 0
status:
  conditions:
    - lastTransitionTime: "2024-07-22T03:12:56Z"
      message: start to drain node
      reason: Started
      status: "True"
      type: Started
    - lastTransitionTime: "2024-07-22T03:13:26Z"
      message: node has been drained
      reason: Succeeded
      status: "True"
      type: Finished
    - lastTransitionTime: "2024-07-22T03:37:48Z"
      message: node drainage has been cancelled
      reason: Cancelled
      status: "True"
      type: Cancelled
  phase: Cancelled
```

----End

## 3.6.7 Deleting or Unsubscribing from a Node

### Scenario

You can delete a pay-per-use node or unsubscribe from a yearly/monthly-billed node that is not needed from the node list.

Deleting or unsubscribing from a node in a CCE cluster will release the node and services running on the node. Drain the node and back up data before the deletion or unsubscription to prevent services running on the node from being affected.

### Precautions

- Deleting a node will lead to pod migration, which may affect services. Perform this operation during off-peak hours. It is a good practice to drain the node before deletion. For details, see [Draining a Node](#).
- Unexpected risks may occur during the operation. Back up data beforehand.
- It takes about 5 to 10 minutes to unsubscribe from a yearly/monthly node. During this period, the node will be unavailable. After that, the node will be automatically cleared.

### Deleting a Pay-per-Use Node

- 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 **Nodes**. On the displayed page, click the **Nodes** tab.
- Step 3** Locate the target node and choose **More > Delete** in the **Operation** column.
- Step 4** In the **Delete Node** dialog box, enter **DELETE** and click **Yes**.

#### NOTE

- After the node is deleted, workload pods on the node are automatically migrated to other available nodes.
- If the disks and EIPs bound to the node are important resources, unbind them first. Otherwise, they will be deleted with the node.

----End

### Unsubscribing from a Yearly/Monthly Node

- 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 **Nodes**. On the displayed page, click the **Nodes** tab.
- Step 3** Locate the target node and choose **More > Unsubscribe** in the **Operation** column. In the displayed dialog box, drain the node.

After a node is selected for draining, CCE sets the node to be unschedulable so that no new pods can be scheduled to this node and then safely evicts all the pods meeting node draining requirements from the node. The time required for draining depends on the pods on the node. If it takes an excessively long time, go back to the node list and check events.



**Step 4** After you confirm the unsubscription, the order page will be displayed. Confirm the unsubscription type and order amount.

 **NOTE**

- It takes about 5 to 10 minutes to process the unsubscription. During this period, the node will be unavailable.

----End

### 3.6.8 Changing the Billing Mode of a Node to Yearly/Monthly

Nodes in CCE clusters can be billed on a pay-per-use or yearly/monthly basis. A pay-per-use node can be changed to yearly/monthly-billed.

#### Notes and Constraints

- To change the billing mode of a node in a pay-per-use node pool to yearly/monthly, you need to upgrade the cluster to v1.19.16-r40, v1.21.11-r0, v1.23.0-r0, v1.25.4-r0, or later.
- After a node in a pay-per-use node pool is changed to a yearly/monthly node, the node does not support elastic scale-in.

#### Changing the Billing Mode of a Node

**NOTICE**

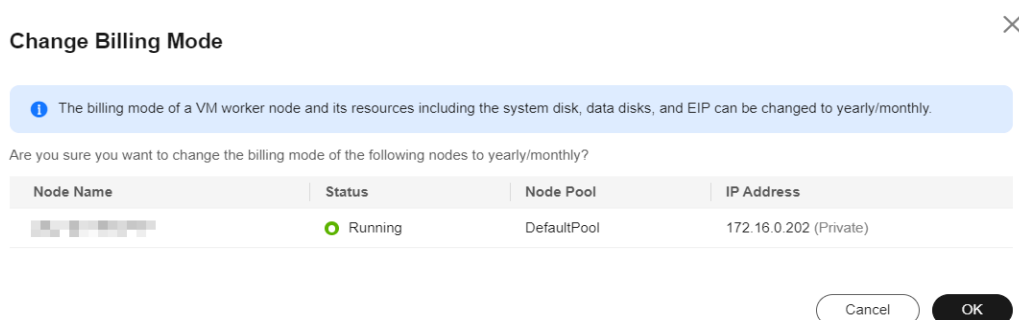
- The billing modes of resources like EVS disks and EIPs used by pay-per-use nodes cannot be changed simultaneously. For details, see [Pay-per-Use to Yearly/Monthly](#).
- To change a pay-per-use node in a node pool to a yearly/monthly one, locate the target node in the node list, choose **More > Forbid node pool scale-in** above the list, and change the billing mode to yearly/monthly.

To change the billing mode of a pay-per-use node to yearly/monthly, perform the following steps:

**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 **Nodes**. Click the **Nodes** tab, locate the row containing the target node, and choose **More > Change Billing Mode** in the **Operation** column.

**Figure 3-4** Changing the billing mode of a node to yearly/monthly



**Step 3** Click **OK**. Wait until the order is processed and the payment is complete.

----End

### 3.6.9 Modifying the Auto-Renewal Configuration of a Yearly/ Monthly Node

After purchasing yearly/monthly nodes, you can enable auto-renewal for them or modify the existing auto-renewal configuration as needed.

#### Enabling Auto-Renewal

**Step 1** Log in to the CCE console and click the cluster name to access the cluster console.

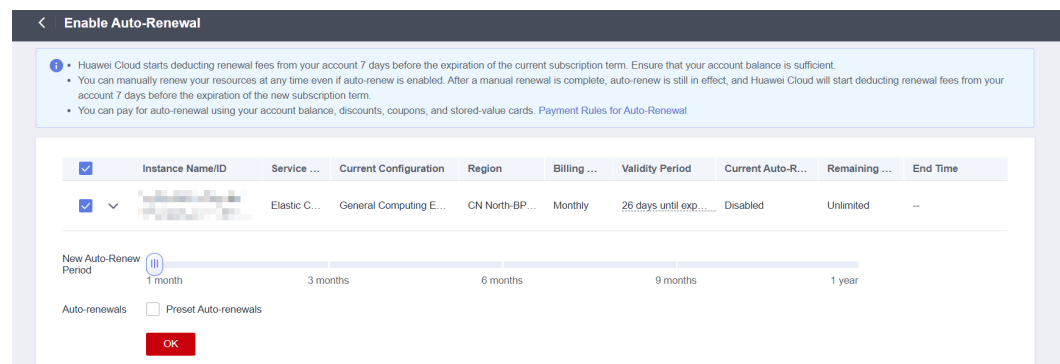
**Step 2** In the navigation pane, choose **Nodes** and switch to the **Nodes** tab page.

**Step 3** Choose **More > Enable Auto-Renewal** in the **Operation** column of a yearly/ monthly node. You can also select multiple nodes at a time. To do so, choose **More > Enable Auto-Renewal** in the function bar on the top of the page. In the displayed dialog box, confirm the nodes for which auto-renewal is to be enabled.

**Step 4** On the **Enable Auto-Renewal** page, configure auto-renewal parameters.

- **New Auto-Renew Period:** Configure the duration of each renewal. The minimum duration of each renewal is one month and the maximum duration is one year.
- **Auto-renewals:** After selecting this option, you can customize the number of renewals. If the number of renewals exceeds the value, auto-renewal will not take effect. If you do not configure the number of auto-renewals, the number of auto-renewals is not limited by default.

**Figure 3-5** Configuring auto-renewal



#### NOTE

If resources such as EIPs are bound to a node, you can select the resources to determine whether to enable auto-renewal with the node.

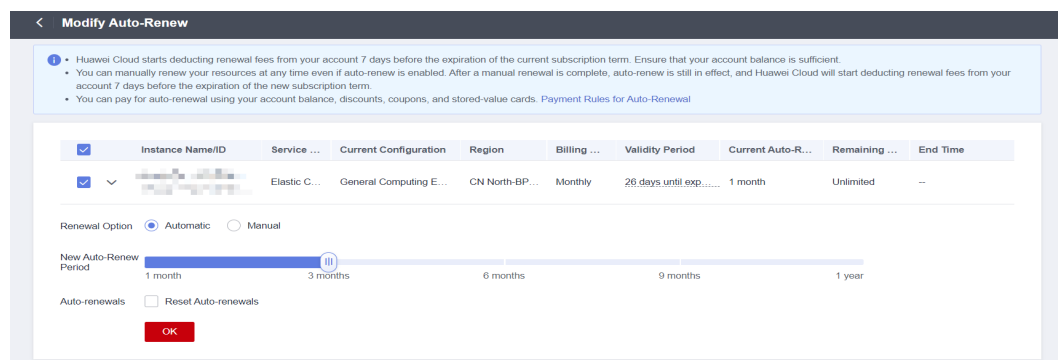
**Step 5** Click **OK**.

----End

## Modifying Auto-Renewal

- 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 **Nodes** and switch to the **Nodes** tab page.
- Step 3** Choose **More > Change Auto-Renewal** in the **Operation** column of a yearly/monthly node. You can also select multiple nodes at a time. To do so, choose **More > Change Auto-Renewal** in the function bar on the top of the page. In the displayed dialog box, confirm the nodes for which auto-renewal is to be modified.
- Step 4** On the **Modify Auto-Renewal** page, modify auto-renewal parameters.
- **Renewal Option:** If auto-renewal is enabled for a node, **Automatic** is selected by default. To disable auto-renewal, select **Manual**.
  - **New Auto-Renew Period:** Configure the duration of each renewal. The minimum duration of each renewal is one month and the maximum duration is one year.
  - **Auto-renewals:** After selecting this option, you can customize the number of renewals. If the number of renewals exceeds the value, auto-renewal will not take effect. If you do not configure the number of auto-renewals, the number of auto-renewals is not limited by default.

Figure 3-6 Modifying auto-renewal



- Step 5** Click **OK**.

----End

### 3.6.10 Stopping a Node

#### Scenario

When a node in the cluster is stopped, all services on that node will also be stopped, and the node will no longer be available for scheduling. Check if your services will be affected before stopping a node.

#### Precautions

- Stopping a node will lead to pod migration, which may affect services. Perform this operation during off-peak hours.
- Unexpected risks may occur during the operation. Back up data beforehand.

## Procedure

- 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 **Nodes**. On the displayed page, click the **Nodes** tab.
- Step 3** Locate the target node and click its name.
- Step 4** In the upper right corner of the ECS details page, click **Stop**. In the displayed dialog box, click **OK**.

----End

## 3.7 Node O&M

### 3.7.1 Node Resource Reservation Policy

Some node resources are used to run mandatory Kubernetes system components and resources to make the node as part of your cluster. Therefore, the total number of node resources and the number of allocatable node resources for your cluster are different. The larger the node specifications, the more the containers deployed on the node. Therefore, more node resources need to be reserved to run Kubernetes components.

To ensure node stability, a certain number of CCE node resources will be reserved for Kubernetes components (such as kubelet, kube-proxy, and docker) based on the node specifications.

CCE calculates the resources that can be allocated to user nodes as follows:

**Allocatable resources = Total amount - Reserved amount - Eviction threshold**

The memory eviction threshold is fixed at 100 MiB.

#### NOTE

**Total amount** indicates the available memory of the ECS, excluding the memory used by system components. Therefore, the total amount is slightly less than the memory of the node flavor.

When the memory consumed by all pods on a node increases, the following behaviors may occur:

1. When the available memory of the node is lower than the eviction threshold, kubelet is triggered to evict the pod. For details about the eviction threshold in Kubernetes, see [Node-pressure Eviction](#).
2. If a node triggers an OS memory insufficiency event (OOM) before kubelet reclaims memory, the system terminates the container. However, different from pod eviction, kubelet restarts the container based on the RestartPolicy of the pod.

## Rules v1 for Reserving Node Memory

### NOTE

For clusters of versions earlier than **v1.21.4-r0** and **v1.23.3-r0**, the v1 model is used for node memory reservation. For clusters of **v1.21.4-r0**, **v1.23.3-r0**, or later, the node memory reservation model is optimized to v2. For details, see [Rules for Reserving Node Memory v2](#).

Upgrading the cluster to **v1.21.4-r0**, **v1.23.3-r0**, or a later version may result in the eviction of the workload on a node that is already fully utilized. This is because system components will need more reserved resources.

You can use the following formula calculate how much memory you should reserve for running containers on a node:

Total reserved amount = **Reserved memory for system components** + **Reserved memory for kubelet to manage pods**

**Table 3-21** Reservation rules for system components

Total Memory (TM)	Reserved Memory for System Components
$TM \leq 8 \text{ GiB}$	0 MiB
$8 \text{ GiB} < TM \leq 16 \text{ GiB}$	$[(TM - 8 \text{ GiB}) \times 1024 \times 10\%]$ MiB
$16 \text{ GiB} < TM \leq 128 \text{ GiB}$	$[8 \text{ GiB} \times 1024 \times 10\% + (TM - 16 \text{ GiB}) \times 1024 \times 6\%]$ MiB
$TM > 128 \text{ GiB}$	$(8 \text{ GiB} \times 1024 \times 10\% + 112 \text{ GiB} \times 1024 \times 6\% + (TM - 128 \text{ GiB}) \times 1024 \times 2\%)$ MiB

**Table 3-22** Reservation rules for kubelet

Total Memory (TM)	Number of Pods	Reserved Memory for kubelet
$TM \leq 2 \text{ GiB}$	None	$TM \times 25\%$
$TM > 2 \text{ GiB}$	$0 < \text{Max. pods on a node} \leq 16$	700 MiB
	$16 < \text{Max. pods on a node} \leq 32$	$[700 + (\text{Max. pods on a node} - 16) \times 18.75]$ MiB
	$32 < \text{Max. pods on a node} \leq 64$	$[1024 + (\text{Max. pods on a node} - 32) \times 6.25]$ MiB
	$64 < \text{Max. pods on a node} \leq 128$	$[1230 + (\text{Max. pods on a node} - 64) \times 7.80]$ MiB
	$\text{Max. pods on a node} > 128$	$[1740 + (\text{Max. pods on a node} - 128) \times 11.20]$ MiB

**NOTICE**

For a small-capacity node, adjust the maximum number of instances based on the site requirements. Alternatively, when creating a node on the CCE console, you can adjust the maximum number of instances for the node based on the node specifications.

## Rules for Reserving Node Memory v2

For clusters of **v1.21.4-r0**, **v1.23.3-r0**, or later, the node memory reservation model is optimized to v2 and can be dynamically adjusted using the node pool parameters **kube-reserved-mem** and **system-reserved-mem**. For details, see [Modifying Node Pool Configurations](#).

The total reserved node memory of the v2 model is equal to the sum of that reserved for the OS and that reserved for CCE to manage pods.

Reserved memory includes basic and floating parts. For the OS, the floating memory depends on the node specifications. For CCE, the floating memory depends on the number of pods on a node.

**Table 3-23** Rules for reserving node memory v2

Reserved for	Basic/Floating	Reservation	Used by
OS	Basic	Fixed at 400 MiB	OS service components such as sshd and systemd-journald.
	Floating (depending on the node memory)	25MiB/GiB	Kernel
CCE	Basic	Fixed at 500 MiB	Container engine components, such as kubelet and kube-proxy, when the node is unloaded
	Floating (depending on the number of pods on the node)	Docker: 20 MiB/Pod containerd: 5 MiB/Pod	Container engine components when the number of pods increases <b>NOTE</b> When the v2 model reserves memory for a node by default, the default maximum number of pods is estimated based on the memory. For details, see <a href="#">Table 3-27</a> .

## Rules for Reserving Node CPU

Table 3-24 Node CPU reservation rules

Total CPU Cores (Total)	Reserved CPU Cores
Total ≤ 1 core	Total x 6%
1 core < Total ≤ 2 cores	1 core x 6% + (Total - 1 core) x 1%
2 cores < Total ≤ 4 cores	1 core x 6% + 1 core x 1% + (Total - 2 cores) x 0.5%
Total > 4 cores	1 core x 6% + 1 core x 1% + 2 cores x 0.5% + (Total - 4 cores) x 0.25%

## Rules for CCE to Reserve Data Disks on Nodes

CCE uses Logical Volume Manager (LVM) to manage disks. LVM creates a metadata area on a disk to store logical and physical volumes, occupying 4 MiB space. Therefore, the actual available disk space of a node is equal to the disk size minus 4 MiB.

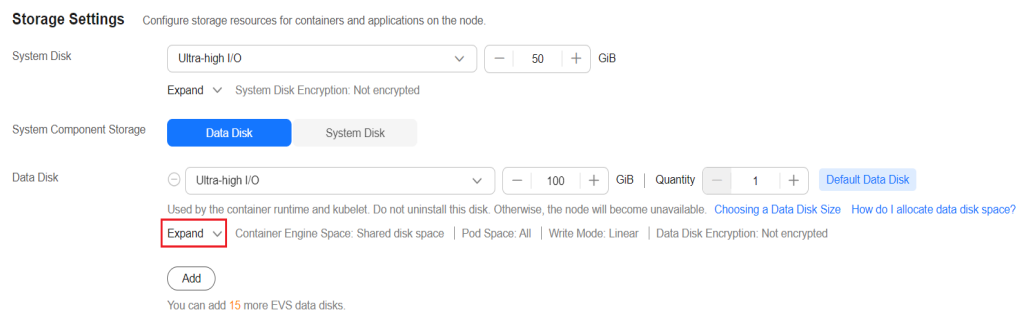
### 3.7.2 Space Allocation of a Data Disk

This section describes how to allocate data disk space to nodes so that you can configure the data disk space accordingly.

#### Allocating Default Data Disk Space

When creating a node, you can customize Data Disk Space Allocation in the expanded area of **Data Disk**.

Figure 3-7 Allocating data disk space



- **Space Allocation for Container Engines**
  - Specified disk space: CCE divides the data disk space for two parts by default. One part is used to store the Docker/containerd working directories, container image data, and image metadata. The other is reserved for kubelet and emptyDir volumes. The available container engine space affects image pulls and container startup and running.

- Container engine and container image space (90% by default): stores the container runtime working directories, container image data, and image metadata.
- kubelet and emptyDir space (10% by default): stores pod configuration files, secrets, and mounted storage such as emptyDir volumes.

#### NOTE

If the sum of the container engine and container image space and the kubelet and emptyDir space is less than 100%, the remaining space will be allocated for user data. You can mount the storage volume to a service path. Do not leave the path empty or set it to a key OS path such as the root directory.

- Shared disk space: In clusters of v1.21.10-r0, v1.23.8-r0, v1.25.3-r0, or later versions, CCE allows [a container engine \(Docker/containerd\) and kubelet components to share data disk space](#).
- **Space Allocation for Pods:** indicates the basesize of a pod. You can set an upper limit for the disk space occupied by each workload pod (including the space occupied by container images). This setting prevents the pods from taking all the disk space available, which may cause 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 details, see [Mapping Between OS and Container Storage Rootfs](#).
- Write Mode
  - **Linear:** A linear logical volume integrates one or more physical volumes. Data is written to the next physical volume when the previous one is used up.
  - **Striped:** available only if there are at least two data disks. A striped logical volume stripes data into blocks of the same size and stores them in multiple physical volumes in sequence. This allows data to be concurrently read and written. A storage pool consisting of striped volumes cannot be scaled-out.

## Space Allocation for Container Engines

For a node using a non-shared data disk (100 GiB for example), the division of the disk space varies depending on the container storage Rootfs type **Device Mapper** or **OverlayFS**. For details about the container storage Rootfs corresponding to different OSs, see [Mapping Between OS and Container Storage Rootfs](#).

- **Rootfs (Device Mapper)**

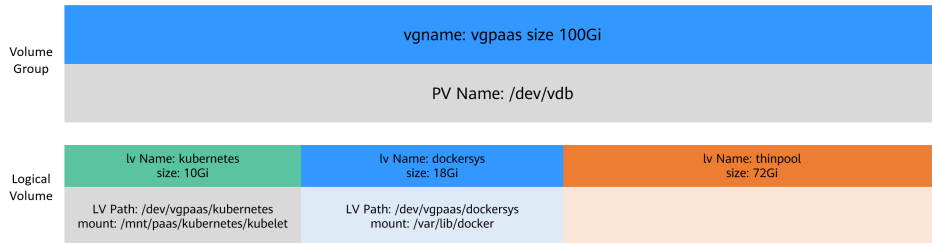
By default, the container engine and image space, occupying 90% of the data disk, can be divided into the following two parts:

- The **/var/lib/docker** directory is used as the Docker working directory and occupies 20% of the container engine and container image space by default. (Space size of the **/var/lib/docker** directory = **Data disk space x 90% x 20%**)
- The thin pool is used to store container image data, image metadata, and container data, and occupies 80% of the container engine and container image space by default. (Thin pool space = **Data disk space x 90% x 80%**)



The thin pool is dynamically mounted. You can view it by running the **lsblk** command on a node, but not the **df -h** command.

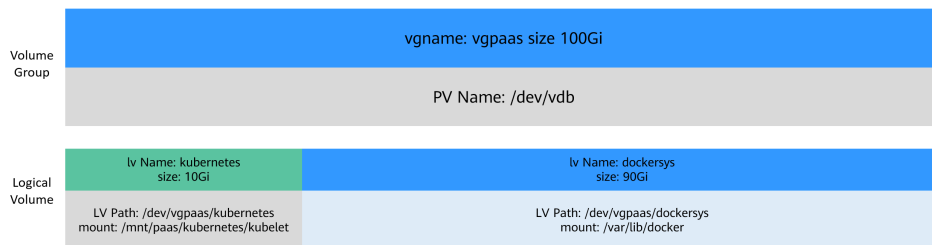
**Figure 3-8** Space allocation for container engines of Device Mapper



- **Rootfs (OverlayFS)**

No separate thin pool. The entire container engine and container image space (90% of the data disk by default) are in the **/var/lib/docker** directory.

**Figure 3-9** Space allocation for container engines of OverlayFS



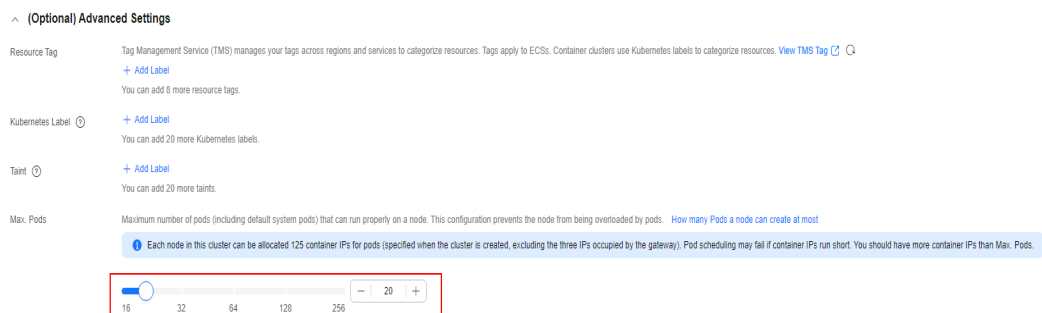
## Space Allocation for Pods

The customized pod container space (basesize) is related to the node OS and container storage Rootfs. For details about the container storage Rootfs, see [Mapping Between OS and Container Storage Rootfs](#).

- Device Mapper supports custom pod basesize. The default value is 10 GiB.
- In OverlayFS mode, the pod container space is not limited by default.

When configuring **basesize**, consider the maximum number of pods allowed on one node. The container engine space should be greater than the total disk space used by containers. Formula: **Container engine space and container image space (90% by default) > Number of containers x basesize**. Otherwise, the container engine space allocated to the node may be insufficient and the container cannot be started.

**Figure 3-10** Configuring the maximum number of pods during node creation



For nodes that support **basesize**, when Device Mapper is used, although you can limit the size of the **/home** directory of a single container (to 10 GiB by default), all containers on the node still share the thin pool of the node for storage. They are not completely isolated. When the sum of the thin pool space used by certain containers reaches the upper limit, other containers cannot run properly.

In addition, after a file is deleted in the **/home** directory of the container, the thin pool space occupied by the file is not released immediately. Therefore, even if **basesize** is set to 10 GiB, the thin pool space occupied by files keeps increasing until 10 GiB when files are created in the container. The space released after file deletion will be reused but after a while. If **the number of containers on the node multiplied by basesize** is greater than the thin pool space size of the node, there is a possibility that the thin pool space has been used up.

## Mapping Between OS and Container Storage Rootfs

**Table 3-25** Node OSs and container engines in CCE clusters

OS	Container Storage Rootfs	Custom Basesize
CentOS 7.x	Clusters of v1.19.16 and earlier use Device Mapper. Clusters of v1.19.16 and later use OverlayFS.	Supported when Rootfs is set to Device Mapper and the runtime is Docker. The default value is 10 GiB. If Rootfs is set to OverlayFS, the basesize cannot be customized.
EulerOS 2.5	Device Mapper	Supported only when the runtime is Docker. The default value is 10 GiB.
EulerOS 2.9	OverlayFS	Supported by Docker clusters of v1.19.16-r0, v1.21.3-r0, v1.23.3-r0, or later. There are no limits by default. Not supported by clusters of a version earlier than v1.19.16-r0, v1.21.3-r0, or v1.23.3-r0.

**Table 3-26** Node OSs and container engines in CCE Turbo clusters

OS	Container Storage Rootfs	Custom Basesize
CentOS 7.x	OverlayFS	Not supported
EulerOS 2.9	OverlayFS	Supported when Rootfs is set to OverlayFS and the runtime is Docker. There are no limits by default.

OS	Container Storage Rootfs	Custom Basesize
EulerOS 2.10	Device Mapper for ECSs (PM)	Supported when Rootfs is set to Device Mapper and the runtime is containerd. The default value is 10 GiB.

## Garbage Collection Policies for Container Images

When the container engine space is insufficient, image garbage collection is triggered.

The policy for garbage collecting images takes two factors into consideration: **HighThresholdPercent** and **LowThresholdPercent**. Disk usage exceeding the high threshold (default: 80%) will trigger garbage collection. The garbage collection will delete least recently used images until the low threshold (default: 70%) is met.

## Recommended Configuration for the Container Engine Space

- The container engine space should be greater than the total disk space used by containers. Formula: **Container engine space > Number of containers x basesize**
- You are advised to create and delete files of containerized services in local storage volumes (such as emptyDir and hostPath volumes) or cloud storage directories mounted to the containers. In this way, the thin pool space is not occupied. emptyDir volumes occupy the kubelet space. Therefore, properly plan the size of the kubelet space.
- You can deploy services on nodes that use the OverlayFS (for details, see [Mapping Between OS and Container Storage Rootfs](#)) so that the disk space occupied by files created or deleted in containers can be released immediately.

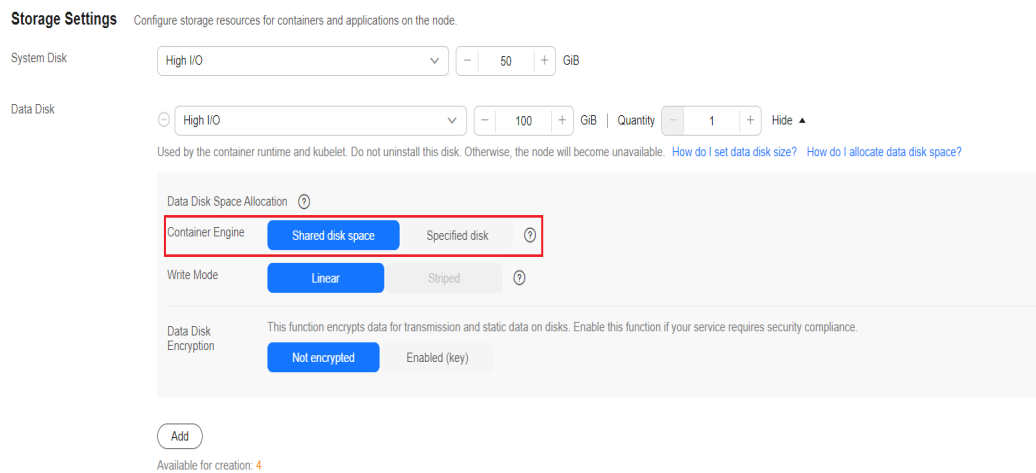
## Data Disk Shared Between a Container Engine and kubelet Components

Docker/containerd and kubelet components share the space of a data disk.

**NOTICE**

- This function is available only to clusters of v1.21.10-r0, v1.23.8-r0, v1.25.3-r0, or later versions.
- If Rootfs is set to OverlayFS, shared data disks are supported. If Rootfs is set to Device Mapper, shared data disks are not supported.
- If you have installed an NPD add-on in the cluster, upgrade the add-on to v1.18.10 or later. Otherwise, false alarms will be generated.
- If you have installed a log-agent add-on in the cluster, upgrade the add-on to v1.3.0 or later. Otherwise, log collection will be affected.
- If you have installed ICAgent in the cluster, upgrade it to v5.12.140 or later. Otherwise, log collection will be affected.

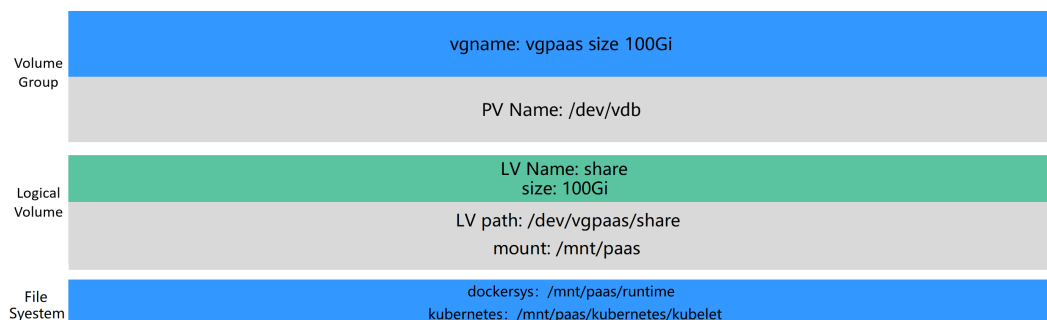
**Figure 3-11** Configuration for sharing disk space



For nodes using a shared data disk, the container storage Rootfs is of the **OverlayFS** type. After such a node is created, the data disk space (for example, 100 GiB) will not be divided for the container engines, container images, and kubelet components. The data disk is mounted to **/mnt/paas**, and the storage space is divided using two file systems.

- dockersys: /mnt/paas/runtime
- Kubernetes: /mnt/paas/kubernetes/kubelet

**Figure 3-12** Allocating the storage space of a shared data disk



### 3.7.3 Maximum Number of Pods That Can Be Created on a Node

#### Calculation of the Maximum Number of Pods on a Node

The maximum number of pods that can be created on a node is calculated based on the cluster type:

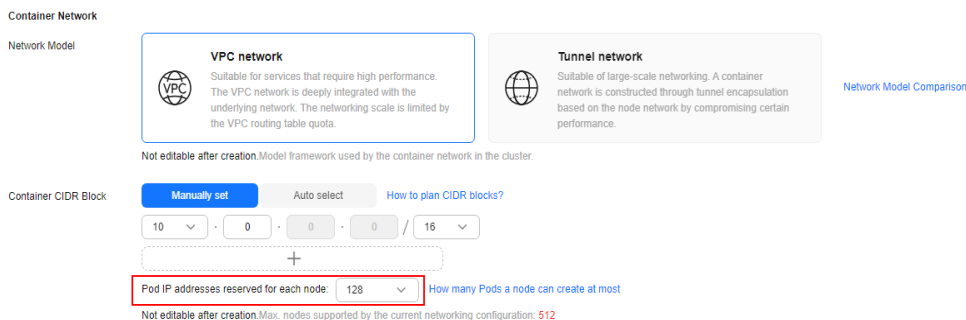
Network Model	Maximum Number of Pods on a Node	Recommended Configuration
Tunnel network	<a href="#">Maximum number of pods on a node</a>	None
VPC network	The smaller value between the <a href="#">maximum number of pods on a node</a> and the <a href="#">number of container IP addresses that can be allocated on a node</a>	To ensure new pods run smoothly on a node, make sure that the number of pods on the node does not exceed the number of container IP addresses that can be assigned to it. If there are not enough container IP addresses available on the node, the new pods may not function properly.
Cloud Native Network 2.0 (for CCE Turbo clusters)	The smaller value between the <a href="#">maximum number of pods on a node</a> and the <a href="#">number of ENIs on a node in a CCE Turbo cluster</a>	To ensure new pods run smoothly on a node, make sure that the number of pods on the node does not exceed the number of ENIs on it. If there are not enough ENIs available on the node, the new pods may not function properly.

#### Number of Allocatable Container IP Addresses on a Node

When creating a cluster in the VPC network model, follow the [rules for managing container IP addresses](#) and specify the number of container IP addresses that can be allocated to each node using `alpha.cce/fixPoolMask`.

The maximum number of pods that can be created on a node is determined by the number of container IP addresses that can be allocated to it. In a [container network](#), each pod needs its own IP address. If there are not enough pre-allocated container IP addresses on the node, pods cannot be created. If `hostNetwork: true` is configured in the YAML file, pods will use the [host network](#) instead of the allocatable container IP addresses. For details, see [Pod IP Address Allocation Differences Between the Container Network and Host Network](#).

**Figure 3-13** Specifying the number of allocatable container IP addresses on a node in the VPC network model



By default, a node occupies three container IP addresses (network address, gateway address, and broadcast address). Therefore, the number of container IP addresses that can be allocated to a node equals the number of selected container IP addresses minus 3. For example, in the preceding figure, the number of container IP addresses that can be allocated to a node is 125 (128 – 3).

## Maximum Number of Pods on a Node

When creating a node, you can configure the maximum number of pods (maxPods) that can be created on the node. This parameter is a configuration item of kubelet and determines the maximum number of pods that can be created by kubelet.

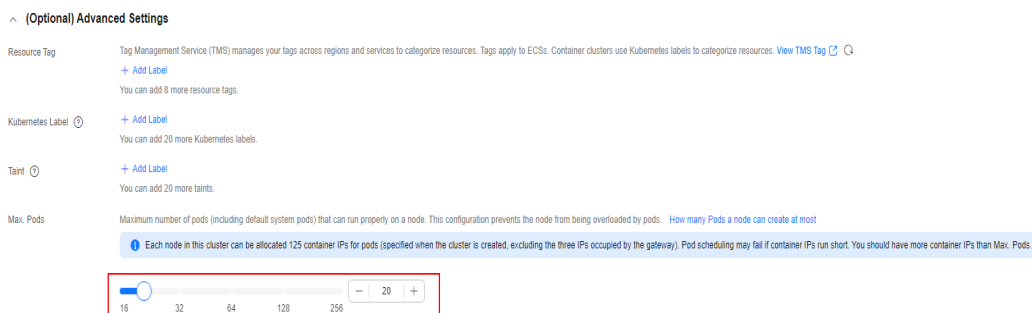
### NOTICE

For nodes in the default node pool (**DefaultPool**), the maximum number of pods cannot be changed after the nodes are created.

After a node in a custom node pool is created, you can modify the **max-pods** parameter in the node pool configuration to change the maximum number of pods on the node.

By default, the maximum number of pods on a node can be adjusted to 256. To increase the deployment density on a node, submit a [service ticket](#) to increase the maximum number of pods on a node, which can be 512.

**Figure 3-14** Configuring the maximum number of pods during node creation



**Table 3-27** lists the default maximum number of pods on a node based on node specifications.

**Table 3-27** Default maximum number of pods on a node

Memory	Max. Pods
4 GB	20
8 GB	40
16 GB	60
32 GB	80
64 GB or above	110

### Number of Node ENIs (Available Only in CCE Turbo Clusters)

In a CCE Turbo cluster, ECS nodes use sub-ENIs. The maximum number of pods that can be created on a node depends on the number of ENIs that can be used by the node.

**Figure 3-15** Node ENIs

Flavor	vCPUs   Memory	Assured/Maximum Bandwidth	Packets Per Second (PPS)	CPU	Max. Pods
<input checked="" type="radio"/> c7.large.2	2 vCPUs   4 GB	0.64 / 4.0 Gbps	400,000 pps	Intel Ice Lake	16
<input type="radio"/> c7.large.4	2 vCPUs   8 GB	0.64 / 4.0 Gbps	400,000 pps	Intel Ice Lake	16
<input type="radio"/> c7.xlarge.2	4 vCPUs   8 GB	1.28 / 8.0 Gbps	800,000 pps	Intel Ice Lake	32
<input type="radio"/> c7.xlarge.4	4 vCPUs   16 GB	1.28 / 8.0 Gbps	800,000 pps	Intel Ice Lake	32

### Pod IP Address Allocation Differences Between the Container Network and Host Network

When creating a pod, you can select the container network or host network for the pod.

- Container network (default): **Each pod is assigned an IP address by the cluster networking add-ons, which occupies the IP addresses of the container network.**
- Host network: Pods with **hostNetwork: true** configured directly use the network of the host. For details, see [Configuring hostNetwork for Pods](#). After the configuration, the pods use the ports on the host and their IP address is identical to that of the host, **without relying on the IP addresses of the container network.** When using a host network, avoid conflicts between the pod ports and service ports on the host. Use a host network only if a particular application requires access to a specific port on the host.

### 3.7.4 Differences in kubelet and Runtime Component Configurations Between CCE and the Native Community

To maintain the stability of nodes, CCE stores Kubernetes and container runtime components on separate data disks. Kubernetes uses the `/mnt/paas/kubernetes`

directory, and the container runtime uses the **/mnt/paas/runtime** directory. The paths to these directories are the same as the default paths used in the community through soft links.

## Differences in Default Paths to kubelet and Runtime

Path Name	Kubernetes Native Path	Path on CCE Nodes
kubelet <b>root-dir</b>	/var/lib/kubelet	/mnt/paas/kubernetes/kubelet
kubelet	/var/lib/kubelet	/mnt/paas/kubernetes/kubelet The soft link from <b>/var/lib/kubelet</b> to <b>/mnt/paas/kubernetes/kubelet</b> is also available.
Container runtime (Docker)	/var/lib/docker	<ul style="list-style-type: none"> <li>• <b>Data Disk Space Allocation</b> set to shared: /mnt/paas/runtime The soft link from <b>/var/lib/docker</b> to <b>/mnt/paas/runtime</b> is also available.</li> <li>• <b>Data Disk Space Allocation</b> set to specific: <b>/var/lib/docker</b>, which is identical to the Kubernetes native path</li> </ul>
Container runtime (containerd)	/var/lib/containerd	<ul style="list-style-type: none"> <li>• <b>Data Disk Space Allocation</b> set to shared: /mnt/paas/runtime The soft link from <b>/var/lib/containerd</b> to <b>/mnt/paas/runtime</b> is also available.</li> <li>• <b>Data Disk Space Allocation</b> set to specific: <b>/var/lib/containerd</b>, which is identical to the Kubernetes native path</li> </ul>
containerd logs	/var/log/pods	/var/lib/containerd/container_logs The soft link from <b>/var/log/pods</b> to <b>/var/lib/containerd/container_logs</b> is also available.



 NOTE

The differences in default paths to kubelet and runtime between CCE and the native community may have the following impacts:

- If a soft link file is mounted into a container, the real path that the soft link file points to cannot be accessed.

For example, if `/var/log` is mounted to `/mnt/log` in a container through a hostPath, `/mnt/log/pods` is an abnormal soft link file in the container, and the actual data in `/var/log/pods` cannot be accessed.

Mount the actual file path to the container to prevent a file read failure caused by soft links.

- The kubelet **root-dir** path (`/mnt/paas/kubernetes/kubelet`) is inconsistent with the community path (`/var/lib/kubelet`). If the container mount path of a third-party CSI add-on is the community path, the file mounting does not take effect.

For example, when Vault (an open-source third-party add-on) uses secrets-store-csi-driver to mount a secret, if the **root-dir** path to the add-on is inconsistent with the CCE configuration path (the default value of the add-on is the same as the community path `/var/lib/kubelet`), the Vault secret cannot be obtained in the container.

This is because CSI uses mountPropagation to mount volumes to target containers. kubelet includes the mount path (`/mnt/paas/kubernetes/kubelet/pods/{podID}/volume/...`) related to kubelet **root-dir** into the CSI request for mounting a volume. When the CSI container mounts a volume to the mount path requested by kubelet CSI, if the mount path is irrelevant to the hostPath, the mount propagation will be invalid. For example, the CSI container mounts host `/var/lib/kubelet` to container `/var/lib/kubelet`, and `/mnt/paas/kubernetes/kubelet/pods/{podID}/volume/...` in the CSI container is irrelevant to the hostPath.

Update the **root-dir** path in CSI to match the kubelet **root-dir** path on the current CCE node.

## 3.7.5 Migrating Nodes from Docker to containerd

Kubernetes has removed dockershim from v1.24 and does not support Docker by default. CCE is going to stop the support for Docker. Change the node container engine from Docker to containerd.

### Prerequisites

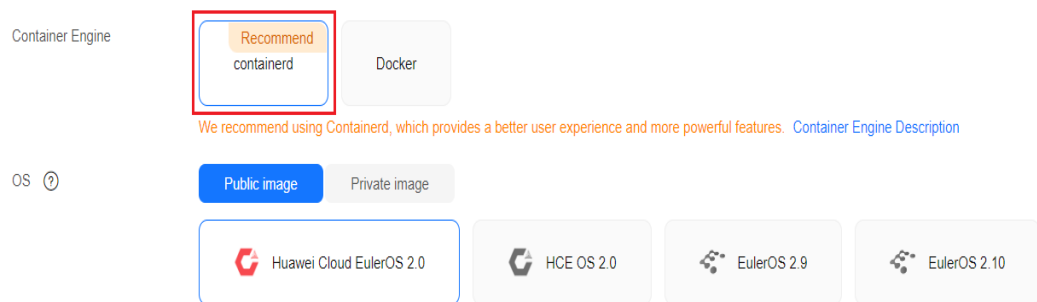
- At least one cluster that supports containerd nodes has been created. For details, see [Mapping Between Node OSs and Container Engines](#).
- There is a Docker node or Docker node pool in your cluster.

### Precautions

- Theoretically, migration during container running will interrupt services for a short period of time. Therefore, it is strongly recommended that the services to be migrated have been deployed as multi-instance. In addition, you are advised to test the migration impact in the test environment to minimize potential risks.
- containerd cannot build images. Do not use the **docker build** command to build images on containerd nodes. For other differences between Docker and containerd, see [Container Engines](#).

## Procedure for Migrating Nodes in the Default Node Pool

- 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 **Nodes**. On the displayed page, click the **Nodes** tab.
- Step 3** In the node list, select one or more nodes to be reset and choose **More > Reset Node** in the **Operation** column.
- Step 4** Set **Container Engine** to **containerd**. You can adjust other parameters as required or retain them as set during creation.



- Step 5** If the node status is **Installing**, the node is being reset.

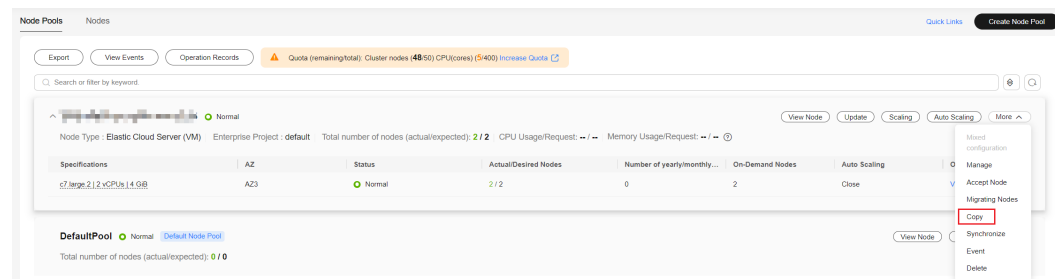
When the node status is **Running**, you can see that the node version is switched to containerd. You can log in to the node and run containerd commands such as **crictl** to view information about the containers running on the node.

----End

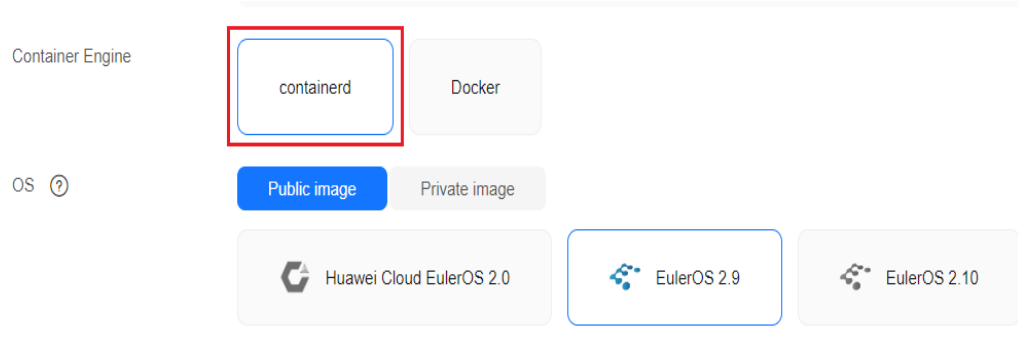
## Procedure for Migrating Nodes in a Custom Node Pool

You can **copy a node pool**, set the container engine of the new node pool to containerd, and keep other configurations the same as those of the original Docker node pool.

- 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 **Nodes**. On the **Node Pools** tab page, locate the Docker node pool to be copied and choose **More > Copy** in the **Operation** column.



- Step 3** In the **Configurations** area, set **Container Engine** to **containerd** and modify other parameter settings as needed to create the node pool.



**Step 4** Scale the number of created containerd node pools to the number of original Docker node pools and delete nodes from the Docker node pools one by one.

Rolling migration is preferred. That is, add some containerd nodes and then delete some Docker nodes until the number of nodes in the new containerd node pool is the same as that in the original Docker node pool.

 **NOTE**

If you have set node affinity for the workloads deployed on the original Docker nodes or node pool, set affinity policies for the workloads to run on the new containerd nodes or node pool.

**Step 5** After the migration, delete the original Docker node pool.

----End

## 3.7.6 Optimizing Node System Parameters

### 3.7.6.1 Optimizable Node System Parameters

CCE provides default node system parameters, which may cause performance bottlenecks in some scenarios. Therefore, you can customize and optimize some node system parameters. [Optimizable Node System Parameters](#) describes the node system parameters.

**NOTICE**

- The modification has certain risks. Be familiar with Linux commands and Linux OS.
- The parameters listed in [Table 3-28](#) have been tested and verified. **Do not modify other parameters.** Otherwise, node faults may occur.
- The commands for modifying node system parameters are valid only when public images are used. The commands provided in this document are for reference only when private images are used.
- After the node is restarted, run the `sysctl -p` command to update the parameter value.

**Table 3-28** System parameters that can be optimized

Parameter	Parameter Location	Description	Reference
RuntimeMaxUse	/etc/systemd/journald.conf	Upper limit of the memory occupied by the node log cache. If this parameter is not set, a large amount of memory will be occupied after the system runs for a long time.  Obtaining the parameter: cat /etc/systemd/journald.conf   grep RuntimeMaxUse	<a href="#">Changing the RuntimeMaxUse of the Memory Used by the Log Cache on a Node</a>
Openfiles	/etc/security/limits.conf	Maximum number of file handles for a single process on a node, which can be adjusted as required.  Obtaining the parameter: ulimit -n	<a href="#">Changing the Maximum Number of File Handles for a Single Process on a Node</a>

Parameter	Parameter Location	Description	Reference
(inside the Openfiles container) LimitNOFILE LimitNPROC	<ul style="list-style-type: none"> <li>• CentOS/ EulerOS:                             <ul style="list-style-type: none"> <li>– Docker nodes: /usr /lib/ systemd/ system/ docker.service</li> <li>– Docker nodes: /usr /lib/ systemd/ system/ containerd.service</li> </ul> </li> <li>• Ubuntu:                             <ul style="list-style-type: none"> <li>– Docker nodes: /lib/ systemd/ system/ docker.service</li> <li>– Docker nodes: /lib/ systemd/ system/ containerd.service</li> </ul> </li> </ul>	<p>Maximum number of file handles for a single process in a container, which can be adjusted as required.</p> <p>Obtaining the parameter:</p> <p>Docker nodes:  <code>cat /proc/`pidof dockerd`/limits   grep files</code></p> <p>containerd nodes:  <code>cat /proc/`pidof containerd`/limits   grep files</code></p>	<a href="#">Changing the Maximum Number of File Handles for a Single Container Process</a>
file-max	/etc/sysctl.conf	<p>Maximum number of file handles in the system, which can be adjusted as required.</p> <p>Obtaining the parameter:  <code>sysctl fs.file-max</code></p>	<a href="#">Changing the Maximum Number of System-Level File Handles on a Node</a>

Parameter	Parameter Location	Description	Reference
nf_conntrack_buckets nf_conntrack_max	/etc/sysctl.conf	Capacity of the connection tracing table, which can be adjusted as required.  Bucket usage = [nf_conntrack_count]/ [nf_conntrack_buckets]  Adjust the buckets value to ensure that the bucket usage is lower than 0.7.  Obtaining the parameter: sysctl net.netfilter.nf_conntrack_count sysctl net.netfilter.nf_conntrack_buckets sysctl net.netfilter.nf_conntrack_max	<a href="#">Modifying Node Kernel Parameters</a>
net.netfilter.nf_conntrack_tcp_timeout_close	/etc/sysctl.conf	Expiration time of the entry of the connection in the close state in the connection tracking table. Shortening the expiration time can speed up the recycling.  Obtaining the parameter: sysctl net.netfilter.nf_conntrack_tcp_timeout_close	
net.netfilter.nf_conntrack_tcp_be_liberal	/etc/sysctl.conf	The parameter value is <b>0</b> or <b>1</b> .  <ul style="list-style-type: none"> <li><b>0</b>: The function is disabled. All RST packets that are not in the TCP window are marked as invalid.</li> <li><b>1</b>: The function is enabled. Only RST packets that are not in the TCP window are marked as invalid. In containers, enabling this parameter can prevent the bandwidth of TCP connections that have been translated using NAT from being limited.</li> </ul> Obtaining the parameter: sysctl net.netfilter.nf_conntrack_tcp_be_liberal	

Parameter	Parameter Location	Description	Reference
tcp_keepalive_time	/etc/sysctl.conf	Interval at which a TCP keepalive message is sent. If this parameter is set to a large value, TCP connections may be suspended in the <b>Close_wait</b> phase for a long time, exhausting system resources.  Obtaining the parameter: sysctl net.ipv4.tcp_keepalive_time	
tcp_max_syn_backlog	/etc/sysctl.conf	Maximum number of TCP half-connections, that is, the maximum number of connections in the <b>SYN_RECV</b> queue.  Obtaining the parameter: sysctl net.ipv4.tcp_max_syn_backlog	
tcp_max_tw_buckets	/etc/sysctl.conf	Specifies the maximum number of sockets in the <b>time-wait</b> state that can exist at any time. If the parameter value is too large, node resources may be exhausted.  Obtaining the parameter: sysctl net.ipv4.tcp_max_tw_buckets	
net.core.somaxconn	/etc/sysctl.conf	Maximum number of TCP connections. This parameter controls the number of TCP connections in a queue. If this parameter is set to a small value, the number of TCP connections is prone to insufficiency. If this parameter is set to a large value, system resources may be wasted because each client waiting for connection in the connection queue occupies certain memory resources.  Obtaining the parameter: sysctl net.core.somaxconn	

Parameter	Parameter Location	Description	Reference
max_user_instances	/etc/sysctl.conf	Maximum number of inotify instances allowed for each user. If the parameter value is too small, the number of inotify instances may be insufficient in containers.  Obtaining the parameter: sysctl fs.inotify.max_user_instances	
max_user_watches	/etc/sysctl.conf	Maximum number of directories of all monitoring instances. If the parameter value is too small, the number of directories may be insufficient in containers.  Obtaining the parameter: sysctl fs.inotify.max_user_watches	
netdev_max_backlog	/etc/sysctl.conf	Size of the packet receiving queue of the network protocol stack. If the parameter value is too small, the queue size may be insufficient.  Obtaining the parameter: sysctl net.core.netdev_max_backlog	
net.core.wmem_max net.core.rmem_max	/etc/sysctl.conf	Memory size (bytes) of the sending and receiving buffer. If this parameter is set to a small value, the memory size may be insufficient in large file scenarios.  Obtaining the parameter: sysctl net.core.wmem_max sysctl net.core.rmem_max	



Parameter	Parameter Location	Description	Reference
net.ipv4.neigh.default.gc_thresh1 net.ipv4.neigh.default.gc_thresh2 net.ipv4.neigh.default.gc_thresh3	/etc/sysctl.conf	Optimization of the garbage collection of ARP entries. <ul style="list-style-type: none"> <li>• <b>gc_thresh1</b>: indicates the minimum number of entries that can be reserved. If the number of entries is less than the <b>gc_thresh1</b> value, the garbage collector (GC) will not reclaim these entries. <b>Do not modify the default parameter setting.</b></li> <li>• <b>gc_thresh2</b>: When the number of entries exceeds the value of this parameter, the GC will clear the entries that have been stored for more than 5 seconds. <b>Do not modify the default parameter setting.</b></li> <li>• <b>gc_thresh3</b>: indicates the maximum number of non-permanent entries. If the system provides a large number of APIs or is directly connected to a large number of devices, increase the value of this parameter.</li> </ul> Obtaining the parameter: <pre>sysctl net.ipv4.neigh.default.gc_thresh1 sysctl net.ipv4.neigh.default.gc_thresh2 sysctl net.ipv4.neigh.default.gc_thresh3</pre>	
vm.max_map_count	/etc/sysctl.conf	If this parameter is set to a small value, a message is displayed indicating that the space is insufficient during ELK installation. Obtaining the parameter: <pre>sysctl vm.max_map_count</pre>	

### 3.7.6.2 Changing the RuntimeMaxUse of the Memory Used by the Log Cache on a Node

Journald is a log system in Linux. It writes log information into binary files and uses `/run/log/journal` as the log cache directory by default. The Journald configuration file is stored in the `/etc/systemd/journald.conf` directory on the node. The `RuntimeMaxUse` parameter indicates the maximum memory usage of the log cache. If `RuntimeMaxUse` is not set, a large amount of memory will be occupied after the system runs for a long time.

#### NOTICE

The commands for modifying node system parameters are valid only when public images are used. The commands provided in this document are for reference only when private images are used.

## Changing RuntimeMaxUse

**Step 1** Log in to the node and view the `/etc/systemd/journald.conf` file.

```
cat /etc/systemd/journald.conf
```

**Step 2** Modify `RuntimeMaxUse`. The recommended value is **100M**.

- If `RuntimeMaxUse` has been set in the `journald.conf` file, run the following command to change the value:

```
sed -i "s/RuntimeMaxUse=[0-9]*M/RuntimeMaxUse=100M/g" /etc/systemd/journald.conf &&  
systemctl restart systemd-journald
```

- If `RuntimeMaxUse` is not set in the `journald.conf` file, run the following command to add it:

```
echo RuntimeMaxUse=100M >> /etc/systemd/journald.conf && systemctl restart systemd-journald
```

**Step 3** If the returned value is the same as the modified value, the modification is correct.

```
cat /etc/systemd/journald.conf | grep RuntimeMaxUse
```

----End

## Automatically Configuring RuntimeMaxUse When Creating a Node or Node Pool

You can set the script to be executed after a node or node pool is created. When creating a node or node pool, you can use the script to configure the `RuntimeMaxUse` size.

**Step 1** Confirm the OS of the node or node pool to be created.

**Step 2** Manually test the script commands on nodes **in the same cluster and running the same OS**. For details about how to manually run the script, see [Changing RuntimeMaxUse](#).

**Step 3** When creating a node or node pool, choose **Advanced Settings > Post-installation Command** to add commands. **(The following commands must be configured after the verification is successful.)**

- Log in to the node and view the `/etc/systemd/journald.conf` file. If **RuntimeMaxUse** has been set, run the following command to change the value:

```
sed -i "s/RuntimeMaxUse=[0-9]*M/RuntimeMaxUse=100M/g" /etc/systemd/journald.conf &&
systemctl restart systemd-journald
```

- Log in to the node and view the `/etc/systemd/journald.conf` file. If **RuntimeMaxUse** is not set, run the following command to add it:

```
echo RuntimeMaxUse=100M >> /etc/systemd/journald.conf && systemctl restart systemd-journald
```

**Step 4** After the node is created, log in to the node to check whether the parameters are successfully modified.

```
cat /etc/systemd/journald.conf | grep RuntimeMaxUse
```

----End

### 3.7.6.3 Changing the Maximum Number of File Handles

The maximum number of file handles is the maximum number of files that can be opened. In Linux, there are two file handle restrictions. The one is system-level restriction, where the maximum number of files that can be opened by all user processes at the same time. The other is user-level restriction, where the maximum number of files that can be opened by a single user process. Containers have the third file handle restriction, that is, the maximum number of file handles of a single process in the container.

#### NOTICE

The commands for modifying node system parameters are valid only when public images are used. The commands provided in this document are for reference only when private images are used.

## Changing the Maximum Number of System-Level File Handles on a Node

**Step 1** Log in to the node and check the `/etc/sysctl.conf` file.

```
cat /etc/sysctl.conf
```

**Step 2** Modify the **fs.file-max** parameter. **fs.file-max=1048576** indicates the kernel parameter name and recommended value.

- If the value of **fs.file-max** has been set in the **sysctl.conf** file, run the following command to change the value:

```
sed -i "s/fs.file-max=[0-9]*$/fs.file-max=1048576/g" /etc/sysctl.conf && sysctl -p
```

- If **fs.file-max** is not set in the **sysctl.conf** file, run the following command to add it:

```
echo fs.file-max=1048576 >> /etc/sysctl.conf && sysctl -p
```

**Step 3** Run the following commands to check whether the change is successful (whether the returned value is the same as that you configure).

```
# sysctl fs.file-max
fs.file-max = 1048576
```

----End

## Changing the Maximum Number of File Handles for a Single Process on a Node

**Step 1** Log in to the node and view the `/etc/security/limits.conf` file.

```
cat /etc/security/limits.conf
```

The maximum number of file handles for a single process of a node is specified by the following parameters:

```
...
root soft nofile 65535
root hard nofile 65535
* soft nofile 65535
* hard nofile 65535
```

**Step 2** Run the `sed` command to change the maximum number of file handles. In the command, **65535** is the recommended maximum number of file handles. The `/etc/security/limits.conf` file on the EulerOS 2.3 node does not contain the default configuration related to `nofile`. Therefore, you cannot run the `sed` command to modify the configuration.

```
sed -i "s/nofile.[0-9]*$/nofile 65535/g" /etc/security/limits.conf
```

**Step 3** **Log in to the node again** and run the following command to check whether the modification is successful. If the returned value is the same as the modified value, the modification is successful.

```
# ulimit -n
65535
```

----End

## Changing the Maximum Number of File Handles for a Single Container Process

**Step 1** Log in to the node and view the `/usr/lib/systemd/system/docker.service` file.

- CentOS/EulerOS:
  - Docker nodes:  

```
cat /usr/lib/systemd/system/docker.service
```
  - containerd nodes:  

```
cat /usr/lib/systemd/system/containerd.service
```
- Ubuntu:
  - Docker nodes:  

```
cat /lib/systemd/system/docker.service
```
  - containerd nodes:  

```
cat /lib/systemd/system/containerd.service
```

### NOTE

If **LimitNOFILE** or **LimitNPROC** is set to **infinity**, the maximum number of file handles supported by a single process of a container is **1,048,576**.

The maximum number of file handles for a single process of a container is specified by the following parameters:

```
...
LimitNOFILE=1048576
LimitNPROC=1048576
...
```

**Step 2** Run the following commands to modify the two parameters. In the command, **1048576** is the recommended value of the maximum number of file handles.

### NOTICE

Changing the maximum number of file handles of a container will restart the docker/containerd process.

- CentOS/EulerOS:
  - Docker nodes:

```
sed -i "s/LimitNOFILE=[0-9a-Z]*$/LimitNOFILE=1048576/g" /usr/lib/systemd/system/docker.service;sed -i "s/LimitNPROC=[0-9a-Z]*$/LimitNPROC=1048576/g" /usr/lib/systemd/system/docker.service && systemctl daemon-reload && systemctl restart docker
```
  - containerd nodes:

```
sed -i "s/LimitNOFILE=[0-9a-Z]*$/LimitNOFILE=1048576/g" /usr/lib/systemd/system/containerd.service;sed -i "s/LimitNPROC=[0-9a-Z]*$/LimitNPROC=1048576/g" /usr/lib/systemd/system/containerd.service && systemctl daemon-reload && systemctl restart containerd
```
- Ubuntu:
  - Docker nodes:

```
sed -i "s/LimitNOFILE=[0-9a-Z]*$/LimitNOFILE=1048576/g" /lib/systemd/system/docker.service;sed -i "s/LimitNPROC=[0-9a-Z]*$/LimitNPROC=1048576/g" /lib/systemd/system/docker.service && systemctl daemon-reload && systemctl restart docker
```
  - containerd nodes:

```
sed -i "s/LimitNOFILE=[0-9a-Z]*$/LimitNOFILE=1048576/g" /usr/lib/systemd/system/containerd.service;sed -i "s/LimitNPROC=[0-9a-Z]*$/LimitNPROC=1048576/g" /usr/lib/systemd/system/containerd.service && systemctl daemon-reload && systemctl restart containerd
```

**Step 3** Check the maximum number of file handles of a single process in the container. If the returned value is the same as the modified value, the modification is successful.

- Docker nodes:

```
# cat /proc/`pidof dockerd`/limits | grep files
Max open files      1048576      1048576      files
```
- containerd nodes:

```
# cat /proc/`pidof containerd`/limits | grep files
Max open files      1048576      1048576      files
```

----End

## Automatically Configuring the Maximum Number of File Handles When Creating a Node or Node Pool

You can set the script to be executed after a node or node pool is created. When creating a node or node pool, you can use the script to configure the maximum number of file handles.

**Step 1** Confirm the OS of the node or node pool to be created.

**Step 2** Manually test the script commands on nodes **in the same cluster and running the same OS**.

- [Changing the Maximum Number of System-Level File Handles on a Node](#)
- [Changing the Maximum Number of File Handles for a Single Process on a Node](#)

- [Changing the Maximum Number of File Handles for a Single Container Process](#)

**Step 3** When creating a node or node pool, choose **Advanced Settings > Post-installation Command** to add commands. **(The following commands must be configured after the verification is successful.)**

- Change the maximum number of system-level file handles on a node.
  - Log in to the node and check the `/etc/sysctl.conf` file. If the value of `fs.file-max` has been set in the file, run the following command to change it:  

```
sed -i "s/fs.file-max=[0-9]*$/fs.file-max=1048576/g" /etc/sysctl.conf && sysctl -p
```
  - Log in to the node and check the `/etc/sysctl.conf` file. If the value of `fs.file-max` is not set in the file, run the following command to add it:  

```
echo fs.file-max=1048576 >> /etc/sysctl.conf && sysctl -p
```

In the preceding command, `fs.file-max=1048576` indicates the kernel parameter name and recommended value.

- Run the following command to change the maximum number of file handles for a single process on a node:  

```
sed -i "s/nofile.[0-9]*$/nofile 65535/g" /etc/security/limits.conf
```

In the preceding command, `65535` is the recommended maximum number of file handles.

- Change the maximum number of file handles for a single process of a container.
  - **CentOS/EulerOS:**  

```
sed -i "s/LimitNOFILE=[0-9a-Z]*$/LimitNOFILE=1048576/g" /usr/lib/systemd/system/docker.service;sed -i "s/LimitNPROC=[0-9a-Z]*$/LimitNPROC=1048576/g" /usr/lib/systemd/system/docker.service && systemctl daemon-reload && systemctl restart docker
```
  - **Ubuntu:**  

```
sed -i "s/LimitNOFILE=[0-9a-Z]*$/LimitNOFILE=1048576/g" /lib/systemd/system/docker.service;sed -i "s/LimitNPROC=[0-9a-Z]*$/LimitNPROC=1048576/g" /lib/systemd/system/docker.service && systemctl daemon-reload && systemctl restart docker
```

In the preceding command, `1048576` is the recommended maximum number of file handles.

The command in the following figure is used only as an example. Change it as required.

The screenshot shows a configuration form with the following fields:

- ECS Group:** A dropdown menu with 'Anti-affinity' selected and a help icon.
- Pre-installation Command:** A text area containing the text: "Command executed before Kubernetes software is installed. Executing this command may cause the installation to fail. It is commonly used to format data disks." with a character count of 0/1,000.
- Post-installation Command:** A text area containing the command: `echo fs.file-max=1048576 >> /etc/sysctl.conf && sysctl -p` with a character count of 57/1,000.
- Agency:** A dropdown menu with '--Select--' selected and a help icon.

**Step 4** After the node is created, log in to the node to check whether the parameters are successfully modified.

----End

### 3.7.6.4 Modifying Node Kernel Parameters

The default Linux kernel parameters may not satisfy all users. You can modify the `/etc/sysctl.conf` configuration file on the node to modify the kernel parameters.

#### NOTICE

- The commands for modifying node system parameters are valid only when public images are used. The commands provided in this document are for reference only when private images are used.
- After the node is restarted, run the `sysctl -p` command to update the parameter value.

**Table 3-29** Kernel parameters of a node

Parameter	Parameter Location	Description	Recommended Value
file-max	/etc/sysctl.conf	Maximum number of file handles in the system, which can be adjusted as required.  Obtaining the parameter: sysctl fs.file-max	fs.file-max=1048576
nf_conntrack_buckets nf_conntrack_max	/etc/sysctl.conf	Capacity of the connection tracing table, which can be adjusted as required.  Bucket usage = [nf_conntrack_count]/ [nf_conntrack_buckets]  If the CPU usage is greater than 0.7 for a long time, increase the value of buckets to lower the CPU usage to less than 0.7.  Obtaining the parameter: sysctl net.netfilter.nf_conntrack_count sysctl net.netfilter.nf_conntrack_buckets sysctl net.netfilter.nf_conntrack_max	The default value is set based on the node memory size. To change the value, refer to the following formula:  • net.netfilter.nf_conntrack_buckets = [nf_conntrack_count] / 0.7  • net.netfilter.nf_conntrack_max = 4* [nf_conntrack_buckets]

Parameter	Parameter Location	Description	Recommended Value
net.netfilter.nf_conntrack_tcp_timeout_close	/etc/sysctl.conf	Expiration time of the entry of the connection in the close state in the connection tracking table. Shortening the expiration time can speed up the recycling.  Obtaining the parameter: sysctl net.netfilter.nf_conntrack_tcp_timeout_close	net.netfilter.nf_conntrack_tcp_timeout_close=3
net.netfilter.nf_conntrack_tcp_be_liberal	/etc/sysctl.conf	The parameter value is <b>0</b> or <b>1</b> . <ul style="list-style-type: none"> <li><b>0</b>: The function is disabled. All RST packets that are not in the TCP window are marked as invalid.</li> <li><b>1</b>: The function is enabled. Only RST packets that are not in the TCP window are marked as invalid. In containers, enabling this parameter can prevent the bandwidth of TCP connections that have been translated using NAT from being limited.</li> </ul> Obtaining the parameter: sysctl net.netfilter.nf_conntrack_tcp_be_liberal	net.netfilter.nf_conntrack_tcp_be_liberal=1
tcp_keepalive_time	/etc/sysctl.conf	Interval for sending keepalive detection messages through TCP. If this parameter is set to a large value, TCP connections may be suspended in the <b>Close_wait</b> phase for a long time, which exhausts system resources.  Obtaining the parameter: sysctl net.ipv4.tcp_keepalive_time	net.ipv4.tcp_keepalive_time=600



Parameter	Parameter Location	Description	Recommended Value
tcp_max_syn_backlog	/etc/sysctl.conf	Maximum number of TCP half-connections, that is, the maximum number of connections in the <b>SYN_RECV</b> queue.  Obtaining the parameter: sysctl net.ipv4.tcp_max_syn_backlog	net.ipv4.tcp_max_syn_backlog=8096
tcp_max_tw_buckets	/etc/sysctl.conf	Specifies the maximum number of sockets in the time-wait state that can exist at any time. If the parameter value is too large, node resources may be exhausted.  Obtaining the parameter: sysctl net.ipv4.tcp_max_tw_buckets	net.ipv4.tcp_max_tw_buckets=5000
net.core.somaxconn	/etc/sysctl.conf	Maximum number of TCP connections and maximum size of the ESTABLISHED queue. If the parameter value is too small, the value may be insufficient.  Obtaining the parameter: sysctl net.core.somaxconn	net.core.somaxconn=32768
max_user_instances	/etc/sysctl.conf	Maximum number of inotify instances allowed for each user. If the parameter value is too small, the number of inotify instances may be insufficient in containers.  Obtaining the parameter: sysctl fs.inotify.max_user_instances	fs.inotify.max_user_instances=8192
max_user_watches	/etc/sysctl.conf	Maximum number of directories of all monitoring instances. If the parameter value is too small, the number of directories may be insufficient in containers.  Obtaining the parameter: sysctl fs.inotify.max_user_watches	fs.inotify.max_user_watches=524288

Parameter	Parameter Location	Description	Recommended Value
netdev_max_backlog	/etc/sysctl.conf	Size of the packet receiving queue of the network protocol stack. If the parameter value is too small, the queue size may be insufficient.  Obtaining the parameter: sysctl net.core.netdev_max_backlog	net.core.netdev_max_backlog=16384
net.core.wmem_max net.core.rmem_max	/etc/sysctl.conf	Memory size (bytes) of the sending and receiving buffer. If this parameter is set to a small value, the memory size may be insufficient in large file scenarios.  Obtaining the parameter: sysctl net.core.wmem_max sysctl net.core.rmem_max	net.core.wmem_max=16777216 net.core.rmem_max=16777216

Parameter	Parameter Location	Description	Recommended Value
net.ipv4.neigh.default.gc_thresh1 net.ipv4.neigh.default.gc_thresh2 net.ipv4.neigh.default.gc_thresh3	/etc/sysctl.conf	Garbage collection optimization of ARP entries. <ul style="list-style-type: none"> <li> <b>gc_thresh1</b>: indicates the minimum number of entries that can be reserved. If the number of entries is less than the <b>gc_thresh1</b> value, the garbage collector (GC) will not reclaim these entries. <b>Do not modify the default parameter setting.</b> </li> <li> <b>gc_thresh2</b>: When the number of entries exceeds the value of this parameter, the GC will clear the entries that have been stored for more than 5 seconds. <b>Do not modify the default parameter setting.</b> </li> <li> <b>gc_thresh3</b>: indicates the maximum number of non-permanent entries. If the system provides a large number of APIs or is directly connected to a large number of devices, increase the value of this parameter.                             </li> </ul> Obtaining the parameter: <pre>sysctl net.ipv4.neigh.default.gc_thresh1 sysctl net.ipv4.neigh.default.gc_thresh2 sysctl net.ipv4.neigh.default.gc_thresh3</pre>	net.ipv4.neigh.default.gc_thresh1=0 net.ipv4.neigh.default.gc_thresh2=4096 net.ipv4.neigh.default.gc_thresh3=163790
vm.max_map_count	/etc/sysctl.conf	If this parameter is set to a small value, a message is displayed indicating that the space is insufficient during ELK installation.  Obtaining the parameter: <pre>sysctl vm.max_map_count</pre>	vm.max_map_count=262144

## Modifying Node Kernel Parameters

**Table 3-29** lists the kernel parameters of nodes. The following describes how to change the value of `net.ipv4.tcp_keepalive_time`, which indicates the interval for sending keepalive detection messages over TCP.

**Step 1** Log in to the node and check the `/etc/sysctl.conf` file.

```
cat /etc/sysctl.conf
```

**Step 2** Modify the `net.ipv4.tcp_keepalive_time` parameter.

`net.ipv4.tcp_keepalive_time=600` indicates the kernel parameter name and recommended value. For details about the recommended value, see **Table 3-29**.

To modify other kernel parameters, replace the parameter names and values in the commands by referring to **Table 3-29**.

- If `net.ipv4.tcp_keepalive_time` has been set in the `sysctl.conf` file, run the following command to change the value:  

```
sed -i "s/net.ipv4.tcp_keepalive_time=[0-9]*$/net.ipv4.tcp_keepalive_time=600/g" /etc/sysctl.conf && sysctl -p
```
- If `net.ipv4.tcp_keepalive_time` is not set in the `sysctl.conf` file, run the following command to add it:  

```
echo net.ipv4.tcp_keepalive_time=600 >> /etc/sysctl.conf && sysctl -p
```

**Step 3** Run the command in **Table 3-29** to check whether the modification is successful. If the returned value is the same as the modified value, the modification is successful.

```
# sysctl net.ipv4.tcp_keepalive_time
net.ipv4.tcp_keepalive_time = 600
```

----End

## Automatically Configuring Kernel Parameters When Creating a Node or Node Pool

You can set the script to be executed after a node or node pool is created. When creating a node or node pool, you can use the script to configure kernel parameters.

The `net.ipv4.tcp_keepalive_time` parameter is used as an example to describe how to change the interval for sending keepalive detection messages over TCP. The value is the recommended value in **Table 3-29**.

**Step 1** Confirm the OS of the node or node pool to be created, for example, CentOS 7.6.

**Step 2** Manually test the script commands on nodes **in the same cluster and running the same OS**. For details about how to manually run the script, see **Modifying Node Kernel Parameters**.

**Step 3** When creating a node or node pool, choose **Advanced Settings > Post-installation Command** to add commands. **(The following commands must be configured after the verification is successful.)** To modify other kernel parameters, replace the parameter names and values in the commands by referring to **Table 3-29**.

- Log in to the node and check the `/etc/sysctl.conf` file. If `net.ipv4.tcp_keepalive_time` has been set in the file, run the following command to change it:

```
sed -i "s/net.ipv4.tcp_keepalive_time=[0-9]*$/net.ipv4.tcp_keepalive_time=600/g" /etc/sysctl.conf && sysctl -p
```

- Log in to the node and check the `/etc/sysctl.conf` file. If `net.ipv4.tcp_keepalive_time` is not set in the file, run the following command to add it:

```
echo net.ipv4.tcp_keepalive_time=600 >> /etc/sysctl.conf && sysctl -p
```

The command in the following figure is used only as an example. Change it as required.

The screenshot shows a configuration interface with the following fields:

- ECS Group:** A dropdown menu with 'Anti-affinity' selected and a help icon.
- Pre-installation Command:** A text area containing the text: "Command executed before Kubernetes software is installed. Executing this command may cause the installation to fail. It is commonly used to format data disks." with a character count of 0/1,000.
- Post-installation Command:** A text area containing the command: `echo net.ipv4.tcp_keepalive_time=600 >> /etc/sysctl.conf && sysctl -p` with a character count of 69/1,000.
- Agency:** A dropdown menu with '--Select--' selected and a help icon.

- Step 4** After the node is created, log in to the node and run the command in [Table 3-29](#) to check whether the modification is successful.

----End

### 3.7.6.5 Changing Process ID Limits (`kernel.pid_max`)

#### Context

Process IDs (PIDs) are a fundamental resource on nodes. It is trivial to hit the task limit without hitting any other resource limits, which can then cause instability to a host machine.

You can adjust the PID limit (`kernel.pid_max`) according to service requirements.

#### `kernel.pid_max` Defaults

Starting from January 2022, CCE changes the default value of `kernel.pid_max` to **4194304** for EulerOS 2.5, CentOS 7.6, and Ubuntu 18.04 nodes in clusters of v1.17 or later. Specific conditions:

- Cluster version: v1.17.17 or later
- Node creation: after January 30, 2022

If the preceding two conditions are not met, `kernel.pid_max` on EulerOS 2.5, CentOS 7.6, and Ubuntu 18.04 nodes defaults to **32768**.

**Table 3-30** kernel.pid\_max defaults

OS	Clusters of 1.17.9 and Earlier	Clusters of 1.17.17 and Later	
		Nodes Created on or Before January 30, 2022	Nodes Created After January 30, 2022
EulerOS 2.5	32768	32768	4194304
CentOS 7.6	32768	32768	4194304
Ubuntu 18.04	N/A	32768	4194304
EulerOS 2.3	57344	57344	57344
EulerOS 2.9	N/A	4194304	4194304

### Change Suggestion

- EulerOS 2.3: Change the default to **4194304** for all nodes. For details, see [Changing kernel.pid\\_max of a Node](#). Use a pre-installation script to do so for new nodes and node pools. For details, see [Configuring kernel.pid\\_max When Creating a Node Pool](#) or [Configuring kernel.pid\\_max When Creating a Node](#).
- EulerOS 2.5, CentOS 7.6, and Ubuntu 18.04:
  - Change the value of **kernel.pid\_max** to **4194304** for nodes created on January 30, 2022 or earlier in clusters of v1.17.17 or later. For details, see [Changing kernel.pid\\_max of a Node](#).
  - For clusters of 1.17.9 and earlier:
    - Change the value of **kernel.pid\_max** to **4194304** for existing nodes. For details, see [Changing kernel.pid\\_max of a Node](#).
    - Use a pre-installation script to do so for new nodes and node pools. For details, see [Configuring kernel.pid\\_max When Creating a Node Pool](#) or [Configuring kernel.pid\\_max When Creating a Node](#).

### Viewing kernel.pid\_max

Log in to the node and run the following command to obtain the value of **kernel.pid\_max**:

```
sysctl kernel.pid_max
```

```
# sysctl kernel.pid_max
kernel.pid_max = 32768
```

Change **kernel.pid\_max**, if necessary, as instructed in [Changing kernel.pid\\_max of a Node](#).

### Checking Node PIDs

Log in to the node and run the following command to check how many PIDs are in use:

```
ps -efLL | wc -l
```

```
# ps -efLL | wc -l
691
```

## Changing kernel.pid\_max of a Node

Log in to the node and run the following command. **4194304** indicates the value of **kernel.pid\_max** and is used as an example here.

```
echo kernel.pid_max = 4194304 >> /etc/sysctl.conf && sysctl -p
```

```
echo 4194304 > /sys/fs/cgroup/pids/kubepods/pids.max
```

Run the following commands to check whether the returned value is the same as that you configured:

```
# sysctl kernel.pid_max
kernel.pid_max = 4194304
# cat /sys/fs/cgroup/pids/kubepods/pids.max
4194304
```

## Configuring kernel.pid\_max When Creating a Node Pool

EulerOS 2.3: Configuration required.

EulerOS 2.5, CentOS 7.6, and Ubuntu 18.04: **Configuration required** for clusters of v1.17.9 and earlier. **Configuration NOT required** for clusters of v1.17.17 and later because the value has been changed.

You can configure **kernel.pid\_max** in the pre-installation script to create a node from a node pool.

When creating a node pool, choose **Advanced Settings > Post-installation Command** and add the following command:

```
echo kernel.pid_max = 4194304 >> /etc/sysctl.conf && sysctl -p
```

ECS Group	<div style="border: 1px solid #ccc; padding: 5px;"> <span style="background-color: #4a7ebb; color: white; padding: 2px 5px;">Anti-affinity</span> <span style="float: right;">?</span> </div> <div style="border: 1px solid #ccc; padding: 5px; margin-top: 5px;"> <span style="background-color: #f0f0f0; display: block; text-align: center;">--Select--</span> </div> <div style="text-align: right; margin-top: 5px;"> <span style="color: #4a7ebb;">+</span> Add ECS Group <a href="#">↗</a> </div>
Pre-installation Command	<div style="border: 1px solid #ccc; padding: 5px; min-height: 40px;"> <p style="font-size: 0.9em; color: #666;">Command executed before Kubernetes software is installed. Executing this command may cause the installation to fail. It is commonly used to format data disks.</p> </div>
Post-installation Command	<div style="border: 1px solid #ccc; padding: 5px; min-height: 40px;"> <pre>echo kernel.pid_max = 4194304 &gt;&gt; /etc/sysctl.conf &amp;&amp; sysctl -p</pre> </div>

## Configuring kernel.pid\_max When Creating a Node

EulerOS 2.3: Configuration required.

EulerOS 2.5, CentOS 7.6, and Ubuntu 18.04: **Configuration required** for clusters of v1.17.9 and earlier. **Configuration NOT required** for clusters of v1.17.17 and later because the value has been changed.

You can configure **kernel.pid\_max** by using the pre-installation script when creating a node.

Choose **Advanced Settings > Post-installation Command** and add the following command:

```
echo kernel.pid_max = 4194304 >> /etc/sysctl.conf && sysctl -p
```

ECS Group	<b>Anti-affinity</b> ⓘ
	<input type="text" value="--Select--"/> <a href="#">Add ECS Group</a>
Pre-installation Command	Command executed before Kubernetes software is installed. Executing this command may cause the installation to fail. It is commonly used to format data disks.
Post-installation Command	<pre>echo kernel.pid_max = 4194304 &gt;&gt; /etc/sysctl.conf &amp;&amp; sysctl -p</pre>

### 3.7.7 Configuring Node Fault Detection Policies

The node fault detection function depends on the **NPD** add-on. The add-on instances run on nodes and monitor nodes. This section describes how to enable node fault detection.

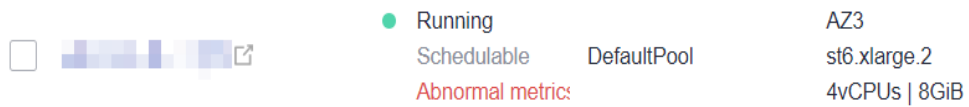
#### Prerequisites

The **CCE Node Problem Detector** add-on has been installed in the cluster.

#### Enabling Node Fault Detection

- 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 **Nodes** and then click the **Nodes** tab. Check whether the NPD add-on has been installed in the cluster or whether the add-on has been upgraded to the latest version. After the NPD add-on has been installed, you can use the fault detection function.
- Step 3** If the NPD add-on is running properly, click **Node Fault Detection Policy** to view the current fault detection items. For details about the NPD check item list, see **NPD Check Items**.
- Step 4** If the check result of the current node is abnormal, a message is displayed in the node list, indicating that the metric is abnormal.





**Step 5** You can click **Abnormal metrics** and rectify the fault as prompted.

### Abnormal metrics ( [redacted] )

#### The Kubelet service is abnormal.

Exception Source	NPD
Exception Occurrence Time	May 08, 2023 14:13:26 GMT+08:00
Last Heartbeat Time	May 08, 2023 14:13:26 GMT+08:00
Exception Information	kubelet:kubelet was found unhealthy; repair flag : false
Handling Suggestion	A Kubelet exception is detected. Generally, the Kubelet configuration is abnormal. Log in to the node and check the Kubelet startup logs.

----End

## Customized Check Items

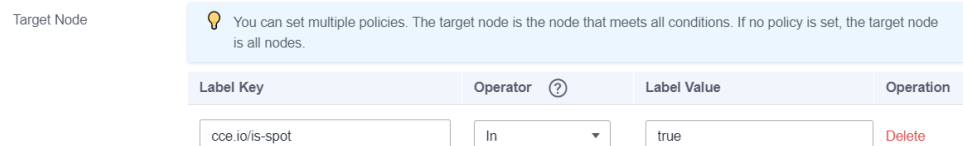
**Step 1** Log in to the CCE console and click the cluster name to access the cluster console.

**Step 2** In the navigation pane, choose **Nodes** and then click the **Nodes** tab. Then, click **Fault Detection Policy**.

**Step 3** On the displayed page, view the current check items. Click **Edit** in the **Operation** column and edit checks.

Currently, the following configurations are supported:

- **Enable/Disable:** Enable or disable a check item.
- **Target Node:** By default, check items run on all nodes. You can change the fault threshold based on special scenarios. For example, the spot price ECS interruption reclamation check runs only on the spot price ECS node.



- **Trigger Threshold:** The default thresholds match common fault scenarios. You can customize and modify the fault thresholds as required. For example, change the threshold for triggering connection tracking table exhaustion from 90% to 80%.

Trigger Threshold  Times  %

If the resource usage reaches 80% for 1 consecutive times, this check item is considered faulty and the fault handling policy is triggered.

- **Check Period:** The default check period is 30 seconds. You can modify this parameter as required.

Check Period  Second

- **Troubleshooting Strategy:** After a fault occurs, you can select the strategies listed in the following table.

**Table 3-31** Troubleshooting strategies

Troubleshooting Strategy	Effect
Prompting Exception	Kubernetes events are reported.
Disabling scheduling	Kubernetes events are reported and the <b>NoSchedule</b> taint is added to the node.
Evict Node Load	Kubernetes events are reported and the <b>NoExecute</b> taint is added to the node. This operation will evict workloads on the node and interrupt services. Exercise caution when performing this operation.

----End

## NPD Check Items

### NOTE

Check items are supported only in 1.16.0 and later versions.

Check items cover events and statuses.

- **Event-related**  
For event-related check items, when a problem occurs, NPD reports an event to the API server. The event type can be **Normal** (normal event) or **Warning** (abnormal event).

**Table 3-32** Event-related check items

Check Item	Function	Description
OOMKilling	<p>Listen to the kernel logs and check whether OOM events occur and are reported.</p> <p>Typical scenario: When the memory usage of a process in a container exceeds the limit, OOM is triggered and the process is terminated.</p>	<p>Warning event</p> <p>Listening object: <b>/dev/kmsg</b></p> <p>Matching rule: "Killed process \\d+ (.+) total-vm:\\d+kB, anon-rss:\\d+kB, file-rss:\\d+kB.*"</p>
TaskHung	<p>Listen to the kernel logs and check whether taskHung events occur and are reported.</p> <p>Typical scenario: Disk I/O suspension causes process suspension.</p>	<p>Warning event</p> <p>Listening object: <b>/dev/kmsg</b></p> <p>Matching rule: "task \\S+:\\w+ blocked for more than \\w+ seconds\\."</p>
ReadOnlyFilesystem	<p>Check whether the <b>Remount root filesystem read-only</b> error occurs in the system kernel by listening to the kernel logs.</p> <p>Typical scenario: A user detaches a data disk from a node by mistake on the ECS, and applications continuously write data to the mount point of the data disk. As a result, an I/O error occurs in the kernel and the disk is remounted as a read-only disk.</p> <p><b>NOTE</b> If the rootfs of node pods is of the device mapper type, an error will occur in the thin pool if a data disk is detached. This will affect NPD and NPD will not be able to detect node faults.</p>	<p>Warning event</p> <p>Listening object: <b>/dev/kmsg</b></p> <p>Matching rule: <b>Remounting filesystem read-only</b></p>

- Status-related

For status-related check items, when a problem occurs, NPD reports an event to the API server and changes the node status synchronously. This function can be used together with [Node-problem-controller fault isolation](#) to isolate nodes.

**If the check period is not specified in the following check items, the default period is 30 seconds.**

**Table 3-33** Checking system components

Check Item	Function	Description
Container network component error CNIPProblem	Check the status of the CNI components (container network components).	None
Container runtime component error CRIPProblem	Check the status of Docker and containerd of the CRI components (container runtime components).	Check object: Docker or containerd
Frequent restarts of Kubelet FrequentKubeletRestart	Periodically backtrack system logs to check whether the key component Kubelet restarts frequently.	<ul style="list-style-type: none"> <li>• Default threshold: 10 restarts within 10 minutes</li> <li>If Kubelet restarts for 10 times within 10 minutes, it indicates that the system restarts frequently and a fault alarm is generated.</li> <li>• Listening object: logs in the <b>/run/log/journal</b> directory</li> </ul> <p><b>NOTE</b> The Ubuntu and HCE 2.0 OSs do not support the preceding check items due to incompatible log formats.</p>
Frequent restarts of Docker FrequentDockerRestart	Periodically backtrack system logs to check whether the container runtime Docker restarts frequently.	
Frequent restarts of containerd FrequentContainerdRestart	Periodically backtrack system logs to check whether the container runtime containerd restarts frequently.	
kubelet error KubeletProblem	Check the status of the key component Kubelet.	None
kube-proxy error KubeProxyProblem	Check the status of the key component kube-proxy.	None

**Table 3-34** Checking system metrics

Check Item	Function	Description
Conntrack table full ConntrackFullProblem	Check whether the conntrack table is full.	<ul style="list-style-type: none"> <li>• Default threshold: 90%</li> <li>• Usage: <b>nf_conntrack_count</b></li> <li>• Maximum value: <b>nf_conntrack_max</b></li> </ul>

Check Item	Function	Description
Insufficient disk resources DiskProblem	Check the usage of the system disk and CCE data disks (including the CRI logical disk and kubelet logical disk) on the node.	<ul style="list-style-type: none"> <li>• Default threshold: 90%</li> <li>• Source: df -h</li> </ul> <p>Currently, additional data disks are not supported.</p>
Insufficient file handles FDProblem	Check if the FD file handles are used up.	<ul style="list-style-type: none"> <li>• Default threshold: 90%</li> <li>• Usage: the first value in <b>/proc/sys/fs/file-nr</b></li> <li>• Maximum value: the third value in <b>/proc/sys/fs/file-nr</b></li> </ul>
Insufficient node memory MemoryProblem	Check whether memory is used up.	<ul style="list-style-type: none"> <li>• Default threshold: 80%</li> <li>• Usage: <b>MemTotal-MemAvailable</b> in <b>/proc/meminfo</b></li> <li>• Maximum value: <b>MemTotal</b> in <b>/proc/meminfo</b></li> </ul>
Insufficient process resources PIDProblem	Check whether PID process resources are exhausted.	<ul style="list-style-type: none"> <li>• Default threshold: 90%</li> <li>• Usage: <b>nr_threads</b> in <b>/proc/loadavg</b></li> <li>• Maximum value: smaller value between <b>/proc/sys/kernel/pid_max</b> and <b>/proc/sys/kernel/threads-max</b>.</li> </ul>

**Table 3-35** Checking the storage

Check Item	Function	Description
Disk read-only DiskReadonly	Periodically perform write tests on the system disk and CCE data disks (including the CRI logical disk and Kubelet logical disk) of the node to check the availability of key disks.	Detection paths: <ul style="list-style-type: none"> <li>• /mnt/paas/kubernetes/kubelet/</li> <li>• /var/lib/docker/</li> <li>• /var/lib/containerd/</li> <li>• /var/paas/sys/log/cceaddon-npd/</li> </ul> The temporary file <b>npd-disk-write-ping</b> is generated in the detection path. Currently, additional data disks are not supported.

Check Item	Function	Description
<p>emptyDir storage pool error</p> <p>EmptyDirVolumeGroupStatusError</p>	<p>Check whether the ephemeral volume group on the node is normal.</p> <p>Impact: Pods that depend on the storage pool cannot write data to the temporary volume. The temporary volume is remounted as a read-only file system by the kernel due to an I/O error.</p> <p>Typical scenario: When creating a node, a user configures two data disks as an ephemeral volume storage pool. Some data disks are deleted by mistake. As a result, the storage pool becomes abnormal.</p>	<ul style="list-style-type: none"> <li>● Detection period: 30s</li> <li>● Source: vgs -o vg_name, vg_attr</li> <li>● Principle: Check whether the VG (storage pool) is in the P state. If yes, some PVs (data disks) are lost.</li> <li>● Joint scheduling: The scheduler can automatically identify a PV storage pool error and prevent pods that depend on the storage pool from being scheduled to the node.</li> <li>● Exceptional scenario: The NPD add-on cannot detect the loss of all PVs (data disks), resulting in the loss of VGs (storage pools). In this case, kubelet automatically isolates the node, detects the loss of VGs (storage pools), and updates the corresponding resources in <b>nodestatus.allocatable</b> to <b>0</b>. This prevents pods that depend on the storage pool from being scheduled to the node. The damage of a single PV cannot be detected by this check item, but by the ReadonlyFilesystem check item.</li> </ul>
<p>PV storage pool error</p> <p>LocalPvVolumeGroupStatusError</p>	<p>Check the PV group on the node.</p> <p>Impact: Pods that depend on the storage pool cannot write data to the persistent volume. The persistent volume is remounted as a read-only file system by the kernel due to an I/O error.</p> <p>Typical scenario: When creating a node, a user configures two data disks as a persistent volume storage pool. Some data disks are deleted by mistake.</p>	

Check Item	Function	Description
<p>Mount point error</p> <p>MountPointProblem</p>	<p>Check the mount point on the node.</p> <p>Exceptional definition: You cannot access the mount point by running the <b>cd</b> command.</p> <p>Typical scenario: Network File System (NFS), for example, obsfs and s3fs is mounted to a node. When the connection is abnormal due to network or peer NFS server exceptions, all processes that access the mount point are suspended. For example, during a cluster upgrade, a kubelet is restarted, and all mount points are scanned. If the abnormal mount point is detected, the upgrade fails.</p>	<p>Alternatively, you can run the following command:</p> <pre>for dir in `df -h   grep -v "Mounted on"   awk "{print \\\$NF}"`;do cd \$dir; done &amp;&amp; echo "ok"</pre>



Check Item	Function	Description
<p>Suspended disk I/O DiskHung</p>	<p>Check whether I/O suspension occurs on all disks on the node, that is, whether I/O read and write operations are not responded.</p> <p>Definition of I/O suspension: The system does not respond to disk I/O requests, and some processes are in the D state.</p> <p>Typical scenario: Disks cannot respond due to abnormal OS hard disk drivers or severe faults on the underlying network.</p>	<ul style="list-style-type: none"> <li>• Check object: all data disks</li> <li>• Source: /proc/diskstat Alternatively, you can run the following command: iostat -xmt 1</li> <li>• Thresholds: (All following conditions must be met). <ul style="list-style-type: none"> <li>- Average usage (<b>ioutil</b>) <math>\geq 0.99</math></li> <li>- Average I/O queue length (<b>avgqu-sz</b>) <math>\geq 1</math></li> <li>- Average I/O transfer volume <math>\leq 1</math> Average I/O transfer volume = Number of writes completed per second (<b>iops</b>, unit: w/s) + Amount of data written per second (<b>ioth</b>, unit: wMB/s)</li> </ul> </li> </ul> <p><b>NOTE</b> In some OSs, no data changes during I/O. In this case, calculate the CPU I/O time usage. The value of <b>iowait</b> should be greater than 0.8.</p>

Check Item	Function	Description
Slow disk I/O DiskSlow	<p>Check whether all disks on the node have slow I/Os, that is, whether I/Os respond slowly.</p> <p>Typical scenario: EVS disks have slow I/Os due to network fluctuation.</p>	<ul style="list-style-type: none"> <li>• Check object: all data disks</li> <li>• Source: /proc/diskstat Alternatively, you can run the following command: iostat -xmt 1</li> <li>• Default threshold: Average I/O latency (<b>await</b>) ≥ 5000 ms</li> </ul> <p><b>NOTE</b> If I/O requests are not responded and the <b>await</b> data is not updated, this check item is invalid.</p>

**Table 3-36** Other check items

Check Item	Function	Description
Abnormal NTP NTPProblem	Check whether the node clock synchronization service ntpd or chronyd is running properly and whether a system time drift is caused.	Default clock offset threshold: 8000 ms
Process D error ProcessD	Check whether there is a process D on the node.	Default threshold: 10 abnormal processes detected for three consecutive times Source: <ul style="list-style-type: none"> <li>• /proc/{PID}/stat</li> <li>• Alternately, you can run the <b>ps aux</b> command.</li> </ul> Exceptional scenario: The ProcessD check item ignores the resident D processes (heartbeat and update) on which the SDI driver on the BMS node depends.
Process Z error ProcessZ	Check whether the node has processes in Z state.	

Check Item	Function	Description
ResolvConf error ResolvConfFileProblem	Check whether the ResolvConf file is lost. Check whether the ResolvConf file is normal. Exceptional definition: No upstream domain name resolution server (nameserver) is included.	Object: <b>/etc/resolv.conf</b>
Existing scheduled event ScheduledEvent	Check whether scheduled live migration events exist on the node. A live migration plan event is usually triggered by a hardware fault and is an automatic fault rectification method at the IaaS layer. Typical scenario: The host is faulty. For example, the fan is damaged or the disk has bad sectors. As a result, live migration is triggered for VMs.	Source: <ul style="list-style-type: none"> <li>• <a href="http://169.254.169.254/metadata/latest/events/scheduled">http://169.254.169.254/metadata/latest/events/scheduled</a></li> </ul> This check item is an Alpha feature and is disabled by default.

The kubelet component has the following default check items, which have bugs or defects. You can fix them by upgrading the cluster or using NPDP.

**Table 3-37** Default kubelet check items

Check Item	Function	Description
Insufficient PID resources PIDPressure	Check whether PIDs are sufficient.	<ul style="list-style-type: none"> <li>• Interval: 10 seconds</li> <li>• Threshold: 90%</li> <li>• Defect: In community version 1.23.1 and earlier versions, this check item becomes invalid when over 65535 PIDs are used. For details, see <a href="#">issue 107107</a>. In community version 1.24 and earlier versions, thread-max is not considered in this check item.</li> </ul>

Check Item	Function	Description
Insufficient memory MemoryPressure	Check whether the allocable memory for the containers is sufficient.	<ul style="list-style-type: none"> <li>● Interval: 10 seconds</li> <li>● Threshold: max. 100 MiB</li> <li>● Allocable = Total memory of a node – Reserved memory of a node</li> <li>● Defect: This check item checks only the memory consumed by containers, and does not consider that consumed by other elements on the node.</li> </ul>
Insufficient disk resources DiskPressure	Check the disk usage and inodes usage of the kubelet and Docker disks.	<ul style="list-style-type: none"> <li>● Interval: 10 seconds</li> <li>● Threshold: 90%</li> </ul>

# 4 Node Pools

---

## 4.1 Node Pool Overview

### Introduction

CCE introduces node pools to help you better manage nodes in Kubernetes clusters. A node pool contains one node or a group of nodes with identical configuration in a cluster.

You can create custom node pools on the CCE console. With node pools, you can quickly create, manage, and destroy nodes without affecting the cluster. All nodes in a custom node pool have identical parameters and node type. You cannot configure a single node in a node pool; any configuration changes affect all nodes in the node pool.

You can also use node pools for auto scaling (supported only by pay-per-use node pools).

- When a pod in a cluster cannot be scheduled due to insufficient resources, scale-out can be automatically triggered.
- When there is an idle node or a monitoring metric threshold is met, scale-in can be automatically triggered.

This section describes how node pools work in CCE and how to create and manage node pools.

### Node Pool Architecture

Generally, all nodes in a node pool have the following same attributes:

- Node OS
- Node login mode
- Node container runtime
- Startup parameters of Kubernetes components on a node
- Custom startup script of a node
- Kubernetes labels and taints

CCE provides the following extended attributes for node pools:

- Node pool OS
- Maximum number of pods on each node in a node pool

## Description of DefaultPool

DefaultPool is not a real node pool. It only **classifies** nodes that are not in the custom node pools. These nodes are directly created on the console or by calling APIs. DefaultPool does not support any user-created node pool functions, including scaling and parameter configuration. DefaultPool cannot be edited, deleted, expanded, or auto scaled, and nodes in it cannot be migrated.

## Applicable Scenarios

When a large-scale cluster is required, you are advised to use node pools to manage nodes.

The following table describes multiple scenarios of large-scale cluster management and the functions of node pools in each scenario.

**Table 4-1** Using node pools for different management scenarios

Scenario	Function
Multiple heterogeneous nodes (with different models and configurations) in the cluster	Nodes can be grouped into different pools for management.
Frequent node scaling required in a cluster	Node pools support auto scaling to dynamically add or reduce nodes.
Complex application scheduling rules in a cluster	Node pool tags can be used to quickly specify service scheduling rules.

## Functions and Precautions

Function	Description	Precaution
Creating a node pool	Add a node pool.	It is recommended that a cluster contains no more than 100 node pools.
Deleting a node pool	Deleting a node pool will delete nodes in the pool. Pods on these nodes will be automatically migrated to available nodes in other node pools.	If pods in the node pool have a specific node selector and none of the other nodes in the cluster satisfies the node selector, the pods will become unschedulable.

Function	Description	Precaution
Enabling auto scaling for a node pool	After auto scaling is enabled, nodes will be automatically created or deleted in the node pool based on the cluster loads.	Do not store important data on nodes in a node pool because the nodes may be deleted after scale-in. Data on the deleted nodes cannot be restored.
Disabling auto scaling for a node pool	After auto scaling is disabled, the number of nodes in a node pool will not automatically change with the cluster loads.	None
Adjusting the size of a node pool	The number of nodes in a node pool can be directly adjusted. If the number of nodes is reduced, nodes are randomly removed from the current node pool.	After auto scaling is enabled, you are not advised to manually adjust the node pool size.
Modifying node pool configurations	You can change the node pool name and number of nodes, add or delete Kubernetes labels and taints, and adjust node pool configurations such as the disk, OS, and container engine of the node pool.	The deleted or added Kubernetes labels and taints (as well as their quantity) will apply to all nodes in the node pool, which may cause pod re-scheduling. Therefore, exercise caution when performing this operation.
Removing a node from a node pool	Nodes in a node pool can be migrated to the default node pool of the same cluster.	Nodes in the default node pool cannot be migrated to other node pools, and nodes in a user-created node pool cannot be migrated to other user-created node pools.
Copying a node pool	You can copy the configuration of an existing node pool to create a new node pool.	None
Setting Kubernetes parameters	You can configure core components with fine granularity.	<ul style="list-style-type: none"> <li>This function is supported only in clusters of v1.15 and later. It is not displayed for versions earlier than v1.15.</li> <li>The default node pool does not support this type of configuration.</li> </ul>

## Deploying a Workload in a Specified Node Pool

When creating a workload, you can constrain pods to run in a specified node pool.

For example, on the CCE console, you can set the affinity between the workload and the node on the **Scheduling Policies** tab page on the workload details page to forcibly deploy the workload to a specific node pool. In this way, the workload runs only on nodes in the node pool. To better control where the workload is to be scheduled, you can use affinity or anti-affinity policies between workloads and nodes described in [Configuring Node Affinity Scheduling \(nodeAffinity\)](#).

For example, you can use container's resource request as a nodeSelector so that workloads will run only on the nodes that meet the resource request.

If the workload definition file defines a container that requires four CPUs, the scheduler will not choose the nodes with two CPUs to run workloads.

## Related Operations

You can log in to the CCE console and refer to the following sections to perform operations on node pools:

- [Creating a Node Pool](#)
- [Managing a Node Pool](#)
- [Creating a Deployment](#)
- [Configuring Node Affinity Scheduling \(nodeAffinity\)](#)

## 4.2 Upgrading to New Node Pools

Upgraded node pools provide additional benefits beyond flexible payment options (pay-per-use and yearly/monthly). They also enhance configuration management and improve resource management efficiency and flexibility.

### Why to Upgrade to the New Version?

- **Flexible resource configuration:** New node pools allow you to create pay-per-use nodes and select more cost-effective yearly/monthly nodes, depending on your needs.
- **Multiple flavors available:** You can easily filter different flavors for nodes, such as GPU nodes, based on CPU and memory parameters to meet the specific requirements of various service scenarios.
- **Advanced auto scaling:** New node pools can automatically scale and you can configure multiple scaling policies to optimize resource utilization for different service scenarios.
- **Enhanced configuration management:** New node pools offer improved customization options for Kubernetes parameters and provide better guidance to meet the complex requirements of managing containerized applications.

### What Are Changes in the New Version?

- Yearly/Monthly nodes created in the new node pools cannot be manually scaled in. They can only be deleted or unsubscribed from.



- During auto scaling, only pay-per-use nodes are scaled in or out by default.
- You cannot create yearly/monthly nodes using the API for updating a node pool. Instead, you must increase or decrease them.
- By default, nodes created in new yearly/monthly node pools will be billed on a pay-per-use basis.

**Table 4-2** Comparison before and after a node pool upgrade

Type	Before the Upgrade		After the Upgrade
	Pay-per-Use Node Pool	Yearly/Monthly Node Pool	
Billing mode of nodes	Pay-per-use only	Yearly/Monthly only	Both pay-per-use and yearly/monthly
Node scale-out	Change the number of nodes in the node pool configuration.	Change the number of nodes in the node pool configuration.	Scale in or out the node pool.
Node scale-in	<ul style="list-style-type: none"> <li>• Change the number of nodes in the node pool configuration.</li> <li>• Delete or remove nodes.</li> </ul>	<ul style="list-style-type: none"> <li>• Change the number of nodes in the node pool configuration.</li> <li>• Unsubscribe from or remove nodes.</li> </ul>	Pay-per-use nodes: Decrease them. Yearly/Monthly nodes: Unsubscribe from or remove them.
Upgrade	Upgrade the cluster to v1.21.11-r0, v1.23.9-r0, v1.25.4-r0, or later. <ul style="list-style-type: none"> <li>• After a pay-per-use node pool is upgraded to the new version, all features of the new version will be readily accessible without any additional steps.</li> <li>• Increasing or decreasing the number of nodes in a yearly/monthly node pool will trigger an automatic upgrade to the new version. Upon confirmation, the node pool will be upgraded without affecting the existing nodes.</li> </ul> <p><b>NOTE</b> After a node pool is upgraded, any nodes added using the API for updating a node pool will be pay-per-use by default.</p>		

## How to Upgrade to the New Version?

You need to upgrade your cluster to v1.21.11-r0, v1.23.9-r0, v1.25.4-r0, or later, and then perform the following operations to trigger the upgrade process of new node pool.

- 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. In the right pane, click the **Node Pools** tab.
  - Step 3** Click **Scaling** in the **Operation** column of the target node pool. In the displayed dialog box, click **Upgrade Now**.
- End

## 4.3 Creating a Node Pool

### Scenario

This section describes how to create a node pool and perform operations on the node pool. For details about how a node pool works, see [Node Pool Overview](#).

### Procedure

- 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. In the right pane, click the **Node Pools** tab.
- Step 3** In the upper right corner of the page, click **Create Node Pool**.

#### Basic Settings

**Table 4-3** Basic settings

Parameter	Description
Node Pool Name	Name of a node pool. By default, the name is in the format of <i>Cluster name-nodepool-Random number</i> . If you do not want to use the default name format, you can customize the name.

#### Configurations

You can configure the flavor and OS of a cloud server, on which your containerized applications run.

**Table 4-4** Node configuration parameters

Parameter	Description
Node Type	<p>Select a node type based on service requirements. Then, you can select a proper flavor from the node flavor list.</p> <p>CCE standard clusters support the following node types:</p> <ul style="list-style-type: none"> <li>ECS (VM): A virtualized ECS is used as a cluster node.</li> </ul> <p>CCE Turbo clusters support the following node types:</p> <ul style="list-style-type: none"> <li>ECS (VM): A virtualized ECS is used as a cluster node. A CCE Turbo cluster supports only the cloud servers that allow multiple ENIs. Select a server type displayed on the CCE console.</li> <li>ECS (physical machine): A QingTian-backed bare metal server is used as a cluster node.</li> </ul>
Specifications	<p>Select a node flavor based on service requirements. The available node flavors vary depending on regions or AZs. For details, see the CCE console.</p>
Container Engine	<p>The container engines supported by CCE include Docker and containerd, which may vary depending on cluster types, cluster versions, and OSs. Select a container engine based on the information displayed on the CCE console. For details, see <a href="#">Mapping Between Node OSs and Container Engines</a>.</p>
OS	<p>Select an OS type. Different types of nodes support different OSs.</p> <ul style="list-style-type: none"> <li><b>Public image:</b> Select a public image for the node.</li> </ul> <p><b>NOTE</b> 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.</p>
Login Mode	

### Storage Settings

Configure storage resources on a node for the containers running on it. Select a disk type and configure its size based on service requirements. For details about EVS disks, see [Disk Types and Performance](#).

**Table 4-5** Configuration parameters

Parameter	Description
System Disk	<p>System disk used by the node OS. The value ranges from 40 GiB to 1024 GiB. The default value is 50 GiB.</p>

Parameter	Description
Data Disk	<p><b>At least one data disk is required</b> for the container runtime and kubelet components. <b>This data disk cannot be deleted or detached. Otherwise, the node will be unavailable.</b></p> <ul style="list-style-type: none"> <li>• Default data disk: used for container runtime and kubelet components. The disk size ranges from 20 GiB to 32768 GiB. The default value is 100 GiB.</li> <li>• Other common data disks: You can set the data disk size to a value ranging from 10 GiB to 32768 GiB. The default value is 100 GiB.</li> </ul> <p><b>NOTE</b></p> <ul style="list-style-type: none"> <li>• If the node flavor is disk-intensive or ultra-high I/O, one data disk can be a local disk.</li> <li>• Local disks may break down and do not ensure data reliability. Store your service data in EVS disks, which are more reliable than local disks.</li> </ul> <p><b>Advanced Settings</b></p> <p>Expand the area and configure the following parameters:</p> <ul style="list-style-type: none"> <li>• <b>Data Disk Space Allocation:</b> allocates space for container engines, images, and ephemeral storage for them to run properly. For details about how to allocate data disk space, see <a href="#">Space Allocation of a Data Disk</a>.</li> </ul> <p><b>Adding data disks</b></p> <p>A maximum of 16 data disks can be attached to an ECS and 10 to a BMS. By default, a raw disk is created without any processing. You can also click <b>Expand</b> and select any of the following options:</p> <ul style="list-style-type: none"> <li>• <b>Default:</b> By default, a raw disk is created without any processing.</li> <li>• <b>Mount Disk:</b> The data disk is attached to a specified directory.</li> </ul>

### Network Settings

Configure networking resources to allow node and containerized application access.

**Table 4-6** Configuration parameters

Parameter	Description
Virtual Private Cloud	The VPC to which the cluster belongs by default, which cannot be changed.

Parameter	Description
Node Subnet	<p>The node subnet selected during cluster creation is used by default. You can choose another subnet instead.</p> <ul style="list-style-type: none"> <li>Multiple subnets: You can select multiple subnets in the same VPC for nodes. Newly added nodes will preferentially use the IP addresses from the top-ranking subnet.</li> <li>Single subnet: Only one subnet is configured for your node pool. If the IP addresses of a single subnet are insufficient, configure multiple subnets. Otherwise, a node pool scale-out may fail.</li> </ul>
Node IP Address	Random allocation is supported.
Associate Security Group	<p>Security group used by the nodes created in the node pool. A maximum of five security groups can be selected.</p> <p>When a cluster is created, a node security group named <b>{Cluster name}-cce-node-{Random ID}</b> is created and used by default.</p> <p>Traffic needs to pass through certain ports in the node security group to ensure node communications. Ensure that you have enabled these ports if you select another security group.</p> <p><b>NOTE</b> After a node pool is created, its associated security group cannot be modified.</p>

### Advanced Settings

Configure advanced node capabilities such as labels, taints, and startup command.

**Table 4-7** Advanced configuration parameters

Parameter	Description
Resource Tag	<p>You can add resource tags to classify resources.</p> <p>You can create <b>predefined tags</b> on the TMS console. The predefined tags are available to all resources that support tags. You can use these tags to improve the tag creation and resource migration efficiency.</p> <p>CCE will automatically create the "CCE-Dynamic-Provisioning-Node=<i>Node ID</i>" tag.</p>
Kubernetes Label	<p>A key-value pair added to a Kubernetes object (such as a pod). After specifying a label, click <b>Add Label</b> for more. A maximum of 20 labels can be added.</p> <p>Labels can be used to distinguish nodes. With workload affinity settings, container pods can be scheduled to a specified node. For more information, see <a href="#">Labels and Selectors</a>.</p>

Parameter	Description
Taint	<p>This parameter is left blank by default. You can add taints to configure anti-affinity for the node. A maximum of 20 taints are allowed for each node. Each taint contains the following parameters:</p> <ul style="list-style-type: none"> <li>• <b>Key:</b> A key must contain 1 to 63 characters, starting with a letter or digit. Only letters, digits, hyphens (-), underscores (_), and periods (.) are allowed. A DNS subdomain name can be used as the prefix of a key.</li> <li>• <b>Value:</b> A value must contain 1 to 63 characters, starting with a letter or digit. Only letters, digits, hyphens (-), underscores (_), and periods (.) are allowed.</li> <li>• <b>Effect:</b> Available options are <b>NoSchedule</b>, <b>PreferNoSchedule</b>, and <b>NoExecute</b>.</li> </ul> <p>For details, see <a href="#">Managing Node Taints</a>.</p> <p><b>NOTE</b> For a cluster of v1.19 or earlier, the workload may have been scheduled to a node before the taint is added. To avoid such a situation, select a cluster of v1.19 or later.</p>
Synchronization for Existing Nodes	<p>After the options are selected, changes to resource tags and Kubernetes labels/taints in a node pool will be synchronized to existing nodes in the node pool.</p>
Max. Pods	<p>Maximum number of pods that can run on the node, including the default system pods.</p> <p>This limit prevents the node from being overloaded with pods.</p> <p>This number is also decided by other factors. For details, see <a href="#">Maximum Number of Pods That Can Be Created on a Node</a>.</p>
ECS Group	<p>An ECS group logically groups ECSs. The ECSs in the same ECS group comply with the same policy associated with the ECS group.</p> <p>Anti-affinity: ECSs in an ECS group are deployed on different physical hosts to improve service reliability.</p> <p>Select an existing ECS group, or click <b>Add ECS Group</b> to create one. After the ECS group is created, click the refresh icon.</p>
Pre-installation Command	<p>Installation script command, in which Chinese characters are not allowed. The script command will be Base64-transcoded.</p> <p>The script will be executed before Kubernetes software is installed. Note that if the script is incorrect, Kubernetes software may fail to be installed.</p>

Parameter	Description
Post-installation Command	<p>Installation script command, in which Chinese characters are not allowed. The script command will be Base64-transcoded. The script will be executed after Kubernetes software is installed, which does not affect the installation.</p> <p><b>NOTE</b> Do not run the <b>reboot</b> command in the post-installation script to restart the system immediately. To restart the system, run the <b>shutdown -r 1</b> command to restart with a delay of one minute.</p>
Agency	<p>An agency is created by the tenant administrator on the IAM console. Using an agency, you can share your cloud server resources with another account, or entrust a more professional person or team to manage your resources.</p> <p>If no agency is available, click <b>Create Agency</b> on the right to create one.</p>

**Step 4** Click **Next: Confirm**.

**Step 5** Click **Submit**.

----End

## Related Operations

By default, the total number of nodes in a node pool is set to 0 upon creation. You will need to manually increase the number of nodes. For details, see [Scaling a Node Pool](#).

## 4.4 Scaling a Node Pool

You can specify a specification in a node pool for scaling.

### NOTICE

The default node pool does not support scaling. Use [Creating a Node](#) to add a node.

**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. In the right pane, click the **Node Pools** tab.

**Step 3** Choose **Scaling** next to the target node pool.

**Step 4** In the displayed **Node Pool Scaling** window, configure scaling parameters.

- Add or reduce nodes for scaling.
- Use the selected flavor to increase or decrease the number of nodes.
- Select a billing mode only when adding nodes.

- Pay-per-use  
Resources will be billed based on usage duration. You can provision or delete resources at any time.
- Configure the number of nodes to be added or deleted.
  - When scaling out a node pool, make sure that the total number of nodes, both existing and new, does not exceed the management scale of the current cluster.
  - When scaling in a node pool, make sure that the number of nodes to be removed does not exceed the number of nodes currently in the pool.  
Scaling in can result in the unavailability of resources associated with a node, such as local storage and workloads that were scheduled to that node. Exercise caution when performing this operation to avoid impact on running services.

### Node Pool Scaling

Node Pool Name XXXXXXXXXX Pay-per-use

Current Quantity 0

Scaling Add Node Reduce Nodes

Expand Specifications c7n.4xlarge.2 | AZ1 v

Use the selected flavor to expand the node capacity. If the flavor resources are insufficient, the capacity expansion will fail.

Billing Mode Pay-per-use

Number of nodes to be added 
- 0 +

Max. nodes that can be created at a time: 50

**Step 5** Click **OK**.

----End

## 4.5 Managing a Node Pool

### 4.5.1 Updating a Node Pool

#### Precautions

- Changes to the container engine, OS, or pre-/post-installation script in a node pool take effect only on new nodes. To synchronize the modification onto existing nodes, manually reset these nodes.
- Changes to resource tags and Kubernetes labels/taints in a node pool will be automatically synchronized to existing nodes after the options of



**Synchronization for Existing Nodes** are selected. You do not need to reset these nodes.

## Updating a Node Pool

- 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. In the right pane, click the **Node Pools** tab.
- Step 3** Click **Update** next to the name of the node pool you will edit. Configure the parameters in the displayed **Update Node Pool** page.

### Basic Settings

**Table 4-8** Basic settings

Parameter	Description
Node Pool Name	Name of the node pool.

### Network Settings

**Table 4-9** Configuration parameters

Parameter	Description
Virtual Private Cloud	The VPC to which the cluster belongs by default, which cannot be changed.
Node Subnet	<p>The node subnet selected during cluster creation is used by default. You can choose another subnet instead.</p> <ul style="list-style-type: none"> <li>• Multiple subnets: You can select multiple subnets in the same VPC for nodes. Newly added nodes will preferentially use the IP addresses from the top-ranking subnet.</li> <li>• Single subnet: Only one subnet is configured for your node pool. If the IP addresses of a single subnet are insufficient, configure multiple subnets. Otherwise, a node pool scale-out may fail.</li> </ul>

### Advanced Settings

**Table 4-10** Advanced settings

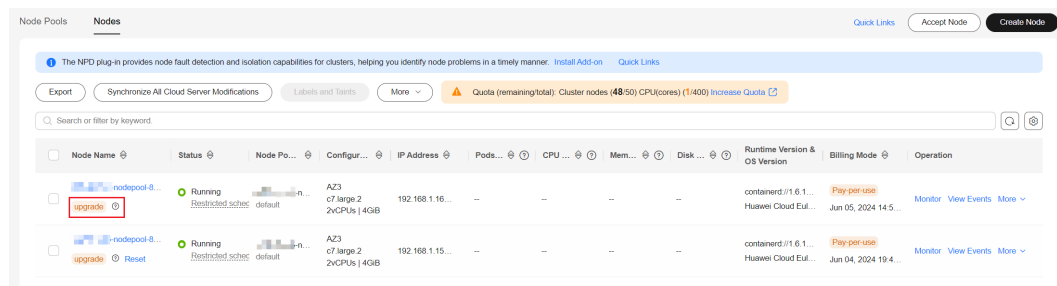
Parameter	Description
Resource Tag	<p>You can add resource tags to classify resources.</p> <p>You can create <b>predefined tags</b> on the TMS console. The predefined tags are available to all resources that support tags. You can use these tags to improve the tag creation and resource migration efficiency.</p> <p>CCE will automatically create the "CCE-Dynamic-Provisioning-Node=<i>Node ID</i>" tag.</p> <p><b>NOTE</b> Modified resource tags automatically take effect on new nodes as well as existing nodes if <b>Resource tags synchronized</b> is selected in <b>Synchronization for Existing Nodes</b>.</p>
Kubernetes Label	<p>A key-value pair added to a Kubernetes object (such as a pod). After specifying a label, click <b>Add Label</b> for more. A maximum of 20 labels can be added.</p> <p>Labels can be used to distinguish nodes. With workload affinity settings, container pods can be scheduled to a specified node. For more information, see <a href="#">Labels and Selectors</a>.</p> <p><b>NOTE</b> Modified Kubernetes labels automatically take effect on new nodes as well as existing nodes if <b>Kubernetes labels</b> is selected in <b>Synchronization for Existing Nodes</b>.</p>
Taint	<p>This parameter is left blank by default. You can add taints to configure anti-affinity for the node. A maximum of 20 taints are allowed for each node. Each taint contains the following parameters:</p> <ul style="list-style-type: none"> <li>• <b>Key:</b> A key must contain 1 to 63 characters, starting with a letter or digit. Only letters, digits, hyphens (-), underscores (_), and periods (.) are allowed. A DNS subdomain name can be used as the prefix of a key.</li> <li>• <b>Value:</b> A value must contain 1 to 63 characters, starting with a letter or digit. Only letters, digits, hyphens (-), underscores (_), and periods (.) are allowed.</li> <li>• <b>Effect:</b> Available options are <b>NoSchedule</b>, <b>PreferNoSchedule</b>, and <b>NoExecute</b>.</li> </ul> <p>For details, see <a href="#">Managing Node Taints</a>.</p> <p><b>NOTE</b> Modified taints automatically take effect on new nodes as well as existing nodes if <b>Taints</b> is selected in <b>Synchronization for Existing Nodes</b>.</p>

Parameter	Description
<p>Synchronization for Existing Nodes</p>	<p>After the options are selected, changes to resource tags and Kubernetes labels/taints in a node pool will be synchronized to existing nodes in the node pool.</p> <p><b>NOTE</b> When you update a node pool, pay attention to the following if you change the state of <b>Resource tags synchronized</b>:</p> <ul style="list-style-type: none"> <li>• After the option is selected: <ul style="list-style-type: none"> <li>– CCE will synchronize the resource tags configured in the node pool to existing nodes. If a resource tag with the same key of a resource tag in the node pool already exists on an ECS, the value of the tag on the ECS will be changed to that of the resource tag in the node pool.</li> <li>– Typically, it takes less than 10 minutes to synchronize resource tags onto existing nodes, depending on the number of nodes in the node pool.</li> <li>– Issue a resource tag synchronization request only after the previous synchronization is complete. Otherwise, the resource tags may be inconsistent between existing nodes.</li> </ul> </li> </ul> <p>When you update a node pool, pay attention to the following if you change the state of <b>Kubernetes labels</b> or <b>Taints</b>:</p> <ul style="list-style-type: none"> <li>• When these options are deselected, the Kubernetes labels/taints of the existing and new nodes in the node pool may be inconsistent. If service scheduling relies on node labels or taints, the scheduling may fail or the node pool may fail to scale.</li> <li>• When these options are selected: <ul style="list-style-type: none"> <li>– If you have <b>modified</b> or <b>added</b> labels or taints in the node pool, the modifications will be automatically synchronized to existing nodes typically in 10 minutes after <b>Kubernetes labels</b> or <b>Taints</b> is selected.</li> <li>– If you have <b>deleted</b> a label or taint in the node pool, you must manually delete the label or taint on the node list page after <b>Kubernetes labels</b> or <b>Taints</b> is selected.</li> <li>– If you have <b>manually changed the key or effect of a taint on an existing node</b>, a new taint will be added to the existing node after <b>Kubernetes labels</b> or <b>Taints</b> is selected. In the new taint, its key is different from the manually changed key but its value and effect are the same as those manually changed ones, or its effect is different from the manually changed effect but its key and value are the same as those manually changed ones. This is because a Kubernetes taint natively uses a key and effect as a key-value pair. The taints with different keys or effects are considered as two taints.</li> </ul> </li> </ul>
<p>Agency</p>	<p>An agency is created by the account administrator on the IAM console. Using an agency, you can share your cloud server resources with another account, or entrust a more professional person or team to manage your resources.</p> <p>If no agency is available, click <b>Create Agency</b> on the right to create one.</p> <p><b>NOTE</b> After an agency is modified, the modification will only apply to new nodes and not to existing ones, even if they are reset.</p>

Parameter	Description
User-defined node name prefix and suffix	<p>Custom name prefix and suffix of a node in a node pool. After the configuration, the nodes in the node pool will be named with the configured prefix and suffix. For example, if the prefix is <b>prefix-</b> and the suffix is <b>-suffix</b>, the nodes in the node pool will be named in the format of "prefix-Node pool name with five-digit random characters-suffix".</p> <ul style="list-style-type: none"> <li>A prefix and suffix can be customized only when a node pool is created, and they cannot be modified after the node pool is created.</li> <li>A prefix can end with a special character, and a suffix can start with a special character.</li> <li>A node name consists of a maximum of 56 characters in the format of "Prefix-Node pool name with five-digit random characters-Suffix".</li> <li>A node name does not support the combination of a period (.) and special characters (such as .., .-, or -.).</li> <li>This function is available only in clusters of v1.28.1, v1.27.3, v1.25.6, v1.23.11, v1.21.12, or later.</li> </ul> <p><b>NOTE</b> After the custom name prefix and suffix are modified, the modification will only apply to new nodes and not to existing ones, even if they are reset.</p>

**Step 4** After the configuration, click **OK**.

After node pool parameters are modified, you can find the update on the **Nodes** page. Reset the nodes in the target node pool to synchronize the configuration update.



----End

## 4.5.2 Updating an AS Configuration

Auto Scaling (AS) enables elastic scaling of nodes in a node pool based on scaling policies. Without this function, you have to manually adjust the number of nodes in a node pool.

## Notes and Constraints

To enable AS, the [CCE Cluster Autoscaler](#) add-on must be installed in the target cluster.

## Procedure

**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 **Nodes**. On the **Node Pools** tab, locate the row containing the target node pool and click **Auto Scaling**.

- If Autoscaler has not been installed, configure add-on parameters based on service requirements, click **Install**, and wait until the add-on is installed. For details about add-on configurations, see [CCE Cluster Autoscaler](#).
- If Autoscaler has been installed, directly configure scaling policies.

**Step 3** Configure auto scaling policies.

### AS Configuration

- **Customized Rule:** Click **Add Rule**. In the dialog box displayed, configure parameters. You can add multiple node scaling policies, a maximum of one CPU usage-based rule, and one memory usage-based rule. The total number of rules cannot exceed 10.

The following table lists custom rules.

**Table 4-11** Custom rules

Rule Type	Configuration
Metric-based	<ul style="list-style-type: none"> <li>- <b>Trigger:</b> Select <b>CPU allocation rate</b> or <b>Memory allocation rate</b> and enter a value. The percentage must be greater than the value specified in the node resource requirements for a node scale-in when you configure a scaling policy (<a href="#">Configuring an Auto Scaling Policy for a Cluster</a>).</li> </ul> <p><b>NOTE</b></p> <ul style="list-style-type: none"> <li>▪ Resource allocation (%) = Resources requested by pods in the node pool/Resources allocatable to pods in the node pool</li> <li>▪ <b>If multiple rules meet the conditions, the rules are executed in either of the following modes:</b> If rules based on the <b>CPU allocation rate</b> and <b>memory allocation rate</b> are configured and two or more rules meet the scale-out conditions, the rule that will add the most nodes will be executed. If a rule is configured based on the <b>CPU allocation rate</b> and a <b>periodic rule</b> and both the rules meet the scale-out conditions, the periodic rule executed early changes the node pool to the scaling state. As a result, the metric-based rule cannot be executed. After the periodic rule is executed and the node pool status becomes normal, the metric-based rule will not be executed. If the metric-based rule is executed early, the periodic rule will be executed after the metric-based rule is executed.</li> <li>▪ If a rule is configured based on the <b>CPU allocation rate</b> and <b>memory allocation rate</b>, the policy detection period varies with the processing logic of each loop of the Autoscaler add-on. A scale-out is triggered once the conditions are met, but it is constrained by other factors such as the cooldown period and node pool status.</li> <li>▪ If the number of nodes reaches the upper limit of the cluster scale, <a href="#">the upper limit of the nodes supported in a node pool</a>, or <a href="#">the upper limit of the nodes of a specific flavor</a>, a metric-based scale-out will not be triggered.</li> <li>▪ If the number of nodes, CPUs, or memory resources reaches <a href="#">the upper limit for a node scale-out</a>, a metric-based scale-out will not be triggered.</li> </ul> <ul style="list-style-type: none"> <li>- <b>Action:</b> Configure an action to be performed when the triggering condition is met. <ul style="list-style-type: none"> <li>▪ <b>Custom:</b> Add a specified number of nodes to a node pool.</li> <li>▪ <b>Auto calculation:</b> When the trigger condition is met, nodes are automatically added and the allocation rate is restored to a value lower than the threshold. The formula is as follows: Number of nodes to be added = [Resource request of pods in the node pool/(Available resources of a single node x Target allocation rate)] – Number of current nodes + 1</li> </ul> </li> </ul>

Rule Type	Configuration
Periodic	<ul style="list-style-type: none"> <li>- <b>Trigger Time:</b> You can select a specific time every day, every week, every month, or every year.</li> <li>- <b>Action:</b> specifies an action to be carried out when the trigger time is reached. A specified number of nodes will be added to the node pool.</li> </ul>

- **Nodes:** 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.

#### AS Object

- **Specifications:** Configure whether to enable auto scaling for node flavors in a node pool.

#### NOTE

If multiple flavors are configured for a node pool, you can specify the upper limit for the number of nodes and the priority for each flavor separately.

**Step 4** Click **OK**.

----End

## 4.5.3 Modifying Node Pool Configurations

### Notes and Constraints

The default node pool does not support the following management operations.

### Configuration Management

CCE allows you to highly customize Kubernetes parameter settings on core components in a cluster. For more information, see [kubelet](#).

This function is supported only in clusters of **v1.15 or later**. It is not displayed for versions earlier than v1.15.

- 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. In the right pane, click the **Node Pools** tab.
- Step 3** Click **Manage** in the **Operation** column of the target node pool
- Step 4** On the **Manage Configurations** page on the right, modify node pool parameter setting.

The following parameters can be configured for a node pool:

- [kubelet](#)

- [kube-proxy](#)
- [Docker \(Available Only for Docker Node Pools\)](#)
- [containerd \(Available Only for containerd Node Pools\)](#)
- [Networking Components \(Available Only for CCE Turbo Clusters\)](#)
- [Pod Security Groups \(Available Only for CCE Turbo Clusters\)](#)

**Step 5** Click **OK**.

----End

## kubelet

Item	Parameter	Description	Value	Modification
CPU management policy	cpu-management-policy	<p>CPU management policy configuration. For details, see <a href="#">CPU Scheduling</a>.</p> <ul style="list-style-type: none"> <li>• <b>none</b>: disables pods from exclusively occupying CPUs. Select this value if you want a large pool of shareable CPU cores.</li> <li>• <b>static</b>: enables pods to exclusively occupy CPUs. Select this value if your workload is sensitive to latency in CPU cache and scheduling.</li> <li>• <b>enhanced-static</b>: allows burstable pods to preferentially use CPU cores. Select this value if your workload has huge peak-trough difference and is in the trough state most of the time.</li> </ul>	Default: none	None



Item	Parameter	Description	Value	Modification
QPS for requests to kube-apiserver	kube-api-qps	Number of queries per second for communication with the API server.	Default: 100	None
Burst for requests to kube-apiserver	kube-api-burst	Maximum number of burst requests sent to the API server per second.	Default: 100	None
Limit on the pods managed by kubelet	max-pods	Maximum number of pods that can run on a node.	<ul style="list-style-type: none"> <li>For a CCE standard cluster, the maximum number of pods is determined based on <b>the maximum number of pods on a node.</b></li> </ul>	None
Limited number of processes in a pod	pod-pids-limit	Maximum number of PIDs that can be used in each pod.	Default: -1, which indicates that the number of PIDs is not limited	None
Whether to use a local IP address as a node's ClusterDNS	with-local-dns	The default ENI IP address of the node will be automatically added to the node's kubelet configuration as the preferred DNS address.	Default: false	None
QPS limit on creating events	event-qps	Number of events that can be generated per second.	Default: 5	None

Item	Parameter	Description	Value	Modification
Upper Limit for Burst Events	event-burst	Upper limit for burst event creation. The number of burst events can be temporarily increased to the specified value.	Default: 10	None
Allowed unsafe sysctls	allowed-unsafe-sysctls	Insecure system configuration allowed. Starting from <b>v1.17.17</b> , CCE enables pod security policies for kube-apiserver. Add corresponding configurations to <b>allowedUnsafeSysctls</b> of a pod security policy to make the policy take effect. (This configuration is not required for clusters earlier than v1.17.17.) For details, see <a href="#">Example of Enabling Unsafe Sysctls in Pod Security Policy</a> .	Default: []	None

Item	Parameter	Description	Value	Modification
Node oversubscription	oversubscription-resource	Whether to enable node oversubscription. If this parameter is set to <b>true</b> , node oversubscription is enabled on nodes. For details, see <a href="#">Dynamic Resource Oversubscription</a> .	<ul style="list-style-type: none"> <li>For clusters of versions earlier than v1.23.9-r0 or v1.25.4-r0: enabled (<b>true</b>) by default</li> <li>Disabled by default if the cluster version is v1.23.9-r0, v1.25.4-r0, v1.27-r0, or later</li> </ul>	None
Hybrid deployment	colocation	Whether to enable hybrid deployment on nodes. If this parameter is set to <b>true</b> , hybrid deployment is enabled on nodes. For details, see <a href="#">Dynamic Resource Oversubscription</a> .	<ul style="list-style-type: none"> <li>For clusters of versions earlier than v1.23.9-r0 or v1.25.4-r0: enabled (<b>true</b>) by default</li> <li>Disabled by default if the cluster version is v1.23.9-r0, v1.25.4-r0, v1.27-r0, or later</li> </ul>	None

Item	Parameter	Description	Value	Modification
Topology management policy	topology-manager-policy	<p>Set the topology management policy. Valid values are as follows:</p> <ul style="list-style-type: none"> <li>• <b>restricted:</b> kubelet accepts only pods that achieve optimal NUMA alignment on the requested resources.</li> <li>• <b>best-effort:</b> kubelet preferentially selects pods that implement NUMA alignment on CPU and device resources.</li> <li>• <b>none (default):</b> The topology management policy is disabled.</li> <li>• <b>single-numa-node:</b> kubelet allows only pods that are aligned to the same NUMA node in terms of CPU and device resources.</li> </ul>	Default: none	<p><b>NOTICE</b> Modifying <b>topology-manager-policy</b> and <b>topology-manager-scope</b> will restart kubelet, and the resource allocation of pods will be recalculated based on the modified policy. In this case, running pods may restart or even fail to receive any resources.</p>
Topology management scope	topology-manager-scope	<p>Configure the resource alignment granularity of the topology management policy. Valid values are as follows:</p> <ul style="list-style-type: none"> <li>• <b>container (default)</b></li> <li>• <b>pod</b></li> </ul>	Default: container	

Item	Parameter	Description	Value	Modification
Specified DNS configuration file	resolv-conf	DNS resolution configuration file specified by the container	Default: null	None
Timeout for all runtime requests except long-running requests	runtime - request - timeout	Timeout interval of all runtime requests except long-running requests (pull, logs, exec, and attach).	Default: 2m0s	This parameter is available only in clusters of v1.21.10-r0, v1.23.8-r0, v1.25.3-r0, or later versions.
Whether to allow kubelet to pull only one image at a time	serialize-image-pulls	<p>Pull an image in serial mode.</p> <ul style="list-style-type: none"> <li><b>false:</b> recommended configuration so that an image can be pulled in parallel mode to improve pod startup.</li> <li><b>true:</b> allows images to be pulled in serial mode.</li> </ul>	<ul style="list-style-type: none"> <li>Enabled by default if the cluster version is earlier than v1.21.12-r0, v1.23.11-r0, v1.27.3-r0 or v1.25.6-r0</li> <li>Disabled by default if the cluster version is v1.21.12-r0, v1.23.11-r0, v1.25.6-r0, v1.27.3-r0, or later</li> </ul>	This parameter is available only in clusters of v1.21.10-r0, v1.23.8-r0, v1.25.3-r0, or later versions.
Image repository pull limit per second	registry-pull-qps	QPS upper limit of an image repository.	Default: 5 Value range: 1 to 50	This parameter is available only in clusters of v1.21.10-r0, v1.23.8-r0, v1.25.3-r0, or later versions.

Item	Parameter	Description	Value	Modification
Upper limit of burst image pull	registry-burst	Maximum number of burst image pulls.	Default: 10 The value ranges from 1 to 100 and must be greater than or equal to the value of <b>registry-pull-qps</b> .	This parameter is available only in clusters of v1.21.10-r0, v1.23.8-r0, v1.25.3-r0, or later versions.
Maximum Number of Container Log Files	container-log-max-files	Maximum number of container log files. When the number of existing log files exceeds this value, the earliest log file will be deleted to release space for new log files.	Default: 10 Value range: 2 to 100	This parameter is available only in clusters of v1.23.14-r0, v1.25.9-r0, v1.27.6-r0, v1.28.4-r0, or later versions.
Maximum Container Log File Size	container-log-max-size	Maximum size of a single container log file. When the size of a log file reaches this value, the current log file will be closed and a new log file will be created to continue logging.	Default: 50 Value range: 1 to 4096	This parameter is available only in clusters of v1.23.14-r0, v1.25.9-r0, v1.27.6-r0, v1.28.4-r0, or later versions.
Upper Limit for Image Garbage Collection	image-gc-high-threshold	When the kubelet disk usage reaches this value, kubelet starts to collect image garbage.	Default: 80 Value range: 1 to 100	To disable image garbage collection, set this parameter to <b>100</b> . This parameter is available only in clusters of v1.23.14-r0, v1.25.9-r0, v1.27.6-r0, v1.28.4-r0, or later versions.

Item	Parameter	Description	Value	Modification
Lower Limit for Image Garbage Collection	image-gc-low-threshold	When the disk usage reduces to this value, image garbage collection stops.	Default: 70 Value range: 1 to 100	The value of this parameter cannot be greater than the upper limit for image garbage collection.  This parameter is available only in clusters of v1.23.14-r0, v1.25.9-r0, v1.27.6-r0, v1.28.4-r0, or later versions.
Node memory reservation	system-reserved-mem	System memory reservation reserves memory resources for OS system daemons such as sshd and udev.	Default value: automatically calculated, which varies depending on node flavors. For details, see <a href="#">Node Resource Reservation Policy</a> .	The sum of <b>kube-reserved-mem</b> and <b>system-reserved-mem</b> must be less than 50% of the minimum memory of nodes in the node pool.
	kube-reserved-mem	Kubernetes memory reservation reserves memory resources for Kubernetes daemons such as kubelet and container runtime.		

## kube-proxy

Item	Parameter	Description	Value	Modification
Maximum number of connection tracking entries	conntrack-min	Maximum number of connection tracking entries  To obtain the value, run the following command: sysctl -w net.nf_conntrack_max	Default: 131072	None

Item	Parameter	Description	Value	Modification
Wait time of a closed TCP connection	contrack-tcp-timeout-close-wait	Wait time of a closed TCP connection  To obtain the value, run the following command: sysctl -w net.netfilter.nf_contrack_tcp_timeout_close_wait	Default: 1h0m0s	None

### Docker (Available Only for Docker Node Pools)

Item	Parameter	Description	Value	Modification
Container umask	native-umask	The default value <b>normal</b> indicates that the umask value of the started container is <b>0022</b> .	Default: normal	The parameter value cannot be changed.
Available data space for a single container	docker-base-size	Maximum data space that can be used by each container.	Default: 0	The parameter value cannot be changed.
Insecure image source address	insecure-registry	Whether an insecure image source address can be used.	false	The parameter value cannot be changed.
Maximum size of a container <b>core</b> file	limitcore	Maximum size of a core file in a container. The unit is byte.  If not specified, the value is <b>infinity</b> .	Default: 5368709120	None



Item	Parameter	Description	Value	Modification
Limit on the number of handles in a container	default-ulimit-nofile	Maximum number of handles that can be used in a container.	Default: {soft}:{hard}	The value cannot exceed the value of the kernel parameter <b>nr_open</b> and cannot be a negative number. You can run the following command to obtain the kernel parameter <b>nr_open</b> : sysctl -a   grep nr_open
Image pull timeout	image-pull-progress-timeout	If the image fails to be pulled before time outs, the image pull will be canceled.	Default: 1m0s	This parameter is supported in v1.25.3-r0 and later.
Maximum Number of Concurrent Requests for Downloading an Image at a Time	max-concurrent-downloads	This parameter specifies the maximum number of concurrent requests for downloading an image at a time.	Default: 3 Value range: 1 to 20	If this parameter is set to a large value, the network performance of other services on the node may be affected or the disk I/O and CPU usage may increase. This parameter is available only in clusters of v1.23.14-r0, v1.25.9-r0, v1.27.6-r0, v1.28.4-r0, or later versions.
Maximum Container Log File Size	max-size	Maximum size of a container log file to be dumped. When the size of a log file reaches this value, the current log file will be closed and a new log file will be created to continue logging.	Default: 50 Value range: 1 to 4096	If this parameter is set to a small value, important logs may be lost. If this parameter is set to a large value, too much disk space may be occupied. This parameter is available only in clusters of v1.23.14-r0, v1.25.9-r0, v1.27.6-r0, v1.28.4-r0, or later versions.

Item	Parameter	Description	Value	Modification
Maximum Number of Container Log Files	max-file	Maximum number of log files that can be retained in a container. When the number of existing log files exceeds this value, the earliest log file will be deleted to release space for new log files.	Default: 20 Value range: 2 to 100	If this parameter is set to a small value, important logs may be lost. If this parameter is set to a large value, too much disk space may be occupied.  This parameter is available only in clusters of v1.23.14-r0, v1.25.9-r0, v1.27.6-r0, v1.28.4-r0, or later versions.

### containerd (Available Only for containerd Node Pools)

Item	Parameter	Description	Value	Modification
Maximum size of a container <b>core</b> file	limitcore	Maximum size of a core file in a container. The unit is byte.  If not specified, the value is <b>infinity</b> .	Default: 5368709120	None
Limit on the number of handles in a container	default-ulimit-nofile	Maximum number of handles that can be used in a container.	Default: 1048576	The value cannot exceed the value of the kernel parameter <b>nr_open</b> and cannot be a negative number.  You can run the following command to obtain the kernel parameter <b>nr_open</b> : <code>sysctl -a   grep nr_open</code>
Image pull timeout	image-pull-progress-timeout	If the image fails to be pulled before time outs, the image pull will be canceled.	Default: 1m0s	This parameter is supported in v1.25.3-r0 and later.

Item	Parameter	Description	Value	Modification
Maximum Number of Concurrent Requests for Downloading an Image at a Time	max-concurrent-downloads	This parameter specifies the maximum number of concurrent requests for downloading an image at a time.	Default: 3 Value range: 1 to 20	If this parameter is set to a large value, the network performance of other services on the node may be affected or the disk I/O and CPU usage may increase.  This parameter is available only in clusters of v1.23.14-r0, v1.25.9-r0, v1.27.6-r0, v1.28.4-r0, or later versions.
Maximum Container Log Line Size	max-container-log-line-size	Maximum log line size of a container, in the unit of bytes. The log lines exceeding the limit will be split into multiple lines.	Default: 16384 Value range: 1 to 2097152	A larger value will lead to more containerd memory consumption.  This parameter is available only in clusters of v1.23.14-r0, v1.25.9-r0, v1.27.6-r0, v1.28.4-r0, or later versions.

### Networking Components (Available Only for CCE Turbo Clusters)

Item	Parameter	Description	Value	Modification
Node pool ENI pre-binding	enable-node-nic-configuration	Whether to enable ENI pre-binding in a node pool.	Default: false	After network component configuration is disabled in a node pool, the dynamic container ENI pre-binding parameter settings of the node pool are the same as those of cluster-level parameter settings.

Item	Parameter	Description	Value	Modification
ENI threshold	nic-threshold	Low threshold of the number of bound ENIs: High threshold of the number of bound ENIs	Default: 0:0	<b>NOTE</b> This parameter is being discarded. Use the dynamic pre-binding parameters of the other four ENIs.
Minimum number of ENIs bound to a node in a node pool	nic-minimum-target	Minimum number of container ENIs bound to a node. The parameter value must be a positive integer. The value <b>10</b> indicates that at least 10 container ENIs must be bound to a node. If the number you specified exceeds the container ENI quota of the node, the ENI quota will be used.	Default: 10	Configure these parameters based on the number of pods typically running on most nodes.

Item	Parameter	Description	Value	Modification
Maximum number of ENIs pre-bound to a node in a node pool	nic-maximum-target	<p>After the number of ENIs bound to a node exceeds the <b>nic-maximum-target</b> value, CCE will not proactively pre-bind ENIs.</p> <p>Checking the upper limit of pre-bound container ENIs is enabled only when the value of this parameter is greater than or equal to the minimum number of container ENIs (<b>nic-minimum-target</b>) bound to a node.</p> <p>The parameter value must be a positive integer. The value <b>0</b> indicates that checking the upper limit of pre-bound container ENIs is disabled. If the number you specified exceeds the container ENI quota of the node, the ENI quota will be used.</p>	Default: 0	Configure these parameters based on the maximum number of pods running on most nodes.

Item	Parameter	Description	Value	Modification
Number of ENIs dynamically pre-bound to a node in a node pool	nic-warm-target	<p>Extra ENIs will be pre-bound after the <b>nic-minimum-target</b> is used up in a pod. The value can only be a number.</p> <p>When the sum of the <b>nic-warm-target</b> value and the number of ENIs bound to the node is greater than the <b>nic-maximum-target</b> value, CCE will pre-bind the number of ENIs specified by the difference between the <b>nic-maximum-target</b> value and the current number of ENIs bound to the node.</p>	Default: 2	Set the parameter value to the number of pods that can be scaled out instantaneously within 10 seconds on most nodes.

Item	Parameter	Description	Value	Modification
Threshold for reclaiming the ENIs pre-bound to a node in a node pool	nic-max-above-warm-target	<p>Only when the difference between the number of idle ENIs on a node and the <b>nic-warm-target</b> value is greater than the threshold, the pre-bound ENIs will be unbound and reclaimed. The value can only be a number.</p> <ul style="list-style-type: none"> <li>• A large value will accelerate pod startup but slow down the unbinding of idle container ENIs and decrease the IP address usage. <b>Exercise caution when performing this operation.</b></li> <li>• A small value will speed up the unbinding of idle container ENIs and increase the IP address usage but will slow down pod startup, especially when a large number of pods increase instantaneously.</li> </ul>	Default: 2	Set the parameter value to 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.

## Pod Security Groups (Available Only for CCE Turbo Clusters)

Item	Parameter	Description	Value	Modification
Default security group used by pods in a node pool	security_groups_for_nodepool	<p>You can enter the security group ID. If this parameter is not configured, the default security group of the cluster container network will be used, and a maximum of five security group IDs that are separated by semicolons (;) can be specified at a time.</p> <p>The priority of the security group is lower than that of the security group configured for <a href="#">SecurityGroups</a>.</p>	None	None

### 4.5.4 Accepting Nodes in a Node Pool

If you want to add a newly purchased ECS to a node pool in a cluster, or remove a node from a node pool and add it to the node pool again, accept the node.

#### NOTICE

- When an ECS is accepted for management, the ECS OS will be reset to the standard image provided by CCE to ensure node stability.
- LVM information, including volume groups (VGs), logical volumes (LVs), and physical volumes (PVs), will be deleted from the system disks and data disks attached to the selected ECSs during acceptance. Ensure that the information has been backed up.
- During the acceptance of an ECS, do not perform any operation on the ECS through the ECS console.
- After a node is managed by a node pool, if a scaling-in task is triggered in the node pool, the node will be deleted.

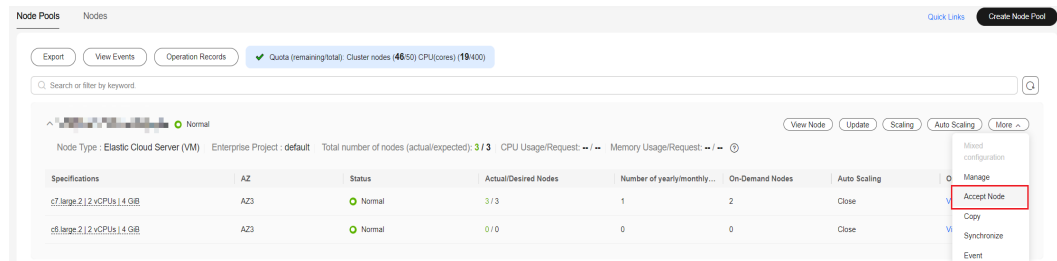
### Procedure

**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. In the right pane, click the **Node Pools** tab.

**Step 3** Choose **More > Accept Node** in the operation bar of the target node pool.



**Step 4** Select one or more nodes that meet the requirements.

- The nodes and the current node pool are deployed in the same VPC and subnet.
- The nodes and the current node pool belong to the same enterprise project.
- The billing mode of the nodes must be the same as that of the current node pool. For example, a pay-per-use node pool can accept only pay-per-use nodes.
- The cloud server group of the nodes must be the same as that of the current node pool.
- The nodes must be running and cannot be labeled with **CCE-Dynamic-Provisioning-Node**.
- Data disks must be attached to the nodes to be managed. A local disk (disk-intensive disk) or a data disk of at least 20 GiB can be attached to the node, and any data disks already attached cannot be smaller than 10 GiB.
- The flavor of the nodes must be at least 2 vCPUs and 4 GiB memory, and only one NIC can be bound to each of the nodes.
- Only the cloud servers that have the same flavor, AZ, resource reservation, system disk, and data disk configurations as the node pool can be added for batch acceptance.
- Partitioned disks on cloud servers will not be accepted as data disks. Back up data and clear the disks before node acceptance.

**Step 5** Click **OK**.

----End

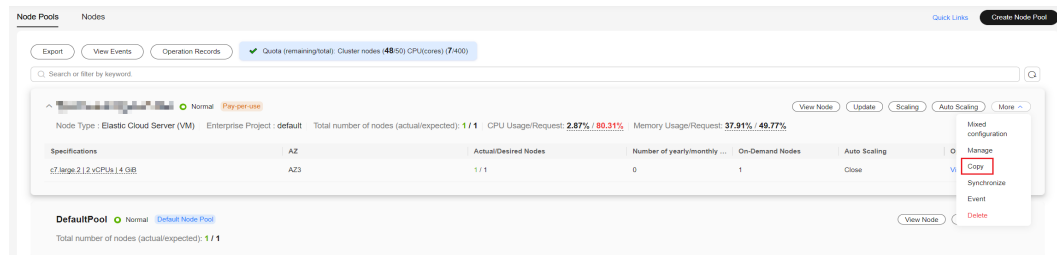
## 4.5.5 Copying a Node Pool

You can copy the configuration of an existing node pool on the CCE console to create new node pools.

**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. In the right pane, click the **Node Pools** tab.

**Step 3** Choose **More > Copy** in the **Operation** column of the target node pool.

**Figure 4-1** Copying a node pool

**Step 4** In the **Copy Resource Pool** window, the configurations of the node pool to be copied are displayed. Modify the configurations as needed. For details, see [Creating a Node Pool](#). After confirming the configuration, click **Next: Confirm**.

**Step 5** On the **Confirm** page, confirm the node pool configurations and click **Submit**. Then, a new node pool is created based on the modified configurations.

----End

## 4.5.6 Synchronizing Node Pools

After the configuration of a node pool is updated, some configurations cannot be automatically synchronized for existing nodes. You can manually synchronize configurations for these nodes.

### NOTICE

- Do not delete or reset nodes during batch synchronization. Otherwise, the synchronization of node pool configuration may fail.
- This operation involves resetting nodes. **Workloads running on a node may be interrupted due to standalone deployment or insufficient schedulable resources.** Evaluate the upgrade risks and perform the upgrade during off-peak hours. Alternatively, [specify a disruption budget for your key applications](#) to ensure the availability of these applications during the upgrade.
- During configuration synchronization for existing nodes, the nodes will be reset, and the system disks and data disks will be cleared. Back up important data before the synchronization.
- Only some node pool parameters can be synchronized by resetting nodes. The constraints are as follows:
  - Changes to the container engine, OS, or pre-/post-installation script in a node pool take effect only on new nodes. To synchronize the modification onto existing nodes, manually reset these nodes.
  - Changes to resource tags and Kubernetes labels/taints in a node pool will be automatically synchronized to existing nodes after the options of **Synchronization for Existing Nodes** are selected. You do not need to reset these nodes.

## Synchronizing a Single Node

**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. In the right pane, click the **Nodes** tab.

**Step 3** Find **upgrade** in the **Node Pool** column of the existing nodes in the node pool.



**Step 4** Click **update**. In the dialog box that is displayed, confirm whether to reset the node immediately.

----End

## Batch Synchronization

**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. In the right pane, click the **Node Pools** tab.

**Step 3** Choose **More > Synchronize** in the **Operation** column of the target node pool.

**Step 4** In the **Batch synchronization** window, configure parameters.

- **OS:** shows the image of the target version. You do not need to configure this parameter.
- **Synchronization Policy:** **Node Reset** is supported.
- **Max. Nodes for Batch Synchronize:** maximum number of nodes that will be unavailable during node synchronization. Nodes will be unavailable during synchronization by resetting the nodes. Properly configure this parameter to prevent pod scheduling failures caused by too many unavailable nodes in the cluster.
- **Node List:** Select the nodes requiring the synchronization of node pool configurations.

**Step 5** Click **OK**.

----End

## 4.5.7 Upgrading an OS

After CCE releases a new OS image, if existing nodes cannot be automatically upgraded, you can manually upgrade them in batches.

### Precautions

- This section describes how to upgrade an OS by resetting the target node. **Workloads running on a node may be interrupted due to standalone deployment or insufficient schedulable resources.** Evaluate the upgrade risks and perform the upgrade during off-peak hours. Alternatively, **specify a disruption budget for your key applications** to ensure the availability of these applications during the upgrade.

- The system disk and data disks of the node will be cleared. **Back up important data** before resetting the node.
- Resetting a node will clear your configured Kubernetes labels and taints. As a result, resources (such as local storage and load of specified scheduling nodes) associated with the node may be unavailable. Exercise caution when performing this operation to avoid impact on running services.
- After the OS is upgraded, the node automatically starts.
- To ensure node stability, CCE will reserve some CPU and memory resources to run necessary system components.

## Notes and Constraints

- Nodes running private images cannot be upgraded.
- Compatibility issues may occur when the node OS of an early version is upgraded. In this case, manually reset the node for the OS upgrade.

## Procedure for Default Node Pools

**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. In the right pane, click the **Node Pools** tab.

**Step 3** Click **Upgrade** next to the default node pool.

**Step 4** In the displayed **Operating System Upgrade** window, configure parameters.

- **Target Operating System:** shows the image of the target version. You do not need to configure this parameter.
- **Upgrade Policy: Node Reset** is supported.
- **Max. Nodes for Batch Upgrade:** maximum number of nodes that will be unavailable during node upgrade. Nodes will be unavailable during synchronization by resetting the nodes. Properly configure this parameter to prevent pod scheduling failures caused by too many unavailable nodes in the cluster.
- **View Node:** Select the nodes to be upgraded.
- Login Mode:
- Pre-installation script:  
Installation script command, in which Chinese characters are not allowed. The script command will be Base64-transcoded.  
The script will be executed before Kubernetes software is installed. Note that if the script is incorrect, Kubernetes software may fail to be installed.
- Post-installation script:  
Installation script command, in which Chinese characters are not allowed. The script command will be Base64-transcoded.  
The script will be executed after Kubernetes software is installed, which does not affect the installation.

**Step 5** Click **OK**.

----End

## Procedure for Non-default Node Pools

**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. In the right pane, click the **Node Pools** tab.

**Step 3** Choose **More > Synchronize** in the **Operation** column of the target node pool.

**Step 4** In the **Batch synchronization** window, configure parameters.

- **OS:** shows the image of the target version. You do not need to configure this parameter.
- **Synchronization Policy:** **Node Reset** is supported.
- **Max. Nodes for Batch Synchronize:** maximum number of nodes that will be unavailable during node synchronization. Nodes will be unavailable during synchronization by resetting the nodes. Properly configure this parameter to prevent pod scheduling failures caused by too many unavailable nodes in the cluster.
- **Node List:** Select the nodes requiring the synchronization of node pool configurations.

**Step 5** Click **OK**.

----End

## 4.5.8 Migrating a Node

### Migrate nodes from a custom node pool to the default node pool.

**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 **Nodes** and click the **Node Pools** tab.

**Step 3** Click **View Node** in the **Operation** column of the node pool to be migrated.

**Step 4** Choose **More > Migrate** in the **Operation** column of the target node to migrate the node.

**Figure 4-2** Migrating nodes to the default node pool

Node Name	Status	Node Pool	Configurations	IP Address	Pods (Alloc./RequestLimit)	CPU RequestLimit	Memory RequestLimit	Runtime Version & OS Version	Billing Mode	Operation
test-kyk4	Running Schedulable	test	AZ3 m5.large.2 2vCPUs   4GB	192.168.2.49 (Private)	2 / 20 41.45% 46.63%	15.53% 15.53%	15.53%	docker://19.9.0 EulerOS 2.0 (SP9KB5)	Pay-per-use Dec 03, 2023 16:30:55 GMT+08:00	Monitor View Events More
test-patch-78085-kidrg	Running Schedulable	DefaultPool	AZ3 m5.large.2 2vCPUs   4GB	192.168.2.53 (Private)	5 / 20 72.54% 124.25%	73.58% 105.00%	105.00%	container://1.4.1-112 EulerOS 2.0 (SP9KB5)	Pay-per-use Nov 30, 2023 10:59:13 GMT+08:00	View Pods View YAML Reset Node
test-patch-78085-avm3	Running Schedulable	DefaultPool	AZ3 m5.large.2 2vCPUs   4GB	192.168.2.17 (Private)	4 / 20 80.31% 85.49%	83.87% 83.87%	83.87%	container://1.4.1-112 EulerOS 2.0 (SP9KB5)	Pay-per-use Nov 30, 2023 10:59:13 GMT+08:00	Monitor Double Scheduling Sync Server Data
test-patch-41571-ep5hm	Running Schedulable	DefaultPool	AZ3 m5.large.2 2vCPUs   4GB	192.168.2.18 (Private)	5 / 20 53.26% 98.49%	87% 87%	87%	container://1.4.1-112 EulerOS 2.0 (SP9KB5)	Pay-per-use Nov 30, 2023 10:39:05 GMT+08:00	Manage Taint Manage Label Forbid node pool scale-in Remove Migrate Delete

**Step 5** In the displayed **Migrate Node** dialog box, confirm the information.

 NOTE

- The migration does not affect custom resource tags, Kubernetes labels, and taints of the node.
- After the migration, system labels **cce.cloud.com** and **cce-nodepool** on the node will be deleted. If an existing workload uses these labels for affinity or anti-affinity scheduling, the existing pods on the node will be stopped and rescheduled when kubelet is restarted.

----End

## 4.5.9 Deleting a Node Pool

Deleting a node pool will delete nodes in the pool. Pods on these nodes will be automatically migrated to available nodes in other node pools.

### Notes and Constraints

- Before deleting a yearly/monthly-billed node pool, remove all nodes in it.

### Precautions

- Deleting a node pool will delete all nodes in the node pool. Back up data in a timely manner to prevent data loss.
- Deleting a node will lead to pod migration, which may affect services. Perform this operation during off-peak hours. If pods in the node pool have a specific node selector and none of the other nodes in the cluster satisfies the node selector, the pods will become unschedulable.
- When deleting a node pool, the system sets all nodes in the current node pool to the unschedulable state.

### Procedure

**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. In the right pane, click the **Node Pools** tab.

**Step 3** Choose **More > Delete** in the **Operation** column of the target node pool.

**Step 4** Read the precautions in the **Delete Node Pool** dialog box.

**Step 5** In the text box, enter **DELETE** and click **Yes** to confirm that you want to continue the deletion.

----End

# 5 Workloads

---

## 5.1 Overview

A workload is an application running on Kubernetes. No matter how many components are there in your workload, you can run it in a group of Kubernetes pods. A workload is an abstract model of a group of pods in Kubernetes. Workloads in Kubernetes are classified as Deployments, StatefulSets, DaemonSets, jobs, and cron jobs.

CCE provides Kubernetes-native container deployment and management and supports lifecycle management of container workloads, including creation, configuration, monitoring, auto scaling, upgrade, uninstall, service discovery, and load balancing.

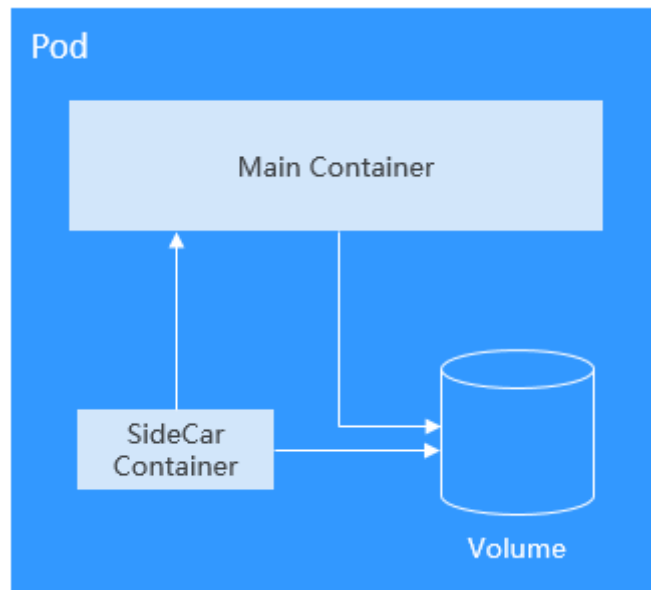
### Overview of Pod

A pod is the smallest and simplest unit in the Kubernetes object model that you create or deploy. A pod is a group of one or more containers, with shared storage and network resources, and a specification for how to run the containers. Each pod has a separate IP address.

Pods can be used in either of the following ways:

- A pod runs only one container. This is the most common usage of pods in Kubernetes. You can consider a pod as a container, but Kubernetes directly manages pods instead of containers.
- A pod runs multiple containers that need to be tightly coupled. In this scenario, a pod contains a main container and several sidecar containers, as shown in [Figure 5-1](#). For example, the main container is a web server that provides file services from a fixed directory, and sidecar containers periodically download files to this fixed directory.

**Figure 5-1** Pod running multiple containers

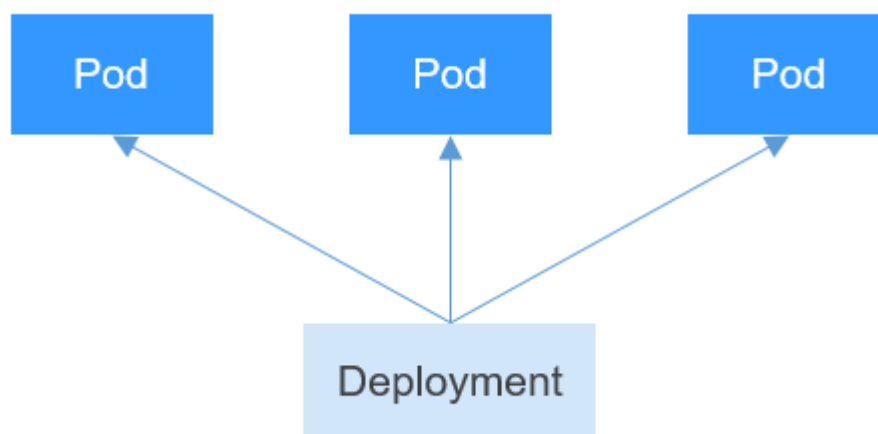


In Kubernetes, pods are rarely created directly. Instead, Kubernetes controller manages pods through pod instances such as Deployments and jobs. A controller typically uses a pod template to create pods. The controller can also manage multiple pods and provide functions such as replica management, rolling upgrade, and self-healing.

## Overview of Deployment

A pod is the smallest and simplest unit that you create or deploy in Kubernetes. It is designed to be an ephemeral, one-off entity. A pod can be evicted when node resources are insufficient and disappears along with a cluster node failure. Kubernetes provides controllers to manage pods. Controllers can create and manage pods, and provide replica management, rolling upgrade, and self-healing capabilities. The most commonly used controller is Deployment.

**Figure 5-2** Relationship between a Deployment and pods





A Deployment can contain one or more pods. These pods have the same role. Therefore, the system automatically distributes requests to multiple pods of a Deployment.

A Deployment integrates a lot of functions, including online deployment, rolling upgrade, replica creation, and restoration of online jobs. To some extent, Deployments can be used to realize unattended rollout, which greatly reduces difficulties and operation risks in the rollout process.

## Overview of StatefulSet

All pods under a Deployment have the same characteristics except for the name and IP address. If required, a Deployment can use a pod template to create new pods. If not required, the Deployment can delete any one of the pods.

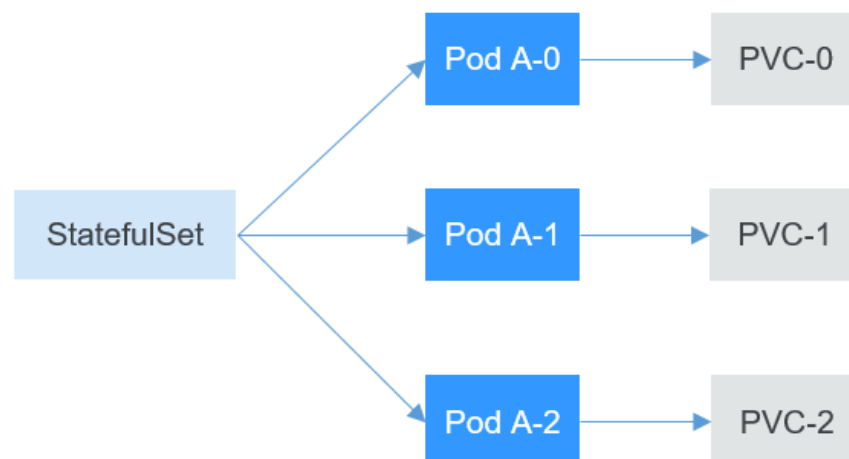
However, Deployments cannot meet the requirements in some distributed scenarios when each pod requires its own status or in a distributed database where each pod requires independent storage.

Distributed stateful applications involve different roles for different responsibilities. For example, databases work in active/standby mode, and pods depend on each other. To deploy stateful applications in Kubernetes, ensure pods meet the following requirements:

- Each pod must have a fixed identifier so that it can be recognized by other pods.
- Separate storage resources must be configured for each pod. In this way, the original data can be retrieved after a pod is deleted and restored. Otherwise, the pod status will be changed after the pod is rebuilt.

To address the preceding requirements, Kubernetes provides StatefulSets.

1. StatefulSets provide a fixed name for each pod following a fixed number ranging from 0 to N. After a pod is rescheduled, the pod name and the hostname remain unchanged.
2. StatefulSets use a headless Service to allocate a fixed domain name for each pod.
3. StatefulSets create PersistentVolumeClaims (PVCs) with fixed identifiers to ensure that pods can access the same persistent data after being rescheduled.

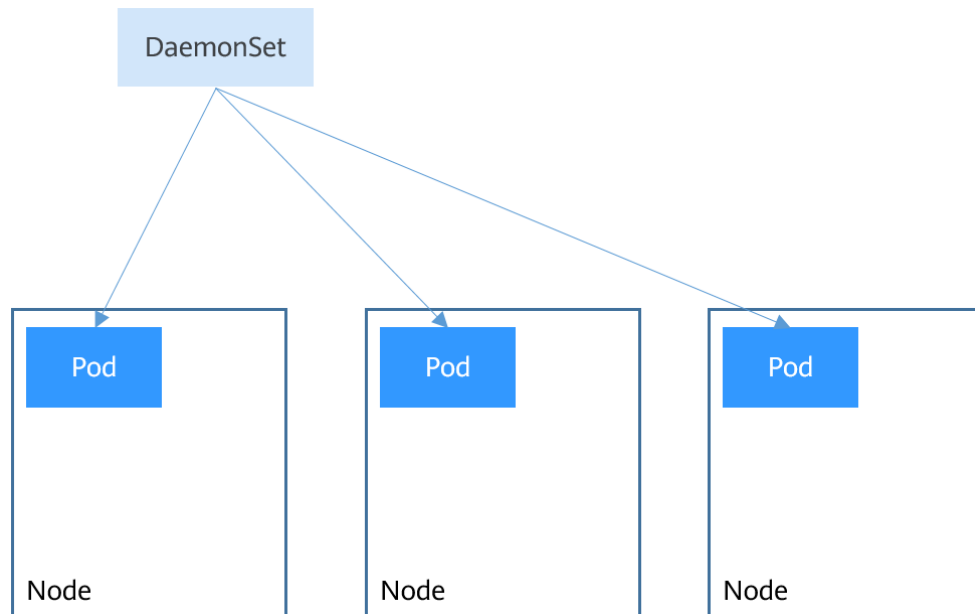


## Overview of DaemonSet

A DaemonSet runs a pod on each node in a cluster and ensures that there is only one pod. This works well for certain system-level applications such as log collection and resource monitoring since they must run on each node and need only a few pods. A good example is kube-proxy.

DaemonSets are closely related to nodes. If a node becomes faulty, the DaemonSet will not create the same pods on other nodes.

**Figure 5-3** DaemonSet



## Overview of Job and CronJob

Jobs and CronJobs allow you to run short lived, one-off tasks in batch. They ensure the task pods run to completion.

- A job is a resource object used by Kubernetes to control batch tasks. Jobs are different from long-term servo tasks (such as Deployments and StatefulSets). The former is started and terminated at specific times, while the latter runs unceasingly unless being terminated. The pods managed by a job will be automatically removed after successfully completing tasks based on user configurations.
- A CronJob runs a job periodically on a specified schedule. A CronJob object is similar to a line of a crontab file in Linux.

This run-to-completion feature of jobs is especially suitable for one-off tasks, such as continuous integration (CI).

## Workload Lifecycle

Table 5-1 Status description

Status	Description
Running	All pods are running or the number of pods is 0.
Unready	The container malfunctions and the pod under the workload is not working.
Processing	The workload is not running but no error is reported.
Available	For a multi-pod Deployment, some pods are abnormal but at least one pod is available.
Completed	The task is successfully executed. This status is available only for common tasks.
Stopped	The workload is stopped and the number of pods changes to 0. This status is available for workloads earlier than v1.13.
Deleting	The workload is being deleted.

## 5.2 Creating a Workload

### 5.2.1 Creating a Deployment

#### Scenario

Deployments are workloads (for example, Nginx) that do not store any data or status. You can create Deployments on the CCE console or by running `kubectl` commands.

#### Prerequisites

- Before creating a workload, you must have an available cluster. For details on how to create a cluster, see [Buying a CCE Standard/Turbo Cluster](#).
- To enable public access to a workload, ensure that an EIP or load balancer has been bound to at least one node in the cluster.

#### NOTE

If a pod has multiple containers, ensure that the ports used by the containers do not conflict with each other. Otherwise, creating the Deployment will fail.

#### Using the CCE Console

**Step 1** Log in to the CCE console.

**Step 2** Click the cluster name to go to the cluster console, choose **Workloads** in the navigation pane, and click the **Create Workload** in the upper right corner.

**Step 3** Set basic information about the workload.

**Basic Info**

- **Workload Type:** Select **Deployment**. For details about workload types, see [Overview](#).
- **Workload Name:** Enter the name of the workload. Enter 1 to 63 characters starting with a lowercase letter and ending with a lowercase letter or digit. Only lowercase letters, digits, and hyphens (-) are allowed.
- **Namespace:** Select the namespace of the workload. The default value is **default**. You can also click **Create Namespace** to create one. For details, see [Creating a Namespace](#).
- **Pods:** Enter the number of pods of the workload.
- **Time Zone Synchronization:** Specify whether to enable time zone synchronization. After time zone synchronization is enabled, the container and node use the same time zone. The time zone synchronization function depends on the local disk mounted to the container. Do not modify or delete the time zone. For details, see [Configuring Time Zone Synchronization](#).

**Container Settings**

- Container Information

Multiple containers can be configured in a pod. You can click **Add Container** on the right to configure multiple containers for the pod.

- **Basic Info:** Configure basic information about the container.

Parameter	Description
Container Name	Name the container.
Pull Policy	Image update or pull policy. If you select <b>Always</b> , the image is pulled from the image repository each time. If you do not select <b>Always</b> , the existing image of the node is preferentially used. If the image does not exist, the image is pulled from the image repository.
Image Name	Click <b>Select Image</b> and select the image used by the container. To use a third-party image, see <a href="#">Using Third-Party Images</a> .
Image Tag	Select the image tag to be deployed.
CPU Quota	<ul style="list-style-type: none"> <li>▪ <b>Request:</b> minimum number of CPU cores required by a container. The default value is 0.25 cores.</li> <li>▪ <b>Limit:</b> maximum number of CPU cores that can be used by a container. This prevents containers from using excessive resources.</li> </ul> <p>If <b>Request</b> and <b>Limit</b> are not specified, the quota is not limited. For more information and suggestions about <b>Request</b> and <b>Limit</b>, see <a href="#">Configuring Container Specifications</a>.</p>

Parameter	Description
Memory Quota	<ul style="list-style-type: none"> <li>▪ <b>Request:</b> minimum amount of memory required by a container. The default value is 512 MiB.</li> <li>▪ <b>Limit:</b> maximum amount of memory available for a container. When memory usage exceeds the specified memory limit, the container will be terminated.</li> </ul> <p>If <b>Request</b> and <b>Limit</b> are not specified, the quota is not limited. For more information and suggestions about <b>Request</b> and <b>Limit</b>, see <a href="#">Configuring Container Specifications</a>.</p>
(Optional) GPU Quota	<p>Configurable only when the cluster contains GPU nodes and the <a href="#">CCE AI Suite (NVIDIA GPU)</a> add-on is installed.</p> <ul style="list-style-type: none"> <li>▪ <b>All:</b> No GPU will be used.</li> <li>▪ <b>Dedicated:</b> GPU resources are dedicated for the container.</li> <li>▪ <b>Shared:</b> percentage of GPU resources used by the container. For example, if this parameter is set to <b>10%</b>, the container uses 10% of GPU resources.</li> </ul> <p>For details about how to use GPUs in the cluster, see <a href="#">Default GPU Scheduling in Kubernetes</a>.</p>
(Optional) Privileged Container	<p>Programs in a privileged container have certain privileges.</p> <p>If <b>Privileged Container</b> is enabled, the container is assigned privileges. For example, privileged containers can manipulate network devices on the host machine and modify kernel parameters.</p>
(Optional) Init Container	<p>Whether to use the container as an init container. An init container does not support health check.</p> <p>An init container is a special container that runs before other app containers in a pod are started. Each pod can contain multiple containers. In addition, a pod can contain one or more init containers. Application containers in a pod are started and run only after the running of all init containers completes. For details, see <a href="#">Init Containers</a>.</p>

- (Optional) **Lifecycle:** Configure operations to be performed in a specific phase of the container lifecycle, such as Startup Command, Post-Start, and Pre-Stop. For details, see [Configuring Container Lifecycle Parameters](#).

- (Optional) **Health Check**: Set the liveness probe, ready probe, and startup probe as required. For details, see [Configuring Container Health Check](#).
- (Optional) **Environment Variables**: Configure variables for the container running environment using key-value pairs. These variables transfer external information to containers running in pods and can be flexibly modified after application deployment. For details, see [Configuring Environment Variables](#).
- (Optional) **Data Storage**: Mount local storage or cloud storage to the container. The application scenarios and mounting modes vary with the storage type. For details, see [Storage](#).

#### NOTE

If the workload contains more than one pod, EVS volumes cannot be mounted.

- (Optional) **Security Context**: Assign container permissions to protect the system and other containers from being affected. Enter the user ID to assign container permissions and prevent systems and other containers from being affected.
- (Optional) **Logging**: Report standard container output logs to AOM by default, without requiring manual settings. You can manually configure the log collection path. For details, see [Collecting Container Logs Using ICAgent](#).

To disable the standard output of the current workload, add the annotation `kubernetes.AOM.log.stdout: []` in [Labels and Annotations](#). For details about how to use this annotation, see [Table 5-14](#).

- **Image Access Credential**: Select the credential used for accessing the image repository. The default value is `default-secret`. You can use `default-secret` to access images in SWR. For details about `default-secret`, see [default-secret](#).
- (Optional) **GPU: All** is selected by default. The workload instance will be scheduled to the node of the specified GPU type.

#### (Optional) Service Settings

A Service provides external access for pods. With a static IP address, a Service forwards access traffic to pods and automatically balances load for these pods.

You can also create a Service after creating a workload. For details about Services of different types, see [Overview](#).

#### (Optional) Advanced Settings

- **Upgrade**: Specify the upgrade mode and parameters of the workload. **Rolling upgrade** and **Replace upgrade** are available. For details, see [Configuring Workload Upgrade Policies](#).
- **Scheduling**: Configure affinity and anti-affinity policies for flexible workload scheduling. Load affinity and node affinity are provided.
  - **Load Affinity**: Common load affinity policies are offered for quick load affinity deployment.
    - **Not configured**: No load affinity policy is configured.
    - **Multi-AZ deployment preferred**: Workload pods are **preferentially** scheduled to nodes in different AZs through pod anti-affinity.

- **Forcible multi-AZ deployment:** Workload pods are forcibly scheduled to nodes in different AZs through pod anti-affinity (**podAntiAffinity**). If there are fewer AZs than pods, the extra pods will fail to run.
- **Custom policies:** Affinity and anti-affinity policies can be customized. For details, see [Configuring Workload Affinity or Anti-affinity Scheduling \(podAffinity or podAntiAffinity\)](#).
- **Node Affinity:** Common load affinity policies are offered for quick load affinity deployment.
  - **Not configured:** No node affinity policy is configured.
  - **Node Affinity:** Workload pods can be deployed on specified nodes through node affinity (**nodeAffinity**). If no node is specified, the pods will be randomly scheduled based on the default scheduling policy of the cluster.
  - **Specified node pool scheduling:** Workload pods can be deployed in a specified node pool through node affinity (**nodeAffinity**). If no node pool is specified, the pods will be randomly scheduled based on the default scheduling policy of the cluster.
  - **Custom policies:** Affinity and anti-affinity policies can be customized. For details, see [Configuring Node Affinity Scheduling \(nodeAffinity\)](#).
- **Toleration:** Using both taints and tolerations allows (not forcibly) the pod to be scheduled to a node with the matching taints, and controls the pod eviction policies after the node where the pod is located is tainted. For details, see [Configuring Tolerance Policies](#).
- **Labels and Annotations:** Add labels or annotations for pods using key-value pairs. After entering the key and value, click **Confirm**. For details about how to use and configure labels and annotations, see [Configuring Labels and Annotations](#).
- **DNS:** Configure a separate DNS policy for the workload. For details, see [DNS Configuration](#).

**Step 4** Click **Create Workload** in the lower right corner.

----End

## Using kubectl

The following procedure uses Nginx as an example to describe how to create a workload using kubectl.

**Step 1** Use kubectl to access the cluster. For details, see [Connecting to a Cluster Using kubectl](#).

**Step 2** Create and edit the **nginx-deployment.yaml** file. **nginx-deployment.yaml** is an example file name, and you can rename it as required.

```
vi nginx-deployment.yaml
```

The following is an example YAML file. For more information about Deployments, see [Kubernetes documentation](#).

```
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 you use an image in My Images, obtain the image path from SWR.
          imagePullPolicy: Always
          name: nginx
          imagePullSecrets:
            - name: default-secret
```

For details about the parameters, see [Table 5-2](#).

**Table 5-2** Deployment YAML parameters

Parameter	Description	Mandatory/Optional
apiVersion	API version. <b>NOTE</b> Set this parameter based on the cluster version. <ul style="list-style-type: none"> <li>For clusters of v1.17 or later, the apiVersion format of Deployments is <b>apps/v1</b>.</li> <li>For clusters of v1.15 or earlier, the apiVersion format of Deployments is <b>extensions/v1beta1</b>.</li> </ul>	Mandatory
kind	Type of a created object.	Mandatory
metadata	Metadata of a resource object.	Mandatory
name	Name of the Deployment.	Mandatory
spec	Detailed description of the Deployment.	Mandatory
replicas	Number of pods.	Mandatory
selector	Determines container pods that can be managed by the Deployment.	Mandatory



Parameter	Description	Mandatory/Optional
strategy	Upgrade mode. Possible values: <ul style="list-style-type: none"> <li>RollingUpdate</li> <li>ReplaceUpdate</li> </ul> By default, rolling update is used.	Optional
template	Detailed description of a created container pod.	Mandatory
metadata	Metadata.	Mandatory
labels	<b>metadata.labels</b> : Container labels.	Optional
spec: containers	<ul style="list-style-type: none"> <li><b>image</b> (mandatory): Name of a container image.</li> <li><b>imagePullPolicy</b> (optional): Policy for obtaining an image. The options include <b>Always</b> (attempting to download images each time), <b>Never</b> (only using local images), and <b>IfNotPresent</b> (using local images if they are available; downloading images if local images are unavailable). The default value is <b>Always</b>.</li> <li><b>name</b> (mandatory): Container name.</li> </ul>	Mandatory
imagePullSecrets	Name of the secret used during image pulling. If a private image is used, this parameter is mandatory. <ul style="list-style-type: none"> <li>To pull an image from the Software Repository for Container (SWR), set this parameter to <b>default-secret</b>.</li> <li>To pull an image from a third-party image repository, set this parameter to the name of the created secret.</li> </ul>	Optional

**Step 3** Create a Deployment.

**kubectl create -f nginx-deployment.yaml**

If the following information is displayed, the Deployment is being created.

```
deployment "nginx" created
```

**Step 4** Obtain the Deployment status.

**kubectl get deployment**

If the following information is displayed, the Deployment is running.

```
NAME      READY   UP-TO-DATE   AVAILABLE   AGE
nginx     1/1     1             1           4m5s
```

**Parameter description**

- **NAME:** Name of the application running in the pod.
- **READY:** indicates the number of available workloads. The value is displayed as "the number of available pods/the number of expected pods".
- **UP-TO-DATE:** indicates the number of replicas that have been updated.
- **AVAILABLE:** indicates the number of available pods.
- **AGE:** period the Deployment keeps running

**Step 5** If the Deployment will be accessed through a ClusterIP or NodePort Service, configure the access mode. For details, see [Network](#).

----End

## 5.2.2 Creating a StatefulSet

### Scenario

StatefulSets are a type of workloads whose data or status is stored while they are running. For example, MySQL is a StatefulSet because it needs to store new data.

A container can be migrated between different hosts, but data is not stored on the hosts. To store StatefulSet data persistently, attach HA storage volumes provided by CCE to the container.

### Notes and Constraints

- When you delete or scale a StatefulSet, the system does not delete the storage volumes associated with the StatefulSet to ensure data security.
- When you delete a StatefulSet, reduce the number of replicas to **0** before deleting the StatefulSet so that pods in the StatefulSet can be stopped in order.
- When you create a StatefulSet, a headless Service is required for pod access. For details, see [Headless Services](#).
- When a node is unavailable, pods become **Unready**. In this case, manually delete the pods of the StatefulSet so that the pods can be migrated to a normal node.

### Prerequisites

- Before creating a workload, you must have an available cluster. For details on how to create a cluster, see [Buying a CCE Standard/Turbo Cluster](#).
- To enable public access to a workload, ensure that an EIP or load balancer has been bound to at least one node in the cluster.

#### NOTE

If a pod has multiple containers, ensure that the ports used by the containers do not conflict with each other. Otherwise, creating the StatefulSet will fail.

## Using the CCE Console

**Step 1** Log in to the CCE console.

**Step 2** Click the cluster name to go to the cluster console, choose **Workloads** in the navigation pane, and click the **Create Workload** in the upper right corner.

**Step 3** Set basic information about the workload.

**Basic Info**

- **Workload Type:** Select **StatefulSet**. For details about workload types, see [Overview](#).
- **Workload Name:** Enter the name of the workload. Enter 1 to 63 characters starting with a lowercase letter and ending with a lowercase letter or digit. Only lowercase letters, digits, and hyphens (-) are allowed.
- **Namespace:** Select the namespace of the workload. The default value is **default**. You can also click **Create Namespace** to create one. For details, see [Creating a Namespace](#).
- **Pods:** Enter the number of pods of the workload.
- **Time Zone Synchronization:** Specify whether to enable time zone synchronization. After time zone synchronization is enabled, the container and node use the same time zone. The time zone synchronization function depends on the local disk mounted to the container. Do not modify or delete the time zone. For details, see [Configuring Time Zone Synchronization](#).

**Container Settings**

- Container Information

Multiple containers can be configured in a pod. You can click **Add Container** on the right to configure multiple containers for the pod.

- **Basic Info:** Configure basic information about the container.

Parameter	Description
Container Name	Name the container.
Pull Policy	Image update or pull policy. If you select <b>Always</b> , the image is pulled from the image repository each time. If you do not select <b>Always</b> , the existing image of the node is preferentially used. If the image does not exist, the image is pulled from the image repository.
Image Name	Click <b>Select Image</b> and select the image used by the container. To use a third-party image, see <a href="#">Using Third-Party Images</a> .
Image Tag	Select the image tag to be deployed.
CPU Quota	<ul style="list-style-type: none"> <li>▪ <b>Request:</b> minimum number of CPU cores required by a container. The default value is 0.25 cores.</li> <li>▪ <b>Limit:</b> maximum number of CPU cores that can be used by a container. This prevents containers from using excessive resources.</li> </ul> <p>If <b>Request</b> and <b>Limit</b> are not specified, the quota is not limited. For more information and suggestions about <b>Request</b> and <b>Limit</b>, see <a href="#">Configuring Container Specifications</a>.</p>

Parameter	Description
Memory Quota	<ul style="list-style-type: none"> <li>▪ <b>Request:</b> minimum amount of memory required by a container. The default value is 512 MiB.</li> <li>▪ <b>Limit:</b> maximum amount of memory available for a container. When memory usage exceeds the specified memory limit, the container will be terminated.</li> </ul> <p>If <b>Request</b> and <b>Limit</b> are not specified, the quota is not limited. For more information and suggestions about <b>Request</b> and <b>Limit</b>, see <a href="#">Configuring Container Specifications</a>.</p>
(Optional) GPU Quota	<p>Configurable only when the cluster contains GPU nodes and the <a href="#">CCE AI Suite (NVIDIA GPU)</a> add-on is installed.</p> <ul style="list-style-type: none"> <li>▪ <b>All:</b> No GPU will be used.</li> <li>▪ <b>Dedicated:</b> GPU resources are dedicated for the container.</li> <li>▪ <b>Shared:</b> percentage of GPU resources used by the container. For example, if this parameter is set to <b>10%</b>, the container uses 10% of GPU resources.</li> </ul> <p>For details about how to use GPUs in the cluster, see <a href="#">Default GPU Scheduling in Kubernetes</a>.</p>
(Optional) Privileged Container	<p>Programs in a privileged container have certain privileges.</p> <p>If <b>Privileged Container</b> is enabled, the container is assigned privileges. For example, privileged containers can manipulate network devices on the host machine and modify kernel parameters.</p>
(Optional) Init Container	<p>Whether to use the container as an init container. An init container does not support health check.</p> <p>An init container is a special container that runs before other app containers in a pod are started. Each pod can contain multiple containers. In addition, a pod can contain one or more init containers. Application containers in a pod are started and run only after the running of all init containers completes. For details, see <a href="#">Init Containers</a>.</p>

- (Optional) **Lifecycle:** Configure operations to be performed in a specific phase of the container lifecycle, such as Startup Command, Post-Start, and Pre-Stop. For details, see [Configuring Container Lifecycle Parameters](#).

- (Optional) **Health Check**: Set the liveness probe, ready probe, and startup probe as required. For details, see [Configuring Container Health Check](#).
- (Optional) **Environment Variables**: Configure variables for the container running environment using key-value pairs. These variables transfer external information to containers running in pods and can be flexibly modified after application deployment. For details, see [Configuring Environment Variables](#).
- (Optional) **Data Storage**: Mount local storage or cloud storage to the container. The application scenarios and mounting modes vary with the storage type. For details, see [Storage](#).

#### NOTE

- StatefulSets support dynamic attachment of EVS disks. For details, see [Dynamically Mounting an EVS Disk to a StatefulSet](#) or [Dynamically Mounting a Local PV to a StatefulSet](#).  
Dynamic mounting is achieved by using the `volumeClaimTemplates` field and depends on the dynamic creation capability of StorageClass. A StatefulSet associates each pod with a PVC using the `volumeClaimTemplates` field, and the PVC is bound to the corresponding PV. Therefore, after the pod is rescheduled, the original data can still be mounted based on the PVC name.
- After a workload is created, the storage that is dynamically mounted cannot be updated.
- (Optional) **Security Context**: Assign container permissions to protect the system and other containers from being affected. Enter the user ID to assign container permissions and prevent systems and other containers from being affected.
- (Optional) **Logging**: Report standard container output logs to AOM by default, without requiring manual settings. You can manually configure the log collection path. For details, see [Collecting Container Logs Using ICAgent](#).  
To disable the standard output of the current workload, add the annotation `kubernetes.AOM.log.stdout: []` in [Labels and Annotations](#). For details about how to use this annotation, see [Table 5-14](#).
- **Image Access Credential**: Select the credential used for accessing the image repository. The default value is `default-secret`. You can use `default-secret` to access images in SWR. For details about `default-secret`, see [default-secret](#).
- (Optional) **GPU**: **All** is selected by default. The workload instance will be scheduled to the node of the specified GPU type.

### Headless Service Parameters

A headless Service is used to solve the problem of mutual access between pods in a StatefulSet. The headless Service provides a fixed access domain name for each pod. For details, see [Headless Services](#).

### (Optional) Service Settings

A Service provides external access for pods. With a static IP address, a Service forwards access traffic to pods and automatically balances load for these pods.

You can also create a Service after creating a workload. For details about Services of different types, see [Overview](#).

**(Optional) Advanced Settings**

- **Upgrade:** Specify the upgrade mode and parameters of the workload. **Rolling upgrade** and **Replace upgrade** are available. For details, see [Configuring Workload Upgrade Policies](#).
- **Pod Management Policies**

For some distributed systems, the StatefulSet sequence is unnecessary and/or should not occur. These systems require only uniqueness and identifiers.

  - **OrderedReady:** The StatefulSet will deploy, delete, or scale pods in order and one by one. (The StatefulSet continues only after the previous pod is ready or deleted.) This is the default policy.
  - **Parallel:** The StatefulSet will create pods in parallel to match the desired scale without waiting, and will delete all pods at once.
- **Scheduling:** Configure affinity and anti-affinity policies for flexible workload scheduling. Load affinity and node affinity are provided.
  - **Load Affinity:** Common load affinity policies are offered for quick load affinity deployment.
    - **Not configured:** No load affinity policy is configured.
    - **Multi-AZ deployment preferred:** Workload pods are **preferentially** scheduled to nodes in different AZs through pod anti-affinity.
    - **Forcible multi-AZ deployment:** Workload pods are forcibly scheduled to nodes in different AZs through pod anti-affinity (**podAntiAffinity**). If there are fewer AZs than pods, the extra pods will fail to run.
    - **Custom policies:** Affinity and anti-affinity policies can be customized. For details, see [Configuring Workload Affinity or Anti-affinity Scheduling \(podAffinity or podAntiAffinity\)](#).
  - **Node Affinity:** Common load affinity policies are offered for quick load affinity deployment.
    - **Not configured:** No node affinity policy is configured.
    - **Node Affinity:** Workload pods can be deployed on specified nodes through node affinity (**nodeAffinity**). If no node is specified, the pods will be randomly scheduled based on the default scheduling policy of the cluster.
    - **Specified node pool scheduling:** Workload pods can be deployed in a specified node pool through node affinity (**nodeAffinity**). If no node pool is specified, the pods will be randomly scheduled based on the default scheduling policy of the cluster.
    - **Custom policies:** Affinity and anti-affinity policies can be customized. For details, see [Configuring Node Affinity Scheduling \(nodeAffinity\)](#).
- **Toleration:** Using both taints and tolerations allows (not forcibly) the pod to be scheduled to a node with the matching taints, and controls the pod eviction policies after the node where the pod is located is tainted. For details, see [Configuring Tolerance Policies](#).

- **Labels and Annotations:** Add labels or annotations for pods using key-value pairs. After entering the key and value, click **Confirm**. For details about how to use and configure labels and annotations, see [Configuring Labels and Annotations](#).
- **DNS:** Configure a separate DNS policy for the workload. For details, see [DNS Configuration](#).
- **Network Configuration**
  - Pod ingress/egress bandwidth limitation: You can set ingress/egress bandwidth limitation for pods. For details, see [Configuring QoS for a Pod](#).
  - Whether to enable a specified container network configuration: available only for clusters that support this function. After you enable a specified container network configuration, the workload will be created using the container subnet and security group in the configuration. For details, see [Binding a Subnet and Security Group to a Namespace or Workload Using a Container Network Configuration](#).
  - Specify the container network configuration name: Only the custom container network configuration whose associated resource type is workload can be selected.

**Step 4** Click **Create Workload** in the lower right corner.

----End

## Using kubectl

In this example, a Nginx workload is used and the EVS volume is dynamically mounted to it using the **volumeClaimTemplates** field.

**Step 1** Use kubectl to access the cluster. For details, see [Connecting to a Cluster Using kubectl](#).

**Step 2** Create and edit the **nginx-statefulset.yaml** file.

**nginx-statefulset.yaml** is an example file name, and you can change it as required.

### vi nginx-statefulset.yaml

The following provides an example of the file contents. For more information on StatefulSet, see the [Kubernetes documentation](#).

```
apiVersion: apps/v1
kind: StatefulSet
metadata:
  name: nginx
spec:
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
        - name: container-1
          image: nginx:latest
```

```
imagePullPolicy: IfNotPresent
resources:
  requests:
    cpu: 250m
    memory: 512Mi
  limits:
    cpu: 250m
    memory: 512Mi
volumeMounts:
- name: test
  readOnly: false
  mountPath: /usr/share/nginx/html
  subPath: ""
imagePullSecrets:
- name: default-secret
dnsPolicy: ClusterFirst
volumes: []
serviceName: nginx-svc
replicas: 2
volumeClaimTemplates: # Dynamically mounts the EVS volume to the workload.
- apiVersion: v1
  kind: PersistentVolumeClaim
  metadata:
    name: test
    namespace: default
    annotations:
      everest.io/disk-volume-type: SAS # SAS EVS volume type.
    labels:
      failure-domain.beta.kubernetes.io/region: eu-west-101 # region where the EVS volume is created.
      failure-domain.beta.kubernetes.io/zone: # AZ where the EVS volume is created. It must be the
same as the AZ of the node.
  spec:
    accessModes:
    - ReadWriteOnce # The value must be ReadWriteOnce for the EVS volume.
    resources:
      requests:
        storage: 10Gi
      storageClassName: csi-disk # StorageClass name. The value is csi-disk for the EVS volume.
    updateStrategy:
      type: RollingUpdate
```

### vi nginx-headless.yaml

```
apiVersion: v1
kind: Service
metadata:
  name: nginx-svc
  namespace: default
  labels:
    app: nginx
spec:
  selector:
    app: nginx
    version: v1
  clusterIP: None
  ports:
  - name: nginx
    targetPort: 80
    nodePort: 0
    port: 80
    protocol: TCP
  type: ClusterIP
```

**Step 3** Create a workload and the corresponding headless service.

### kubectl create -f nginx-statefulset.yaml

If the following information is displayed, the StatefulSet has been successfully created.



```
statefulset.apps/nginx created
```

### **kubectl create -f nginx-headless.yaml**

If the following information is displayed, the headless service has been successfully created.

```
service/nginx-svc created
```

**Step 4** If the workload will be accessed through a ClusterIP or NodePort Service, configure the access mode. For details, see [Network](#).

----End

## 5.2.3 Creating a DaemonSet

### Scenario

CCE provides deployment and management capabilities for multiple types of containers and supports features of container workloads, including creation, configuration, monitoring, scaling, upgrade, uninstall, service discovery, and load balancing.

DaemonSet ensures that only one pod runs on all or some nodes. When a node is added to a cluster, a new pod is also added for the node. When a node is removed from a cluster, the pod is also reclaimed. If a DaemonSet is deleted, all pods created by it will be deleted.

The typical application scenarios of a DaemonSet are as follows:

- Run the cluster storage daemon, such as glusterd or Ceph, on each node.
- Run the log collection daemon, such as Fluentd or Logstash, on each node.
- Run the monitoring daemon, such as Prometheus Node Exporter, collectd, Datadog agent, New Relic agent, or Ganglia (gmond), on each node.

You can deploy a DaemonSet for each type of daemons on all nodes, or deploy multiple DaemonSets for the same type of daemons. In the second case, DaemonSets have different flags and different requirements on memory and CPU for different hardware types.

### Prerequisites

Before creating a DaemonSet, you must have an available cluster. For details on how to create a cluster, see [Buying a CCE Standard/Turbo Cluster](#).

### Using the CCE Console

**Step 1** Log in to the CCE console.

**Step 2** Click the cluster name to go to the cluster console, choose **Workloads** in the navigation pane, and click the **Create Workload** in the upper right corner.

**Step 3** Set basic information about the workload.

#### Basic Info

- **Workload Type:** Select **DaemonSet**. For details about workload types, see [Overview](#).

- **Workload Name:** Enter the name of the workload. Enter 1 to 63 characters starting with a lowercase letter and ending with a lowercase letter or digit. Only lowercase letters, digits, and hyphens (-) are allowed.
- **Namespace:** Select the namespace of the workload. The default value is **default**. You can also click **Create Namespace** to create one. For details, see [Creating a Namespace](#).
- **Time Zone Synchronization:** Specify whether to enable time zone synchronization. After time zone synchronization is enabled, the container and node use the same time zone. The time zone synchronization function depends on the local disk mounted to the container. Do not modify or delete the time zone. For details, see [Configuring Time Zone Synchronization](#).

### Container Settings

- Container Information

Multiple containers can be configured in a pod. You can click **Add Container** on the right to configure multiple containers for the pod.

- **Basic Info:** Configure basic information about the container.

Parameter	Description
Container Name	Name the container.
Pull Policy	Image update or pull policy. If you select <b>Always</b> , the image is pulled from the image repository each time. If you do not select <b>Always</b> , the existing image of the node is preferentially used. If the image does not exist, the image is pulled from the image repository.
Image Name	Click <b>Select Image</b> and select the image used by the container. To use a third-party image, see <a href="#">Using Third-Party Images</a> .
Image Tag	Select the image tag to be deployed.
CPU Quota	<ul style="list-style-type: none"> <li>▪ <b>Request:</b> minimum number of CPU cores required by a container. The default value is 0.25 cores.</li> <li>▪ <b>Limit:</b> maximum number of CPU cores that can be used by a container. This prevents containers from using excessive resources.</li> </ul> <p>If <b>Request</b> and <b>Limit</b> are not specified, the quota is not limited. For more information and suggestions about <b>Request</b> and <b>Limit</b>, see <a href="#">Configuring Container Specifications</a>.</p>

Parameter	Description
Memory Quota	<ul style="list-style-type: none"> <li>▪ <b>Request:</b> minimum amount of memory required by a container. The default value is 512 MiB.</li> <li>▪ <b>Limit:</b> maximum amount of memory available for a container. When memory usage exceeds the specified memory limit, the container will be terminated.</li> </ul> <p>If <b>Request</b> and <b>Limit</b> are not specified, the quota is not limited. For more information and suggestions about <b>Request</b> and <b>Limit</b>, see <a href="#">Configuring Container Specifications</a>.</p>
(Optional) GPU Quota	<p>Configurable only when the cluster contains GPU nodes and the <a href="#">CCE AI Suite (NVIDIA GPU)</a> add-on is installed.</p> <ul style="list-style-type: none"> <li>▪ <b>All:</b> No GPU will be used.</li> <li>▪ <b>Dedicated:</b> GPU resources are dedicated for the container.</li> <li>▪ <b>Shared:</b> percentage of GPU resources used by the container. For example, if this parameter is set to <b>10%</b>, the container uses 10% of GPU resources.</li> </ul> <p>For details about how to use GPUs in the cluster, see <a href="#">Default GPU Scheduling in Kubernetes</a>.</p>
(Optional) Privileged Container	<p>Programs in a privileged container have certain privileges.</p> <p>If <b>Privileged Container</b> is enabled, the container is assigned privileges. For example, privileged containers can manipulate network devices on the host machine and modify kernel parameters.</p>
(Optional) Init Container	<p>Whether to use the container as an init container. An init container does not support health check.</p> <p>An init container is a special container that runs before other app containers in a pod are started. Each pod can contain multiple containers. In addition, a pod can contain one or more init containers. Application containers in a pod are started and run only after the running of all init containers completes. For details, see <a href="#">Init Containers</a>.</p>

- (Optional) **Lifecycle:** Configure operations to be performed in a specific phase of the container lifecycle, such as Startup Command, Post-Start, and Pre-Stop. For details, see [Configuring Container Lifecycle Parameters](#).

- (Optional) **Health Check**: Set the liveness probe, ready probe, and startup probe as required. For details, see [Configuring Container Health Check](#).
- (Optional) **Environment Variables**: Configure variables for the container running environment using key-value pairs. These variables transfer external information to containers running in pods and can be flexibly modified after application deployment. For details, see [Configuring Environment Variables](#).
- (Optional) **Data Storage**: Mount local storage or cloud storage to the container. The application scenarios and mounting modes vary with the storage type. For details, see [Storage](#).
- (Optional) **Security Context**: Assign container permissions to protect the system and other containers from being affected. Enter the user ID to assign container permissions and prevent systems and other containers from being affected.
- (Optional) **Logging**: Report standard container output logs to AOM by default, without requiring manual settings. You can manually configure the log collection path. For details, see [Collecting Container Logs Using ICAgent](#).

To disable the standard output of the current workload, add the annotation `kubernetes.AOM.log.stdout: []` in [Labels and Annotations](#). For details about how to use this annotation, see [Table 5-14](#).

- **Image Access Credential**: Select the credential used for accessing the image repository. The default value is `default-secret`. You can use `default-secret` to access images in SWR. For details about `default-secret`, see [default-secret](#).
- (Optional) **GPU: All** is selected by default. The workload instance will be scheduled to the node of the specified GPU type.

### (Optional) Service Settings

A Service provides external access for pods. With a static IP address, a Service forwards access traffic to pods and automatically balances load for these pods.

You can also create a Service after creating a workload. For details about Services of different types, see [Overview](#).

### (Optional) Advanced Settings

- **Upgrade**: Specify the upgrade mode and parameters of the workload. **Rolling upgrade** and **Replace upgrade** are available. For details, see [Configuring Workload Upgrade Policies](#).
- **Scheduling**: Configure affinity and anti-affinity policies for flexible workload scheduling. Node affinity is provided.
  - **Node Affinity**: Common load affinity policies are offered for quick load affinity deployment.
    - **Not configured**: No node affinity policy is configured.
    - **Specified node scheduling**: Workload pods can be deployed on specified nodes through node affinity (`nodeAffinity`). If no node is specified, the pods will be randomly scheduled based on the default scheduling policy of the cluster.

- **Specified node pool scheduling:** Workload pods can be deployed in a specified node pool through node affinity (**nodeAffinity**). If no node pool is specified, the pods will be randomly scheduled based on the default scheduling policy of the cluster.
- **Custom policies:** Affinity and anti-affinity policies can be customized. For details, see [Configuring Node Affinity Scheduling \(nodeAffinity\)](#).
- **Toleration:** Using both taints and tolerations allows (not forcibly) the pod to be scheduled to a node with the matching taints, and controls the pod eviction policies after the node where the pod is located is tainted. For details, see [Configuring Tolerance Policies](#).
- **Labels and Annotations:** Add labels or annotations for pods using key-value pairs. After entering the key and value, click **Confirm**. For details about how to use and configure labels and annotations, see [Configuring Labels and Annotations](#).
- **DNS:** Configure a separate DNS policy for the workload. For details, see [DNS Configuration](#).

**Step 4** Click **Create Workload** in the lower right corner.

----End

## Using kubectl

The following procedure uses Nginx as an example to describe how to create a workload using kubectl.

**Step 1** Use kubectl to access the cluster. For details, see [Connecting to a Cluster Using kubectl](#).

**Step 2** Create and edit the **nginx-daemonset.yaml** file. **nginx-daemonset.yaml** is an example file name, and you can change it as required.

**vi nginx-daemonset.yaml**

The content of the description file is as follows: The following provides an example. For more information on DaemonSets, see [Kubernetes documentation](#).

```
apiVersion: apps/v1
kind: DaemonSet
metadata:
  name: nginx-daemonset
  labels:
    app: nginx-daemonset
spec:
  selector:
    matchLabels:
      app: nginx-daemonset
  template:
    metadata:
      labels:
        app: nginx-daemonset
    spec:
      nodeSelector:          # Node selection. A pod is created on a node only when the node meets
daemon=need.
      daemon: need
      containers:
        - name: nginx-daemonset
```

```
image: nginx:alpine
resources:
  limits:
    cpu: 250m
    memory: 512Mi
  requests:
    cpu: 250m
    memory: 512Mi
imagePullSecrets:
  - name: default-secret
```

The **replicas** parameter used in defining a Deployment or StatefulSet does not exist in the above configuration for a DaemonSet, because each node has only one replica. It is fixed.

The nodeSelector in the preceding pod template specifies that a pod is created only on the nodes that meet **daemon=need**. If you want to create a pod on each node, delete the label.

**Step 3** Create a DaemonSet.

**kubectl create -f nginx-daemonset.yaml**

If the following information is displayed, the DaemonSet is being created.

```
daemonset.apps/nginx-daemonset created
```

**Step 4** Obtain the DaemonSet status.

**kubectl get ds**

```
$ kubectl get ds
NAME          DESIRED  CURRENT  READY  UP-TO-DATE  AVAILABLE  NODE SELECTOR  AGE
nginx-daemonset  1        1        0      1           0          daemon=need    116s
```

**Step 5** If the workload will be accessed through a ClusterIP or NodePort Service, configure the access mode. For details, see [Network](#).

----End

## 5.2.4 Creating a Job

### Scenario

Jobs are short-lived and run for a certain time to completion. They can be executed immediately after being deployed. It is completed after it exits normally (exit 0).

A job is a resource object that is used to control batch tasks. It is different from a long-term servo workload (such as Deployment and StatefulSet).

A job is started and terminated at specific times, while a long-term servo workload runs unceasingly unless being terminated. The pods managed by a job automatically exit after successfully completing the job based on user configurations. The success flag varies according to the spec.completions policy.

- One-off jobs: A single pod runs once until successful termination.
- Jobs with a fixed success count: N pods run until successful termination.
- A queue job is considered completed based on the global success confirmed by the application.

## Prerequisites

Resources have been created. For details, see [Creating a Node](#). If clusters and nodes are available, you need not create them again.

## Using the CCE Console

- Step 1** Log in to the CCE console.
- Step 2** Click the cluster name to go to the cluster console, choose **Workloads** in the navigation pane, and click the **Create Workload** in the upper right corner.
- Step 3** Set basic information about the workload.

### Basic Info

- **Workload Type:** Select **Job**. For details about workload types, see [Overview](#).
- **Workload Name:** Enter the name of the workload. Enter 1 to 63 characters starting with a lowercase letter and ending with a lowercase letter or digit. Only lowercase letters, digits, and hyphens (-) are allowed.
- **Namespace:** Select the namespace of the workload. The default value is **default**. You can also click **Create Namespace** to create one. For details, see [Creating a Namespace](#).
- **Pods:** Enter the number of pods of the workload.

### Container Settings

- Container Information
  - Multiple containers can be configured in a pod. You can click **Add Container** on the right to configure multiple containers for the pod.
  - **Basic Info:** Configure basic information about the container.

Parameter	Description
Container Name	Name the container.
Pull Policy	Image update or pull policy. If you select <b>Always</b> , the image is pulled from the image repository each time. If you do not select <b>Always</b> , the existing image of the node is preferentially used. If the image does not exist, the image is pulled from the image repository.
Image Name	Click <b>Select Image</b> and select the image used by the container. To use a third-party image, see <a href="#">Using Third-Party Images</a> .
Image Tag	Select the image tag to be deployed.

Parameter	Description
CPU Quota	<ul style="list-style-type: none"> <li>▪ <b>Request:</b> minimum number of CPU cores required by a container. The default value is 0.25 cores.</li> <li>▪ <b>Limit:</b> maximum number of CPU cores that can be used by a container. This prevents containers from using excessive resources.</li> </ul> <p>If <b>Request</b> and <b>Limit</b> are not specified, the quota is not limited. For more information and suggestions about <b>Request</b> and <b>Limit</b>, see <a href="#">Configuring Container Specifications</a>.</p>
Memory Quota	<ul style="list-style-type: none"> <li>▪ <b>Request:</b> minimum amount of memory required by a container. The default value is 512 MiB.</li> <li>▪ <b>Limit:</b> maximum amount of memory available for a container. When memory usage exceeds the specified memory limit, the container will be terminated.</li> </ul> <p>If <b>Request</b> and <b>Limit</b> are not specified, the quota is not limited. For more information and suggestions about <b>Request</b> and <b>Limit</b>, see <a href="#">Configuring Container Specifications</a>.</p>
(Optional) GPU Quota	<p>Configurable only when the cluster contains GPU nodes and the <a href="#">CCE AI Suite (NVIDIA GPU)</a> add-on is installed.</p> <ul style="list-style-type: none"> <li>▪ <b>All:</b> No GPU will be used.</li> <li>▪ <b>Dedicated:</b> GPU resources are dedicated for the container.</li> <li>▪ <b>Shared:</b> percentage of GPU resources used by the container. For example, if this parameter is set to <b>10%</b>, the container uses 10% of GPU resources.</li> </ul> <p>For details about how to use GPUs in the cluster, see <a href="#">Default GPU Scheduling in Kubernetes</a>.</p>
(Optional) Privileged Container	<p>Programs in a privileged container have certain privileges.</p> <p>If <b>Privileged Container</b> is enabled, the container is assigned privileges. For example, privileged containers can manipulate network devices on the host machine and modify kernel parameters.</p>



Parameter	Description
(Optional) Init Container	Whether to use the container as an init container. An init container does not support health check. An init container is a special container that runs before other app containers in a pod are started. Each pod can contain multiple containers. In addition, a pod can contain one or more init containers. Application containers in a pod are started and run only after the running of all init containers completes. For details, see <a href="#">Init Containers</a> .

- (Optional) **Lifecycle**: Configure operations to be performed in a specific phase of the container lifecycle, such as Startup Command, Post-Start, and Pre-Stop. For details, see [Configuring Container Lifecycle Parameters](#).
- (Optional) **Environment Variables**: Configure variables for the container running environment using key-value pairs. These variables transfer external information to containers running in pods and can be flexibly modified after application deployment. For details, see [Configuring Environment Variables](#).
- (Optional) **Data Storage**: Mount local storage or cloud storage to the container. The application scenarios and mounting modes vary with the storage type. For details, see [Storage](#).

 NOTE

If the workload contains more than one pod, EVS volumes cannot be mounted.

- (Optional) **Logging**: Report standard container output logs to AOM by default, without requiring manual settings. You can manually configure the log collection path. For details, see [Collecting Container Logs Using ICAgent](#).

To disable the standard output of the current workload, add the annotation `kubernetes.AOM.log.stdout: []` in [Labels and Annotations](#). For details about how to use this annotation, see [Table 5-14](#).

- **Image Access Credential**: Select the credential used for accessing the image repository. The default value is `default-secret`. You can use `default-secret` to access images in SWR. For details about `default-secret`, see [default-secret](#).
- (Optional) **GPU: All** is selected by default. The workload instance will be scheduled to the node of the specified GPU type.

**(Optional) Advanced Settings**

- **Labels and Annotations**: Add labels or annotations for pods using key-value pairs. After entering the key and value, click **Confirm**. For details about how to use and configure labels and annotations, see [Configuring Labels and Annotations](#).
- **Job Settings**
  - **Parallel Pods**: Maximum number of pods that can run in parallel during job execution. The value cannot be greater than the total number of pods in the job.

- **Timeout (s):** Once a job reaches this time, the job status becomes failed and all pods in this job will be deleted. If you leave this parameter blank, the job will never time out.
- Completion Mode
  - **Non-indexed:** A job is considered complete when all the pods are successfully executed. Each pod completion is homologous to each other.
  - **Indexed:** Each pod gets an associated completion index from 0 to the number of pods minus 1. The job is considered complete when every pod allocated with an index is successfully executed. For an indexed job, pods are named in the format of \$(job-name)-\$(index).
- **Suspend Job:** By default, a job is executed immediately after being created. The job's execution will be suspended if you enable this option, and resumed after you disable it.

**Step 4** Click **Create Workload** in the lower right corner.

----End

## Using kubectl

A job has the following configuration parameters:

- **.spec.completions:** indicates the number of pods that need to run successfully to end a job. The default value is **1**.
- **.spec.parallelism:** indicates the number of pods that run concurrently. The default value is **1**.
- **.spec.backoffLimit:** indicates the maximum number of retries performed if a pod fails. When the limit is reached, the pod will not try again.
- **.spec.activeDeadlineSeconds:** indicates the running time of pods. Once the time is reached, all pods of the job are terminated. The priority of **.spec.activeDeadlineSeconds** is higher than that of **.spec.backoffLimit**. That is, if a job reaches the **.spec.activeDeadlineSeconds**, the **.spec.backoffLimit** is ignored.

Based on the **.spec.completions** and **.spec.parallelism** settings, jobs are classified into the following types.

**Table 5-3** Job types

Job Type	Description	.spec.completions	.spec.parallelism
One-off jobs	A job creates one pod until it successfully completes.	1	1
Jobs with a fixed completion count	A job creates one pod in sequence and is complete when the number of successful pods reaches the value of <b>.spec.completions</b> .	>1	1

Job Type	Description	.spec.comple tions	.spec.parall elism
Parallel jobs with a fixed completion count	A job creates multiple pods in sequence and is complete when the number of successful pods reaches the value of <b>.spec.completions</b> .	>1	>1
Parallel jobs with a work queue	A job creates one or more pods. Each pod takes one task from the message queue, processes it, and repeats until the end of the queue is reached. Then the pod deletes the task and exists. For details, see <a href="#">Fine Parallel Processing Using a Work Queue</a> .	Leave this parameter blank.	>1 or =1

The following is an example job, which calculates  $\pi$  till the 2000<sup>th</sup> digit and prints the output.

```
apiVersion: batch/v1
kind: Job
metadata:
  name: myjob
spec:
  completions: 50      # A total of 50 pods need to run to finish a job. In this example,  $\pi$  is printed for 50
                        times.
  parallelism: 5      # A total of 5 pods run in parallel.
  backoffLimit: 5     # A maximum of 5 retries is allowed.
  template:
    spec:
      containers:
        - name: pi
          image: perl
          command: ["perl", "-Mbignum=bpi", "-wle", "print bpi(2000)"]
          restartPolicy: Never # For a job, set this parameter to Never or OnFailure. For other controllers (such
                               as Deployments), set this parameter to Always.
      imagePullSecrets:
        - name: default-secret
```

**Run the job.**

**Step 1** Start the job.

```
[root@k8s-master k8s]# kubectl apply -f myjob.yaml
job.batch/myjob created
```

**Step 2** View the job details.

**kubectl get job**

```
[root@k8s-master k8s]# kubectl get job
NAME      COMPLETIONS  DURATION  AGE
myjob    50/50         23s      3m45s
```

If the value of **COMPLETIONS** is **50/50**, the job is successfully executed.

**Step 3** View the pod status.

### kubectctl get pod

```
[root@k8s-master k8s]# kubectctl get pod
NAME      READY  STATUS   RESTARTS  AGE
myjob-29qlw  0/1    Completed  0          4m5s
...
```

If the status is **Completed**, the job is complete.

#### Step 4 View the pod logs.

### kubectctl logs <pod\_name>

```
# kubectctl logs myjob-29qlw
3.141592653589793238462643383279502884197169399375105820974944592307816406286208998628034
8253421170679821480865132823066470938446095505822317253594081284811174502841027019385211
0555964462294895493038196442881097566593344612847564823378678316527120190914564856692346
0348610454326648213393607260249141273724587006606315588174881520920962829254091715364367
8925903600113305305488204665213841469519415116094330572703657595919530921861173819326117
9310511854807446237996274956735188575272489122793818301194912983367336244065664308602139
4946395224737190702179860943702770539217176293176752384674818467669405132000568127145263
5608277857713427577896091736371787214684409012249534301465495853710507922796892589235420
1995611212902196086403441815981362977477130996051870721134999999837297804995105973173281
6096318595024459455346908302642522308253344685035261931188171010003137838752886587533208
3814206171776691473035982534904287554687311595628638823537875937519577818577805321712268
0661300192787661119590921642019893809525720106548586327886593615338182796823030195203530
1852968995773622599413891249721775283479131515574857242454150695950829533116861727855889
0750983817546374649393192550604009277016711390098488240128583616035637076601047101819429
5559619894676783744944825537977472684710404753464620804668425906949129331367702898915210
4752162056966024058038150193511253382430035587640247496473263914199272604269922796782354
7816360093417216412199245863150302861829745557067498385054945885869269956909272107975093
0295532116534498720275596023648066549911988183479775356636980742654252786255181841757467
2890977772793800081647060016145249192173217214772350141441973568548161361157352552133475
7418494684385233239073941433345477624168625189835694855620992192221842725502542568876717
9049460165346680498862723279178608578438382796797668145410095388378636095068006422512520
5117392984896084128488626945604241965285022210661186306744278622039194945047123713786960
9563643719172874677646575739624138908658326459958133904780275901
```

----End

## Related Operations

After a one-off job is created, you can perform operations listed in [Table 5-4](#).

**Table 5-4** Other operations

Operation	Description
Editing a YAML file	Click <b>More</b> > <b>Edit YAML</b> next to the job name to edit the YAML file of the current job.
Deleting a job	<ol style="list-style-type: none"> <li>Select the target job and choose <b>More</b> &gt; <b>Delete</b> in the <b>Operation</b> column.</li> <li>Click <b>Yes</b>. Deleted jobs cannot be restored. Exercise caution when deleting a job.</li> </ol>

## 5.2.5 Creating a Cron Job

### Scenario

A cron job runs on a repeating schedule. You can perform time synchronization for all active nodes at a fixed time point.

A cron job runs periodically at the specified time. It is similar with Linux crontab. A cron job has the following characteristics:

- Runs only once at the specified time.
- Runs periodically at the specified time.

The typical usage of a cron job is as follows:

- Schedules jobs at the specified time.
- Creates jobs to run periodically, for example, database backup and email sending.

### Prerequisites

Resources have been created. For details, see [Creating a Node](#).

### Using the CCE Console

**Step 1** Log in to the CCE console.

**Step 2** Click the cluster name to go to the cluster console, choose **Workloads** in the navigation pane, and click the **Create Workload** in the upper right corner.

**Step 3** Set basic information about the workload.

#### Basic Info

- **Workload Type:** Select **Cron Job**. For details about workload types, see [Overview](#).
- **Workload Name:** Enter the name of the workload. Enter 1 to 63 characters starting with a lowercase letter and ending with a lowercase letter or digit. Only lowercase letters, digits, and hyphens (-) are allowed.
- **Namespace:** Select the namespace of the workload. The default value is **default**. You can also click **Create Namespace** to create one. For details, see [Creating a Namespace](#).

#### Container Settings

- Container Information

Multiple containers can be configured in a pod. You can click **Add Container** on the right to configure multiple containers for the pod.

- **Basic Info:** Configure basic information about the container.

Parameter	Description
Container Name	Name the container.

Parameter	Description
Pull Policy	Image update or pull policy. If you select <b>Always</b> , the image is pulled from the image repository each time. If you do not select <b>Always</b> , the existing image of the node is preferentially used. If the image does not exist, the image is pulled from the image repository.
Image Name	Click <b>Select Image</b> and select the image used by the container. To use a third-party image, see <a href="#">Using Third-Party Images</a> .
Image Tag	Select the image tag to be deployed.
CPU Quota	<ul style="list-style-type: none"> <li>▪ <b>Request:</b> minimum number of CPU cores required by a container. The default value is 0.25 cores.</li> <li>▪ <b>Limit:</b> maximum number of CPU cores that can be used by a container. This prevents containers from using excessive resources.</li> </ul> <p>If <b>Request</b> and <b>Limit</b> are not specified, the quota is not limited. For more information and suggestions about <b>Request</b> and <b>Limit</b>, see <a href="#">Configuring Container Specifications</a>.</p>
Memory Quota	<ul style="list-style-type: none"> <li>▪ <b>Request:</b> minimum amount of memory required by a container. The default value is 512 MiB.</li> <li>▪ <b>Limit:</b> maximum amount of memory available for a container. When memory usage exceeds the specified memory limit, the container will be terminated.</li> </ul> <p>If <b>Request</b> and <b>Limit</b> are not specified, the quota is not limited. For more information and suggestions about <b>Request</b> and <b>Limit</b>, see <a href="#">Configuring Container Specifications</a>.</p>
(Optional) GPU Quota	Configurable only when the cluster contains GPU nodes and the <a href="#">CCE AI Suite (NVIDIA GPU)</a> add-on is installed. <ul style="list-style-type: none"> <li>▪ <b>All:</b> No GPU will be used.</li> <li>▪ <b>Dedicated:</b> GPU resources are dedicated for the container.</li> <li>▪ <b>Shared:</b> percentage of GPU resources used by the container. For example, if this parameter is set to <b>10%</b>, the container uses 10% of GPU resources.</li> </ul> <p>For details about how to use GPUs in the cluster, see <a href="#">Default GPU Scheduling in Kubernetes</a>.</p>

Parameter	Description
(Optional) Privileged Container	<p>Programs in a privileged container have certain privileges.</p> <p>If <b>Privileged Container</b> is enabled, the container is assigned privileges. For example, privileged containers can manipulate network devices on the host machine and modify kernel parameters.</p>
(Optional) Init Container	<p>Whether to use the container as an init container. An init container does not support health check.</p> <p>An init container is a special container that runs before other app containers in a pod are started. Each pod can contain multiple containers. In addition, a pod can contain one or more init containers.</p> <p>Application containers in a pod are started and run only after the running of all init containers completes. For details, see <a href="#">Init Containers</a>.</p>

- (Optional) **Lifecycle**: Configure operations to be performed in a specific phase of the container lifecycle, such as Startup Command, Post-Start, and Pre-Stop. For details, see [Configuring Container Lifecycle Parameters](#).
- (Optional) **Environment Variables**: Configure variables for the container running environment using key-value pairs. These variables transfer external information to containers running in pods and can be flexibly modified after application deployment. For details, see [Configuring Environment Variables](#).
- **Image Access Credential**: Select the credential used for accessing the image repository. The default value is **default-secret**. You can use default-secret to access images in SWR. For details about **default-secret**, see [default-secret](#).
- (Optional) **GPU: All** is selected by default. The workload instance will be scheduled to the node of the specified GPU type.

#### Schedule

- **Concurrency Policy**: The following three modes are supported:
  - **Forbid**: A new job cannot be created before the previous job is completed.
  - **Allow**: The cron job allows concurrently running jobs, which preempt cluster resources.
  - **Replace**: A new job replaces the previous job when it is time to create a job but the previous job is not completed.
- **Policy Settings**: specifies when a new cron job is executed. Policy settings in YAML are implemented using cron expressions.
  - A cron job is executed at a fixed interval. The unit can be minute, hour, day, or month. For example, if a cron job is executed every 30 minutes and the corresponding cron expression is **\*/30 \* \* \* \***, the execution time starts from 0 in the unit range, for example, **00:00:00, 00:30:00, 01:00:00**, and ....

- The cron job is executed at a fixed time (by month). For example, if a cron job is executed at 00:00 on the first day of each month, the cron expression is `0 0 1 */1 *`, and the execution time is `****-01-01 00:00:00`, `****-02-01 00:00:00`, and ....
- The cron job is executed by week. For example, if a cron job is executed at 00:00 every Monday, the cron expression is `0 0 * * 1`, and the execution time is `****-**-01 00:00:00 on Monday`, `****-**-08 00:00:00 on Monday`, and ....
- **Custom Cron Expression:** For details about how to use cron expressions, see [CronJob](#).

#### NOTE

- If a cron job is executed at a fixed time (by month) and the number of days in a month does not exist, the cron job will not be executed in this month. For example, the execution will skip February if the date is set to 30.
- Due to the definition of cron, the fixed period is not a strict period. The time unit range is divided from 0 by period. For example, if the unit is minute, the value ranges from 0 to 59. If the value cannot be exactly divided, the last period is reset. Therefore, an accurate period can be represented only when the period can be evenly divided.  

Take a cron job that runs hourly as an example. If an interval that can divide 24 hours exactly, such as `/2`, `/3`, `/4`, `/6`, `/8`, and `/12`, the period can be accurately represented. However, if a different period is used, the last period will reset at the beginning of each new day. For example, if the cron expression is `**/12 * * *`, the task will run at `00:00:00` and `12:00:00` every day. Similarly, if the cron expression is `**/13 * * *`, the task will run at `00:00:00` and `13:00:00` every day. Note that the last period will reset at 00:00:00 on the following day, even if it has not reached 13 hours.
- **Time Zone:** You can specify a time zone. If this parameter is not specified, the time zone of the master node will be used by default.
- **Job Records:** You can set the number of jobs that are successfully executed or fail to be executed. Setting a limit to `0` corresponds to keeping none of the jobs after they finish.

#### (Optional) Advanced Settings

- **Labels and Annotations:** Add labels or annotations for pods using key-value pairs. After entering the key and value, click **Confirm**. For details about how to use and configure labels and annotations, see [Configuring Labels and Annotations](#).

**Step 4** Click **Create Workload** in the lower right corner.

----End

## Using kubectl

A cron job has the following configuration parameters:

- **.spec.schedule:** takes a [Cron](#) format string, for example, `0 * * * *` or `@hourly`, as schedule time of jobs to be created and executed.
- **.spec.jobTemplate:** specifies jobs to be run, and has the same schema as when you are [Creating a Job Using kubectl](#).
- **.spec.startingDeadlineSeconds:** specifies the deadline for starting a job.



- **.spec.concurrencyPolicy**: specifies how to treat concurrent executions of a job created by the CronJob. The following options are supported:
  - **Allow** (default value): allows concurrently running jobs.
  - **Forbid**: forbids concurrent runs, skipping next run if previous has not finished yet.
  - **Replace**: cancels the currently running job and replaces it with a new one.

The following is an example cron job, which is saved in the **cronjob.yaml** file.

 **NOTE**

In clusters of v1.21 or later, CronJob apiVersion is **batch/v1**.

In clusters earlier than v1.21, CronJob apiVersion is **batch/v1beta1**.

```
apiVersion: batch/v1
kind: CronJob
metadata:
  name: hello
spec:
  schedule: "*/1 * * * *"
  jobTemplate:
    spec:
      template:
        spec:
          containers:
            - name: hello
              image: busybox
              command:
                - /bin/sh
                - -c
                - date; echo Hello from the Kubernetes cluster
          restartPolicy: OnFailure
          imagePullSecrets:
            - name: default-secret
```

**Run the job.**

**Step 1** Create a CronJob.

**kubectl create -f cronjob.yaml**

Information similar to the following is displayed:

```
cronjob.batch/hello created
```

**Step 2** Query the running status of the cron job:

**kubectl get cronjob**

NAME	SCHEDULE	SUSPEND	ACTIVE	LAST SCHEDULE	AGE
hello	*/1 * * * *	False	0	<none>	9s

**kubectl get jobs**

NAME	COMPLETIONS	DURATION	AGE
hello-1597387980	1/1	27s	45s

**kubectl get pod**

NAME	READY	STATUS	RESTARTS	AGE
hello-1597387980-tjv8f	0/1	Completed	0	114s
hello-1597388040-lckg9	0/1	Completed	0	39s

**kubectl logs hello-1597387980-tjv8f**

```
Fri Aug 14 06:56:31 UTC 2020
Hello from the Kubernetes cluster
```

### kubectrl delete cronjob hello

```
cronjob.batch "hello" deleted
```

#### NOTICE

When a CronJob is deleted, the related jobs and pods are deleted accordingly.

----End

## Related Operations

After a CronJob is created, you can perform operations listed in [Table 5-5](#).

**Table 5-5** Other operations

Operation	Description
Editing a YAML file	Click <b>More</b> > <b>Edit YAML</b> next to the cron job name to edit the YAML file of the current job.
Stopping a CronJob	<ol style="list-style-type: none"> <li>1. Select the job to be stopped and click <b>Stop</b> in the <b>Operation</b> column.</li> <li>2. Click <b>Yes</b>.</li> </ol>
Deleting a CronJob	<ol style="list-style-type: none"> <li>1. Select the CronJob to be deleted and click <b>More</b> &gt; <b>Delete</b> in the <b>Operation</b> column.</li> <li>2. Click <b>Yes</b>. Deleted jobs cannot be restored. Exercise caution when deleting a job.</li> </ol>

## 5.3 Configuring a Workload

### 5.3.1 Configuring Time Zone Synchronization

When creating a workload, you can configure containers to use the same time zone as the node. You can enable time zone synchronization when creating a workload.

The time zone synchronization function depends on the local disk (hostPath) mounted to the container. After time zone synchronization is enabled, **/etc/localtime** of the node is mounted to **/etc/localtime** of the container in HostPath mode, in this way, the node and container use the same time zone configuration file.

```
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
```

### 5.3.2 Configuring an Image Pull Policy

When a workload is created, the container image is pulled from the image repository to the node. The image is also pulled when the workload is restarted or upgraded.

By default, **imagePullPolicy** is set to **IfNotPresent**, indicating that if the image exists on the node, the existing image is used. If the image does not exist on the node, the image is pulled from the image repository.

The image pull policy can also be set to **Always**, indicating that the image is pulled from the image repository and overwrites the image on the node regardless of whether the image exists on the node.

```
apiVersion: v1
kind: Pod
metadata:
  name: nginx
spec:
  containers:
    - image: nginx:alpine
      name: container-0
  resources:
    limits:
      cpu: 100m
      memory: 200Mi
    requests:
      cpu: 100m
      memory: 200Mi
    imagePullPolicy: Always
  imagePullSecrets:
    - name: default-secret
```

An image pull policy can also be configured on the CCE console. When creating a workload, configure **Pull Policy**. If **Always** is selected, images are always pulled. If **Always** is not selected, images are pulled as needed.

**NOTICE**

Use a new tag each time you create an image. If you do not update the tag but only update the image, when **Pull Policy** is set to **IfNotPresent**, CCE considers that an image with the tag already exists on the current node and will not pull the image again.

### 5.3.3 Using Third-Party Images

#### Scenario

CCE allows you to create workloads using images pulled from third-party image repositories.

Generally, a third-party image repository can be accessed only after authentication (using your account and password). CCE uses the secret-based authentication to pull images. Therefore, create a secret for an image repository before pulling images from the repository.

#### Prerequisites

The node where the workload is running is accessible from public networks.

#### Using the Console

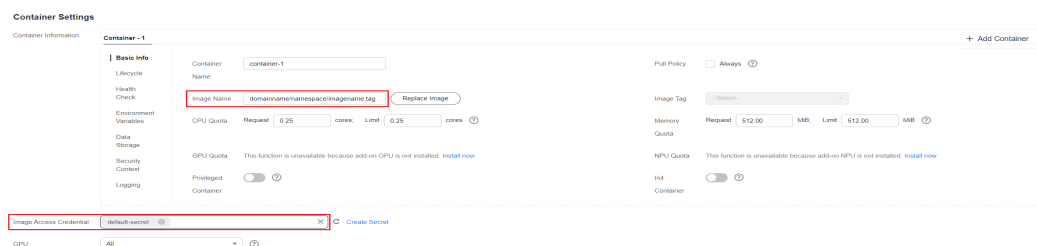
**Step 1** Create a secret for accessing a third-party image repository.

Click the cluster name to access the cluster console. In the navigation pane, choose **ConfigMaps and Secrets**. On the **Secrets** tab page, click **Create Secret** in the upper right corner. Set **Secret Type** to **kubernetes.io/dockerconfigjson**. For details, see [Creating a Secret](#).

Enter the username and password used to access the third-party image repository.

**Step 2** When creating a workload, enter a private image path in the format of *domainname/namespace/imagename:tag* in **Image Name** and select the key created in **Step 1**.

**Figure 5-4** Private image path



**Step 3** Set other parameters and click **Create Workload**.

----End

## Using kubectl

**Step 1** Use kubectl to access the cluster. For details, see [Connecting to a Cluster Using kubectl](#).

**Step 2** Use kubectl to create a secret of the kubernetes.io/dockerconfigjson.

```
kubectl create secret docker-registry myregistrykey -n default --docker-server=DOCKER_REGISTRY_SERVER  
--docker-username=DOCKER_USER --docker-password=DOCKER_PASSWORD --docker-  
email=DOCKER_EMAIL
```

In the preceding command, *myregistrykey* indicates the key name, *default* indicates the namespace where the key is located, and other parameters are as follows:

- **DOCKER\_REGISTRY\_SERVER**: address of a third-party image repository, for example, **www.3rdregistry.com** or **10.10.10.10:443**
- **DOCKER\_USER**: account used for logging in to a third-party image repository
- **DOCKER\_PASSWORD**: password used for logging in to a third-party image repository
- **DOCKER\_EMAIL**: email of a third-party image repository

**Step 3** Use a third-party image to create a workload.

A kubernetes.io/dockerconfigjson secret is used for authentication when you obtain a private image. The following is an example of using the myregistrykey for authentication.

```
apiVersion: v1  
kind: Pod  
metadata:  
  name: foo  
  namespace: default  
spec:  
  containers:  
  - name: foo  
    image: www.3rdregistry.com/janedoe/awesomeapp:v1  
  imagePullSecrets:  
  - name: myregistrykey          # Use the created secret.
```

----End

## 5.3.4 Configuring Container Specifications

### Scenario

CCE allows you to set resource requirements and limits, such as CPU and RAM, for added containers during workload creation. Kubernetes also allows using YAML to set requirements of other resource types.

### Request and Limit

For **CPU** and **Memory**, the meanings of **Request** and **Limit** are as follows:

- **Request**: The system schedules a pod to the node that meets the requirements for workload deployment based on the request value.
- **Limit**: The system limits the resources used by the workload based on the limit value.

If a node has sufficient resources, the pod on this node can use more resources than requested, but no more than limited.

For example, if you set the memory request of a container to 1 GiB and the limit value to 2 GiB, a pod is scheduled to a node with 8 GiB CPUs with no other pod running. In this case, the pod can use more than 1 GiB memory when the load is heavy, but the memory usage cannot exceed 2 GiB. If a process in a container attempts to use more than 2 GiB resources, the system kernel attempts to terminate the process. As a result, an out of memory (OOM) error occurs.

 **NOTE**

When creating a workload, you are advised to set the upper and lower limits of CPU and memory resources. If the upper and lower resource limits are not set for a workload, a resource leak of this workload will make resources unavailable for other workloads deployed on the same node. In addition, workloads that do not have upper and lower resource limits cannot be accurately monitored.

## Configuration

In real-world scenarios, the recommended ratio of **Request** to **Limit** is about 1:1.5. For some sensitive services, the recommended ratio is 1:1. If the **Request** is too small and the **Limit** is too large, node resources are oversubscribed. During service peaks, the memory or CPU of a node may be used up. As a result, the node is unavailable.

- CPU quota: The unit of CPU resources is core, which can be expressed by quantity or an integer suffixed with the unit (m). For example, 0.1 core in the quantity expression is equivalent to 100m in the expression. However, Kubernetes does not allow CPU resources whose precision is less than 1m.

**Table 5-6** CPU quotas

Parameter	Description
CPU request	Minimum number of CPU cores required by a container. Resources are scheduled for the container based on this value. The container can be scheduled to this node only when the total available CPU on the node is greater than or equal to the number of containerized CPU applications.
CPU limit	Maximum number of CPU cores available for a container.

### Recommended configuration

Actual available CPU of a node  $\geq$  Sum of CPU limits of all containers on the current node  $\geq$  Sum of CPU requests of all containers on the current node. You can view the actual available CPUs of a node on the CCE console (**Resource Management > Nodes > Allocatable**).

- Memory quota: The default unit of memory resources is byte. You can also use an integer with the unit suffix, for example, 100 Mi. Note that the unit is case-sensitive.

**Table 5-7** Description of memory quotas

Parameter	Description
Memory request	Minimum amount of memory required by a container. Resources are scheduled for the container based on this value. The container can be scheduled to this node only when the total available memory on the node is greater than or equal to the number of containerized memory applications.
Memory Limit	Maximum amount of memory available for a container. When the memory usage exceeds the configured memory limit, the instance may be restarted, which affects the normal use of the workload.

### Recommended configuration

Actual available memory of a node  $\geq$  Sum of memory limits of all containers on the current node  $\geq$  Sum of memory requests of all containers on the current node. You can view the actual available memory of a node on the CCE console (**Resource Management > Nodes > Allocatable**).

#### NOTE

The allocatable resources are calculated based on the resource request value (**Request**), which indicates the upper limit of resources that can be requested by pods on this node, but does not indicate the actual available resources of the node (for details, see [Example of CPU and Memory Quota Usage](#)). The calculation formula is as follows:

- Allocatable CPU = Total CPU – Requested CPU of all pods – Reserved CPU for other resources
- Allocatable memory = Total memory – Requested memory of all pods – Reserved memory for other resources

## Example of CPU and Memory Quota Usage

Assume that a cluster contains a node with 4 CPU cores and 8 GiB memory. Two pods (pod 1 and pod 2) have been deployed on the cluster. Pod 1 oversubscribes resources (that is **Limit > Request**). The specifications of the two pods are as follows.

Pod	CPU Request	CPU Limit	Memory Request	Memory Limit
Pod 1	1 core	2 cores	1 GiB	4 GiB
Pod 2	2 cores	2 cores	2 GiB	2 GiB

The CPU and memory usage of the node is as follows:

- Allocatable CPUs = 4 cores – (1 core requested by pod 1 + 2 cores requested by pod 2) = 1 core
- Allocatable memory = 8 GiB – (1 GiB requested by pod 1 + 2 GiB requested by pod 2) = 5 GiB

In this case, the remaining 1 core 5 GiB can be used by the next new pod.

If pod 1 is under heavy load during peak hours, it will use more CPUs and memory within the limit. Therefore, the actual allocatable resources are fewer than 1 core 5 GiB.

## Quotas of Other Resources

Typically, nodes support local ephemeral storage, which is provided by locally mounted writable devices or RAM. EV does not ensure long-term data availability. Pods can use local EVs to buffer data and store logs, or mount emptyDir volumes to containers. For details, see [Local ephemeral storage](#).

Kubernetes allows you to specify the requested value and limit value of ephemeral storage in container configurations to manage the local ephemeral storage. The following attributes can be configured for each container in a pod:

- `spec.containers[].resources.limits.ephemeral-storage`
- `spec.containers[].resources.requests.ephemeral-storage`

In the following example, a pod contains two containers. The requested value of each container for local ephemeral storage is 2 GiB, and the limit value is 4 GiB. Therefore, the requested value of the pod for local ephemeral storage is 4 GiB, the limit value is 8 GiB, and the emptyDir volume uses 500 MiB of the local ephemeral storage.

```
apiVersion: v1
kind: Pod
metadata:
  name: frontend
spec:
  containers:
    - name: container-1
      image: <example_app_image>
      resources:
        requests:
          ephemeral-storage: "2Gi"
        limits:
          ephemeral-storage: "4Gi"
      volumeMounts:
        - name: ephemeral
          mountPath: "/tmp"
    - name: container-2
      image: <example_log_aggregator_image>
      resources:
        requests:
          ephemeral-storage: "2Gi"
        limits:
          ephemeral-storage: "4Gi"
      volumeMounts:
        - name: ephemeral
          mountPath: "/tmp"
  volumes:
    - name: ephemeral
      emptyDir:
        sizeLimit: 500Mi
```



## 5.3.5 Configuring Container Lifecycle Parameters

### Scenario

CCE provides callback functions for the lifecycle management of containerized applications. For example, if you want a container to perform a certain operation before stopping, you can register a hook function.

CCE provides the following lifecycle callback functions:

- **Startup Command:** executed to start a container. For details, see [Startup Commands](#).
- **Post-Start:** executed immediately after a container is started. For details, see [Post-Start Processing](#).
- **Pre-Stop:** executed before a container is stopped. The pre-stop processing function helps you ensure that the services running on the pods can be completed in advance in the case of pod upgrade or deletion. For details, see [Pre-Stop Processing](#).

### Startup Commands

By default, the default command is executed during image start. To run a specific command or rewrite the default image setting, you must perform specific operations.

A Docker image has metadata that stores image information. If lifecycle commands and arguments are not set, CCE runs the default commands and arguments, that is, Docker instructions **ENTRYPOINT** and **CMD**, provided during image creation.

If the commands and arguments used to run a container are set during application creation, the default commands **ENTRYPOINT** and **CMD** are overwritten during image build. The rules are as follows:

**Table 5-8** Commands and arguments used to run a container

Image ENTRYPOINT	Image CMD	Command to Run a Container	Parameters to Run a Container	Command Executed
[touch]	[/root/test]	Not set	Not set	[touch /root/test]
[touch]	[/root/test]	[mkdir]	Not set	[mkdir]
[touch]	[/root/test]	Not set	[/opt/test]	[touch /opt/test]
[touch]	[/root/test]	[mkdir]	[/opt/test]	[mkdir /opt/test]

**Step 1** Log in to the CCE console. When creating a workload, configure container information and select **Lifecycle**.

**Step 2** Enter a command and arguments on the **Startup Command** tab page.

**Table 5-9** Container startup command

Configuration Item	Procedure
Command	<p>Enter an executable command, for example, <b>/run/server</b>.</p> <p>If there are multiple executable commands, write them in different lines.</p> <p><b>NOTE</b> In the case of multiple commands, you are advised to run <b>/bin/sh</b> or other <b>shell</b> commands. Other commands are used as parameters.</p>
Args	<p>Enter the argument that controls the container running command, for example, <b>--port=8080</b>.</p> <p>If there are multiple arguments, separate them in different lines.</p>

----End

## Post-Start Processing

**Step 1** Log in to the CCE console. When creating a workload, configure container information and select **Lifecycle**.

**Step 2** Set the post-start processing parameters on the **Post-Start** tab page.

**Table 5-10** Post-start processing parameters

Parameter	Description
CLI	<p>Set commands to be executed in the container for post-start processing. The command format is <b>Command Args[1] Args[2]...</b> <b>Command</b> is a system command or a user-defined executable program. If no path is specified, an executable program in the default path will be selected. If multiple commands need to be executed, write the commands into a script for execution. <b>Commands that are executed in the background or asynchronously are not supported.</b></p> <p>Example command: exec: command: - /install.sh - install_agent</p> <p>Enter <b>/install install_agent</b> in the script. This command indicates that <b>install.sh</b> will be executed after the container is created successfully.</p>

Parameter	Description
HTTP request	<p>Send an HTTP request for post-start processing. The related parameters are described as follows:</p> <ul style="list-style-type: none"> <li>● <b>Path:</b> (optional) request URL.</li> <li>● <b>Port:</b> (mandatory) request port.</li> <li>● <b>Host:</b> (optional) requested host IP address. The default value is the IP address of the pod.</li> </ul>

----End

## Pre-Stop Processing

**Step 1** Log in to the CCE console. When creating a workload, configure container information and select **Lifecycle**.

**Step 2** Set the pre-start processing parameters on the **Pre-Stop** tab page.

**Table 5-11** Pre-stop processing parameters

Parameter	Description
CLI	<p>Set commands to be executed in the container for pre-stop processing. The command format is <b>Command Args[1] Args[2]...</b> <b>Command</b> is a system command or a user-defined executable program. If no path is specified, an executable program in the default path will be selected. If multiple commands need to be executed, write the commands into a script for execution.</p> <p>Example command:</p> <pre>exec: command: - /uninstall.sh - uninstall_agent</pre> <p>Enter <b>/uninstall uninstall_agent</b> in the script. This command indicates that <b>uninstall.sh</b> will be executed before the container completes its execution and stops running.</p>
HTTP request	<p>Send an HTTP request for pre-stop processing. The related parameters are described as follows:</p> <ul style="list-style-type: none"> <li>● <b>Path:</b> (optional) request URL.</li> <li>● <b>Port:</b> (mandatory) request port.</li> <li>● <b>Host:</b> (optional) requested host IP address. The default value is the IP address of the pod.</li> </ul>

----End

## YAML Example

This section uses Nginx as an example to describe how to set the container lifecycle.

In the following configuration file, the **postStart** command is defined to run the **install.sh** command in the **/bin/bash** directory. **preStop** is defined to run the **uninstall.sh** command.

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx
spec:
  replicas: 1
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
        - image: nginx
          command:
            - sleep 3600                # Startup command
          imagePullPolicy: Always
          lifecycle:
            postStart:
              exec:
                command:
                  - /bin/bash
                  - install.sh          # Post-start command
            preStop:
              exec:
                command:
                  - /bin/bash
                  - uninstall.sh       # Pre-stop command
      name: nginx
      imagePullSecrets:
        - name: default-secret
```

### 5.3.6 Configuring Container Health Check

#### Scenario

Health check regularly checks the health status of containers during container running. If the health check function is not configured, a pod cannot detect application exceptions or automatically restart the application to restore it. This will result in a situation where the pod status is normal but the application in the pod is abnormal.

Kubernetes provides the following health check probes:

- **Liveness probe** (livenessProbe): checks whether a container is still alive. It is similar to the **ps** command that checks whether a process exists. If the liveness check of a container fails, the cluster restarts the container. If the liveness check is successful, no operation is executed.
- **Readiness probe** (readinessProbe): checks whether a container is ready to process user requests. Upon that the container is detected unready, service traffic will not be directed to the container. It may take a long time for some

applications to start up before they can provide services. This is because that they need to load disk data or rely on startup of an external module. In this case, although the application process has started, the application cannot provide services. To address this issue, this health check probe is used. If the container readiness check fails, the cluster masks all requests sent to the container. If the container readiness check is successful, the container can be accessed.

- **Startup probe** (startupProbe): checks when a containerized application has started. If such a probe is configured, it disables liveness and readiness checks until it succeeds, ensuring that those probes do not interfere with the application startup. This can be used to adopt liveness checks on slow starting containers, avoiding them getting terminated by the kubelet before they are started.

## Check Method

- **HTTP request**

This health check mode applies to containers that provide HTTP/HTTPS services. The cluster periodically initiates an HTTP/HTTPS GET request to such containers. If the return code of the HTTP/HTTPS response is within 200–399, the probe is successful. Otherwise, the probe fails. In this health check mode, you must specify a container listening port and an HTTP/HTTPS request path.

For example, for a container that provides HTTP services, the HTTP check path is **/health-check**, the port is 80, and the host address is optional (which defaults to the container IP address). Here, 172.16.0.186 is used as an example, and we can get such a request: GET http://172.16.0.186:80/health-check. The cluster periodically initiates this request to the container. You can also add one or more headers to an HTTP request. For example, set the request header name to **Custom-Header** and the corresponding value to **example**.

- **TCP port**

For a container that provides TCP communication services, the cluster periodically establishes a TCP connection to the container. If the connection is successful, the probe is successful. Otherwise, the probe fails. In this health check mode, you must specify a container listening port.

For example, if you have an Nginx container with service port 80, after you specify TCP port 80 for container listening, the cluster will periodically initiate a TCP connection to port 80 of the container. If the connection is successful, the probe is successful. Otherwise, the probe fails.

- **CLI**

CLI is an efficient tool for health check. When using the CLI, you must specify an executable command in a container. The cluster periodically runs the command in the container. If the command output is 0, the health check is successful. Otherwise, the health check fails.

The CLI mode can be used to replace the HTTP request-based and TCP port-based health check.

- For a TCP port, you can use a program script to connect to a container port. If the connection is successful, the script returns **0**. Otherwise, the script returns **-1**.
- For an HTTP request, you can use the script command to run the **wget** command to detect the container.

### wget http://127.0.0.1:80/health-check

Check the return code of the response. If the return code is within 200–399, the script returns **0**. Otherwise, the script returns **-1**.

#### NOTICE

- Put the program to be executed in the container image so that the program can be executed.
- If the command to be executed is a shell script, do not directly specify the script as the command, but add a script parser. For example, if the script is `/data/scripts/health_check.sh`, you must specify `sh/data/scripts/health_check.sh` for command execution.

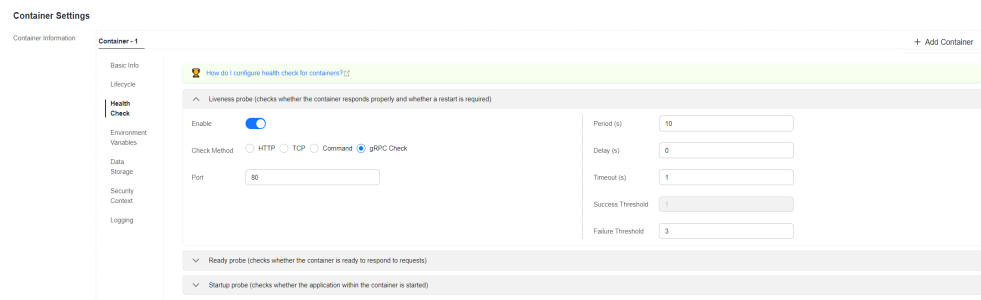
- **gRPC check**

gRPC checks can configure startup, liveness, and readiness probes for your gRPC application without exposing any HTTP endpoint, nor do you need an executable. Kubernetes can connect to your workload via gRPC and obtain its status.

#### NOTICE

- The gRPC check is supported only in CCE clusters of v1.25 or later.
- To use gRPC for check, your application must support the [gRPC health checking protocol](#).
- Similar to HTTP and TCP probes, if the port is incorrect or the application does not support the health checking protocol, the check fails.

Figure 5-5 gRPC check



## Common Parameters

Table 5-12 Common parameters

Parameter	Description
Period (periodSeconds)	Indicates the probe detection period, in seconds. For example, if this parameter is set to <b>30</b> , the detection is performed every 30 seconds.

Parameter	Description
Delay (initialDelaySeconds)	Check delay time in seconds. Set this parameter according to the normal startup time of services. For example, if this parameter is set to <b>30</b> , the health check will be started 30 seconds after the container is started. The time is reserved for containerized services to start.
Timeout (timeoutSeconds)	Number of seconds after which the probe times out. Unit: second. For example, if this parameter is set to <b>10</b> , the timeout wait time for performing a health check is 10s. If the wait time elapses, the health check is regarded as a failure. If the parameter is left blank or set to <b>0</b> , the default timeout time is 1s.
Success Threshold (successThreshold)	Minimum consecutive successes for the probe to be considered successful after having failed. For example, if this parameter is set to <b>1</b> , the workload status is normal only when the health check is successful for one consecutive time after the health check fails. The default value is <b>1</b> , which is also the minimum value. The value of this parameter is fixed to <b>1</b> in <b>Liveness Probe</b> and <b>Startup Probe</b> .
Failure Threshold (failureThreshold)	Number of retry times when the detection fails. Giving up in case of liveness probe means to restart the container. In case of readiness probe the pod will be marked Unready. The default value is <b>3</b> , and the minimum value is <b>1</b> .

## YAML Example

```

apiVersion: v1
kind: Pod
metadata:
  labels:
    test: liveness
  name: liveness-http
spec:
  containers:
  - name: liveness
    image: <image_address>
    args:
    - /server
    livenessProbe:
      # Liveness probe
      httpGet:
        # Checking an HTTP request is used as an example.
        path: /healthz
        # The HTTP check path is /healthz.
        port: 80
        # The check port number is 80.
        httpHeaders:
          # (Optional) The request header name is Custom-Header and the value is
          # Awesome.
          - name: Custom-Header
            value: Awesome
      initialDelaySeconds: 3
      periodSeconds: 3
    readinessProbe:
      # Readiness probe
  
```

```
exec:          # Checking an execution command is used as an example.
  command:     # Command to be executed
    - cat
    - /tmp/healthy
  initialDelaySeconds: 5
  periodSeconds: 5
  startupProbe: # Startup probe
  httpGet:      # Checking an HTTP request is used as an example.
    path: /healthz # The HTTP check path is /healthz.
    port: 80       # The check port number is 80.
  failureThreshold: 30
  periodSeconds: 10
```

## 5.3.7 Configuring Environment Variables

### Scenario

An environment variable is a variable whose value can affect the way a running container will behave. You can modify environment variables even after workloads are deployed, increasing flexibility in workload configuration.

The function of setting environment variables on CCE is the same as that of specifying **ENV** in a Dockerfile.

---

#### NOTICE

After a container is started, do not modify configurations in the container. If configurations in the container are modified (for example, passwords, certificates, and environment variables of a containerized application are added to the container), the configurations will be lost after the container restarts and container services will become abnormal. An example scenario of container restart is pod rescheduling due to node anomalies.

Configurations must be imported to a container as arguments. Otherwise, configurations will be lost after the container restarts.

---

Environment variables can be set in the following modes:

- **Custom:** Enter the environment variable name and parameter value.
- **Added from ConfigMap:** Import all key values in a ConfigMap as environment variables.
- **Added from ConfigMap key:** Import the value of a key in a ConfigMap as the value of an environment variable. As shown in [Figure 5-6](#), if you import **configmap\_value** of **configmap\_key** in **configmap-example** as the value of environment variable **key1**, an environment variable named **key1** whose value is **configmap\_value** is available in the container.
- **Added from secret:** Import all key values in a secret as environment variables.
- **Added from secret key:** Import the value of a key in a secret as the value of an environment variable. As shown in [Figure 5-6](#), if you import **secret\_value** of **secret\_key** in **secret-example** as the value of environment variable **key2**, an environment variable named **key2** whose value is **secret\_value** is available in the container.
- **Variable Value/Reference:** Use the field defined by a pod as the value of the environment variable. As shown in [Figure 5-6](#), if the pod name is imported as



the value of environment variable **key3**, an environment variable named **key3** whose value is the pod name is available in the container.

- **Resource Reference:** The value of **Request** or **Limit** defined by the container is used as the value of the environment variable. As shown in [Figure 5-6](#), if you import the CPU limit of container-1 as the value of environment variable **key4**, an environment variable named **key4** whose value is the CPU limit of container-1 is available in the container.

## Adding Environment Variables

**Step 1** Log in to the CCE console.

**Step 2** Click the cluster name to go to the cluster console, choose **Workloads** in the navigation pane, and click the **Create Workload** in the upper right corner.

**Step 3** When creating a workload, modify the container information in **Container Settings** and click the **Environment Variables** tab.

**Step 4** Configure environment variables.

- To add environment variables one by one, click **Adding a Variable** and configure its parameters.
- To add environment variables in batches, click **Editing Custom Variables in Batches**. Then, in the displayed dialog box, enter environment variables in the format of "Variable name=Variable or variable reference".

**Figure 5-6** Configuring environment variables

Type	Variable Name	Variable Value/Reference	
Custom	key	value	
Added from ConfigMap key	key1	configmap-example	configmap_key
Added from secret key	key2	secret-example	secret_key
Variable Value/Reference	key3	metadata.name	
Resource Reference	key4	container-1	limits.cpu
Added from ConfigMap		configmap-example	
Added from secret		secret-example	

----End

## YAML Example

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: env-example
  namespace: default
spec:
  replicas: 1
  selector:
    matchLabels:
      app: env-example
  template:
    metadata:
      labels:
        app: env-example
    spec:
      containers:
```

```

- name: container-1
  image: nginx:alpine
  imagePullPolicy: Always
  resources:
    requests:
      cpu: 250m
      memory: 512Mi
    limits:
      cpu: 250m
      memory: 512Mi
  env:
    - name: key                # Custom
      value: value
    - name: key1              # Added from ConfigMap key
      valueFrom:
        configMapKeyRef:
          name: configmap-example
          key: configmap_key
    - name: key2              # Added from secret key
      valueFrom:
        secretKeyRef:
          name: secret-example
          key: secret_key
    - name: key3              # Variable reference, which uses the field defined by a pod as the value
of the environment variable.
      valueFrom:
        fieldRef:
          apiVersion: v1
          fieldPath: metadata.name
    - name: key4              # Resource reference, which uses the field defined by a container as the
value of the environment variable.
      valueFrom:
        resourceFieldRef:
          containerName: container1
          resource: limits.cpu
          divisor: 1
    envFrom:
      - configMapRef:          # Added from ConfigMap
        name: configmap-example
      - secretRef:            # Added from secret
        name: secret-example
  imagePullSecrets:
    - name: default-secret

```

## Viewing Environment Variables

If the contents of **configmap-example** and **secret-example** are as follows:

```

$ kubectl get configmap configmap-example -oyaml
apiVersion: v1
data:
  configmap_key: configmap_value
kind: ConfigMap
...

$ kubectl get secret secret-example -oyaml
apiVersion: v1
data:
  secret_key: c2VjcmV0X3ZhbHVL          # c2VjcmV0X3ZhbHVL is the value of secret_value in Base64
mode:
kind: Secret
...

```

The environment variables in the pod are as follows:

```

$ kubectl get pod
NAME                                READY STATUS RESTARTS AGE
env-example-695b759569-lx9jp        1/1   Running 0    17m

```

```
$ kubectl exec env-example-695b759569-lx9jp -- printenv
/ # env
key=value # Custom environment variable
key1=configmap_value # Added from ConfigMap key
key2=secret_value # Added from secret key
key3=env-example-695b759569-lx9jp # metadata.name defined by the pod
key4=1 # limits.cpu defined by container1. The value is rounded up, in unit of cores.
configmap_key=configmap_value # Added from ConfigMap. The key value in the original ConfigMap
secret_key=secret_value # Added from key. The key value in the original secret is directly imported.
```

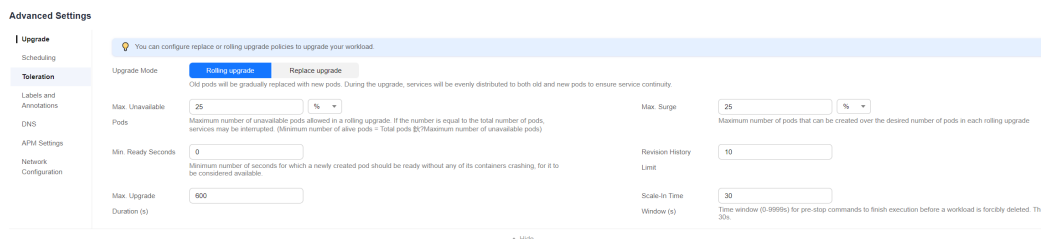
### 5.3.8 Configuring Workload Upgrade Policies

In actual applications, upgrade is a common operation. A Deployment, StatefulSet, or DaemonSet can easily support application upgrade.

You can set different upgrade policies:

- **Rolling upgrade:** New pods are created gradually and then old pods are deleted. This is the default policy.
- **Replace upgrade:** The current pods are deleted and then new pods are created.

Figure 5-7 Workload upgrade mode



### Upgrade Parameters

Parameter	Description	Constraint
Max. Surge (maxSurge)	Specifies the maximum number of pods that can exist compared with <b>spec.replicas</b> . The default value is <b>25%</b> .  For example, if <b>spec.replicas</b> is set to <b>4</b> , a maximum of five pods can exist during the upgrade. That is, the upgrade is performed at a step of 1. During the actual upgrade, the value is converted into a number and rounded up. The value can also be set to an absolute number.	This parameter is supported only by Deployments and DaemonSets.

Parameter	Description	Constraint
Max. Unavailable Pods (maxUnavailable)	Specifies the maximum number of pods that can be unavailable compared with <b>spec.replicas</b> . The default value is <b>25%</b> . For example, if <b>spec.replicas</b> is set to <b>4</b> , at least three pods exist during the upgrade. That is, the deletion is performed at a step of 1. The value can also be set to an absolute number.	This parameter is supported only by Deployments and DaemonSets.
Min. Ready Seconds (minReadySeconds)	A pod is considered available only when the minimum readiness time is exceeded without any of its containers crashing. The default value is <b>0</b> (the pod is considered available immediately after it is ready).	None
Revision History Limit (revisionHistoryLimit)	Specifies the number of old ReplicaSets to retain to allow rollback. These old ReplicaSets consume resources in etcd and crowd the output of <b>kubectl get rs</b> . The configuration of each Deployment revision is stored in its ReplicaSets. Therefore, once the old ReplicaSet is deleted, you lose the ability to roll back to that revision of Deployment. By default, 10 old ReplicaSets will be kept, but the ideal value depends on the frequency and stability of the new Deployments.	None
Max. Upgrade Duration (progressDeadlineSeconds)	Specifies the number of seconds that the system waits for a Deployment to make progress before reporting a Deployment progress failure. It is surfaced as a condition with Type=Progressing, Status=False, and Reason=ProgressDeadlineExceeded in the status of the resource. The Deployment controller will keep retrying the Deployment. In the future, once automatic rollback will be implemented, the Deployment controller will roll back a Deployment as soon as it observes such a condition.  If this parameter is specified, the value of this parameter must be greater than that of <b>.spec.minReadySeconds</b> .	None

Parameter	Description	Constraint
Scale-In Time Window (terminationGracePeriodSeconds)	Graceful deletion time. The default value is 30 seconds. When a pod is deleted, a SIGTERM signal is sent and the system waits for the applications in the container to terminate. If the application is not terminated within the time specified by <b>terminationGracePeriodSeconds</b> , a SIGKILL signal is sent to forcibly terminate the pod.	None

## Upgrade Example

The Deployment can be upgraded in a declarative mode. That is, you only need to modify the YAML definition of the Deployment. For example, you can run the **kubectl edit** command to change the Deployment image to **nginx:alpine**. After the modification, query the ReplicaSet and pod. The query result shows that a new ReplicaSet is created and the pod is re-created.

```
$ kubectl edit deploy nginx

$ kubectl get rs
NAME                DESIRED  CURRENT  READY  AGE
nginx-6f9f58dff  2        2        2      1m
nginx-7f98958cdf  0        0        0      48m

$ kubectl get pods
NAME                READY  STATUS   RESTARTS  AGE
nginx-6f9f58dff-tdmqk  1/1    Running  0         1m
nginx-6f9f58dff-tesqr  1/1    Running  0         1m
```

The Deployment can use the **maxSurge** and **maxUnavailable** parameters to control the proportion of pods to be re-created during the upgrade, which is useful in many scenarios. The configuration is as follows:

```
spec:
  strategy:
    rollingUpdate:
      maxSurge: 1
      maxUnavailable: 0
    type: RollingUpdate
```

In the preceding example, the value of **spec.replicas** is 2. If both **maxSurge** and **maxUnavailable** are the default value 25%, **maxSurge** allows a maximum of three pods to exist ( $2 \times 1.25 = 2.5$ , rounded up to 3), and **maxUnavailable** does not allow a maximum of two pods to be unavailable ( $2 \times 0.75 = 1.5$ , rounded up to 2). That is, during the upgrade process, there will always be two pods running. Each time a new pod is created, an old pod is deleted, until all pods are new.

## Rollback

Rollback is to roll an application back to an earlier version when a fault occurs during an upgrade. A Deployment can be easily rolled back to the earlier version.

For example, if the upgraded image is faulty, you can run the **kubectl rollout undo** command to roll back the Deployment.

```
$ kubectl rollout undo deployment nginx
deployment.apps/nginx rolled back
```

A Deployment can be easily rolled back because it uses a ReplicaSet to control a pod. After the upgrade, the previous ReplicaSet still exists. The Deployment is rolled back by using the previous ReplicaSet to re-create the pod. The number of ReplicaSets stored in a Deployment can be restricted by the **revisionHistoryLimit** parameter. The default value is **10**.

### 5.3.9 Configuring Tolerance Policies

Tolerations allow the scheduler to schedule pods to nodes with target taints. Tolerances work with **node taints**. Each node allows one or more taints. If no tolerance is configured for a pod, the scheduler will schedule the pod based on node taint policies to prevent the pod from being scheduled to an inappropriate node. For more examples, see **Taints and Tolerations**.

The following table shows how taint policies and tolerations affect pod running.

Taint Policy	No Taint Toleration Configured	Taint Toleration Configured
NoExecute	<ul style="list-style-type: none"> <li>Pods running on the node will be evicted immediately.</li> <li>Inactive pods will not be scheduled to the node.</li> </ul>	<ul style="list-style-type: none"> <li>If the tolerance time window <b>tolerationSeconds</b> is not specified, pods can always run on this node.</li> <li>If the tolerance time window <b>tolerationSeconds</b> is specified, pods still run on the node with taints within the time window. After the time expires, the pods will be evicted.</li> </ul>
PreferNoSchedule	<ul style="list-style-type: none"> <li>Pods running on the node will not be evicted.</li> <li>Inactive pods will not be scheduled to the node <b>to the best extend</b>.</li> </ul>	Pods can always run on this node.
NoSchedule	<ul style="list-style-type: none"> <li>Pods running on the node will not be evicted.</li> <li>Inactive pods will not be scheduled to the node.</li> </ul>	Pods can always run on this node.

### Configuring Tolerance Policies on the Console

- Step 1** Log in to the CCE console.
- Step 2** When creating a workload, click **Toleration** in the **Advanced Settings** area.
- Step 3** Add a taint tolerance policy.

**Table 5-13** Parameters for configuring a taint tolerance policy

Parameter	Description
Taint key	Key of a node taint
Operator	<ul style="list-style-type: none"> <li>• <b>Equal: Exact match</b> for the specified taint key (mandatory) and taint value. If the taint value is left blank, all taints with the key the same as the specified taint key will be matched.</li> <li>• <b>Exists: matches only</b> the nodes with the specified taint key. In this case, the taint value cannot be specified. If the taint key is left blank, all taints will be tolerated.</li> </ul>
Taint value	Taint value specified if the operator is set to <b>Equal</b> .
Taint Policy	<ul style="list-style-type: none"> <li>• <b>All:</b> All taint policies are matched.</li> <li>• <b>NoSchedule:</b> Only the <b>NoSchedule</b> taint is matched.</li> <li>• <b>PreferNoSchedule:</b> Only the <b>PreferNoSchedule</b> taint is matched.</li> <li>• <b>NoExecute:</b> Only the <b>NoExecute</b> taint is matched.</li> </ul>
Toleration Time Window	<p><b>tolerationSeconds</b>, which is configurable only when <b>Taint Policy</b> is set to <b>NoExecute</b>.</p> <p>Within the tolerance time window, pods still run on the node with taints. After the time expires, the pods will be evicted.</p>

----End

## Default Tolerance Policy

Kubernetes automatically adds tolerances for the **node.kubernetes.io/not-ready** and **node.kubernetes.io/unreachable** taints to pods, and sets the tolerance time window (**tolerationSeconds**) to 300s. These default tolerance policies indicate that when either of the preceding taint is added to the node where pods are running, the pods can still run on the node for 5 minutes.

### NOTE

When a DaemonSet pod is created, no tolerance time window will be specified for the tolerances automatically added for the preceding taints. When either of the preceding taints is added to the node where the DaemonSet pod is running, the DaemonSet pod will never be evicted.

```
tolerations:
- key: node.kubernetes.io/not-ready
  operator: Exists
  effect: NoExecute
  tolerationSeconds: 300
- key: node.kubernetes.io/unreachable
  operator: Exists
  effect: NoExecute
  tolerationSeconds: 300
```

## 5.3.10 Configuring Labels and Annotations

### Pod Annotations

CCE allows you to add annotations to a YAML file to realize some advanced pod functions. The following table describes the annotations you can add.

**Table 5-14** Pod annotations

Annotation	Description	Default Value
kubernetes.AOM.log.stdout	Standard output parameter. If not specified, the standard log output of all containers is reported to AOM. You can collect stdout logs from certain containers or ignore them at all. Example: <ul style="list-style-type: none"> <li>Collecting none of the stdout logs: kubernetes.AOM.log.stdout: '[]'</li> <li>Collecting stdout logs of container-1 and container-2: kubernetes.AOM.log.stdout: '["container-1","container-2"]'</li> </ul>	None
metrics.alpha.kubernetes.io/custom-endpoints	Parameter for reporting AOM monitoring metrics that you specify. For details, see <a href="#">Monitoring Custom Metrics on AOM</a> .	None
prometheus.io/scrape	Parameter for reporting Prometheus metrics. If the value is <b>true</b> , the current workload reports the monitoring metrics. For details, see <a href="#">Monitoring Custom Metrics Using Cloud Native Cluster Monitoring</a> .	None
prometheus.io/path	URL for Prometheus to collect data. For details, see <a href="#">Monitoring Custom Metrics Using Cloud Native Cluster Monitoring</a> .	/metrics
prometheus.io/port	Endpoint port number for Prometheus to collect data. For details, see <a href="#">Monitoring Custom Metrics Using Cloud Native Cluster Monitoring</a> .	None



Annotation	Description	Default Value
prometheus.io/scheme	Protocol used by Prometheus to collect data. The value can be <b>http</b> or <b>https</b> . For details, see <a href="#">Monitoring Custom Metrics Using Cloud Native Cluster Monitoring</a> .	None

## Pod Labels

When you create a workload on the console, the following labels are added to the pod by default. The value of **app** is the workload name.

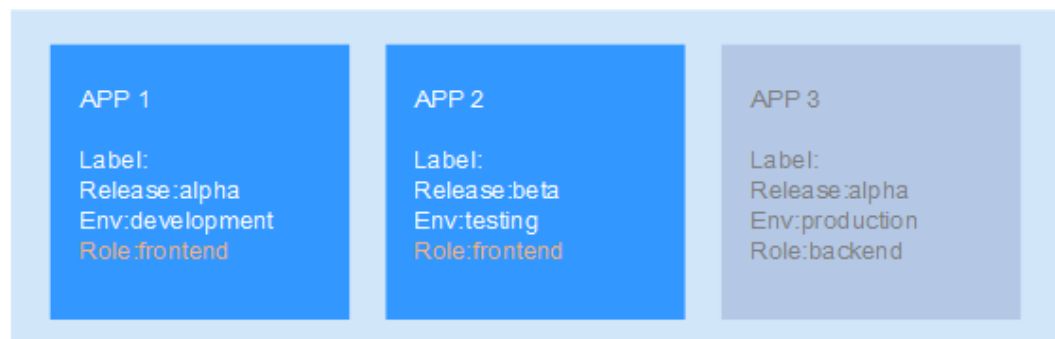
Example YAML:

```
...
spec:
  selector:
    matchLabels:
      app: nginx
      version: v1
  template:
    metadata:
      labels:
        app: nginx
        version: v1
    spec:
      ...
```

You can also add other labels to the pod for affinity and anti-affinity scheduling. In the following figure, three pod labels (release, env, and role) are defined for workload APP 1, APP 2, and APP 3. The values of these labels vary with workload.

- APP 1: [release:alpha;env:development;role:frontend]
- APP 2: [release:beta;env:testing;role:frontend]
- APP 3: [release:alpha;env:production;role:backend]

**Figure 5-8** Label example



For example, if **key/value** is set to **role/backend**, APP 3 will be selected for affinity scheduling. For details, see [Configuring Workload Affinity or Anti-affinity Scheduling \(podAffinity or podAntiAffinity\)](#).

## 5.4 Scheduling a Workload

### 5.4.1 Overview

Kubernetes schedules workloads based on pods. After you create a workload, the scheduler automatically assigns pods. For example, the scheduler distributes pods to nodes that have enough resources.

While the default scheduler behavior is sufficient for most needs, there may be situations where you require more precise control over where pods are deployed. To address this, Kubernetes enables you to configure scheduling policies when creating workloads. For example, you may need to:

- Deploy a frontend application and a backend application on the same node. This reduces latency because the pods of both applications can use the physical resources of the node.
- Deploy a certain type of applications on specific nodes to ensure that these critical applications always run on the best hardware or configuration.
- Deploy different applications on separate nodes to isolate them from each other. This prevents any issues with one application from affecting others.

Use the methods listed in the following table to select a pod scheduling policy in Kubernetes.

**Table 5-15** Workload scheduling policies

Scheduling Policy	YAML Field	Description	Reference
Node selector	nodeSelector	A basic scheduling mode, in which Kubernetes selects the target node according to node labels. This means that pods are only scheduled to the node that has the specific label.	<a href="#">Configuring Specified Node Scheduling (nodeSelector)</a>
Node affinity	nodeAffinity	Node affinity is more expressive than nodeSelector. It allows you to use <a href="#">Label Selectors</a> to filter nodes that require affinity based on their labels and choose between <b>Required</b> and <b>Preferred</b> for <a href="#">Affinity Rules</a> . <b>NOTE</b> When both nodeSelector and nodeAffinity are specified, a pod can only be scheduled to a candidate node if both conditions are met.	<a href="#">Configuring Node Affinity Scheduling (nodeAffinity)</a>

Scheduling Policy	YAML Field	Description	Reference
Workload affinity or anti-affinity scheduling	podAffinity/ podAntiAffinity	<p>You can use <a href="#">Label Selectors</a> to filter pods that require affinity or anti-affinity based on workload labels. You can then determine whether to schedule new workload pods to the same node (or node group) as the target pod or not. Additionally, you can choose between <b>Required</b> and <b>Preferred</b> for <a href="#">Affinity Rules</a>.</p> <p><b>NOTE</b> Workload affinity and anti-affinity require a certain amount of computing time, which significantly slows down scheduling in large-scale clusters. Do not enable workload affinity and anti-affinity in a cluster that contains hundreds of nodes.</p>	<a href="#">Configuring Workload Affinity or Anti-affinity Scheduling (podAffinity or podAntiAffinity)</a>

## Affinity Rules

Scheduling policies that use node affinity or workload affinity/anti-affinity can include both hard and soft constraints to meet complex scheduling requirements. Hard constraints must be met, while soft constraints should be met as much as possible.

**Table 5-16** Affinity rules

Rule Type	YAML Field	Description	Example
Required	<code>requiredDuringSchedulingIgnoredDuringExecution</code>	Hard constraint that <b>must be met</b> . The scheduler can perform scheduling only when the rule is met.	<ul style="list-style-type: none"> <li>• <a href="#">Configuring Node Affinity Scheduling (nodeAffinity)</a></li> <li>• <a href="#">Configuring Workload Affinity or Anti-affinity Scheduling (podAffinity or podAntiAffinity)</a></li> </ul>
Preferred	<code>preferredDuringSchedulingIgnoredDuringExecution</code>	Soft constraint. The scheduler tries to locate the target object that satisfies the target rule. The scheduler will schedule the pod even if it cannot find a matching target object that satisfies the target rule.  When using preferred affinity, you can set a weight field ranging from 1 to 100 for each pod. Assigning a higher weight to a pod will increase its priority in the scheduling process.	

 **NOTE**

The YAML field **requiredDuringScheduling** or **preferredDuringScheduling** in the affinity rules above indicates that a label rule must be forcibly met or needs to be met as much as possible during scheduling. **IgnoredDuringExecution** indicates that any changes to the node label after Kubernetes schedules the pod will not affect the pod's running or cause it to be rescheduled. However, if kubelet on the node is restarted, kubelet will recheck the node affinity rule, and the pod will still be scheduled to another node.

## Label Selectors

When creating a scheduling policy use the logical operators of a label selector to filter label values and identify the objects that need affinity or anti-affinity.

**Table 5-17** Label selectors

Parameter	Description	YAML Example
key	Label key. The objects that meet the filter criteria must contain the label of the key, and the label value must meet the operation relationship between the label value list ( <b>values</b> field) and logical operators.	In the example below, objects that meet the filter criteria must have a label with a key of <b>topology.kubernetes.io/zone</b> and a value of either <b>az1</b> or <b>az2</b> . <pre> matchExpressions: - key: topology.kubernetes.io/zone   operator: In   values:   - az1   - az2                     </pre>
operator	Logical operator that can be used to determine filtering rules for label values. Options: <ul style="list-style-type: none"> <li>• <b>In</b>: The label of the affinity or anti-affinity object <b>is</b> in the label value list (<b>values</b> field).</li> <li>• <b>NotIn</b>: The label of the affinity or anti-affinity object is <b>not</b> in the label value list (<b>values</b> field).</li> <li>• <b>Exists</b>: The affinity or anti-affinity object <b>has</b> a specified label key. There is no need to configure the label value list (<b>values</b> field).</li> <li>• <b>DoesNotExist</b>: The affinity or anti-affinity object does <b>not</b> have a specified label key. There is no need to configure the label value list (<b>values</b> field).</li> <li>• <b>Gt</b>: (available only for node affinity) The label value of the scheduled node is <b>greater</b> than what is listed (string comparison).</li> <li>• <b>Lt</b>: (available only for node affinity) The label value of the scheduled node is <b>less</b> than what is listed (string comparison).</li> </ul>	
values	Label values	

## 5.4.2 Configuring Specified Node Scheduling (nodeSelector)

To select a node for scheduling in Kubernetes, simply configure the **nodeSelector** field in the workload. This field allows you to configure the label of the desired node to be scheduled. Kubernetes schedules pods only to nodes with specified labels.

### Prerequisites

A custom label has been added to the target node so that workload pods can be scheduled based on the node label. For details, see [Adding or Deleting a Node Label](#).

## Creating a Workload Scheduled to a Specified Node

- Step 1** Use `kubectl` to access the cluster. For details, see [Connecting to a Cluster Using kubectl](#).
- Step 2** Create a YAML file named `nginx.yaml`. The file name can be customized.

Configure `nodeSelector` for the workload. For example, if the key is `deploy_qa` and the value is `true`, the pod will be scheduled to the node with the `deploy_qa=true` label. Example:

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx
spec:
  replicas: 3
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      nodeSelector:
        deploy_qa: "true"
      containers:
        - image: nginx:latest
          imagePullPolicy: IfNotPresent
          name: nginx
          imagePullSecrets:
            - name: default-secret
```

- Step 3** Create a workload.

```
kubectl apply -f nginx.yaml
```

- Step 4** Verify that all pods run on the target node.

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

In the following command output, node 192.168.0.103 is labeled with `deploy_qa=true`:

NAME	READY	STATUS	RESTARTS	AGE	IP	NODE
NOMINATED NODE READINESS GATES						
nginx-66859f4f48-xgp2g	1/1	Running	0	6h57m	172.16.3.0	192.168.0.103 <none>
nginx-66859f4f48-t9gqj	1/1	Running	0	6h57m	172.16.3.1	192.168.0.103 <none>
nginx-66859f4f48-2grhq	1/1	Running	0	6h57m	172.16.3.2	192.168.0.103 <none>

----End

### 5.4.3 Configuring Node Affinity Scheduling (nodeAffinity)

Kubernetes can schedule workload pods to affinity nodes based on their labels and label values. For example, some nodes support GPU computing, and node affinity scheduling can guarantee that high-performance computing pods run on these GPU nodes.

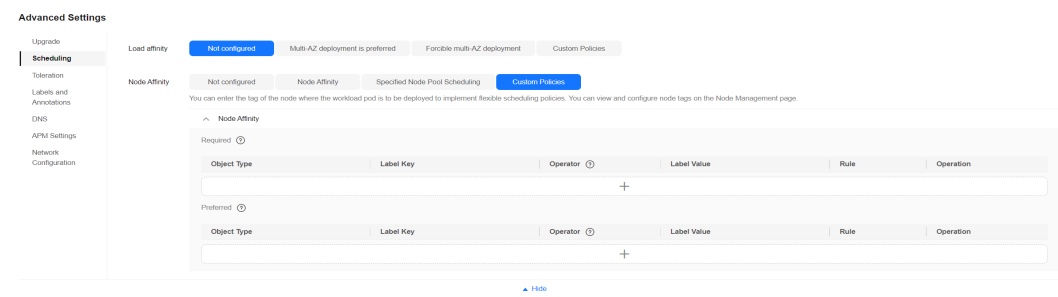
## Configuring a Node Affinity Scheduling Policy

There are various ways to configure node affinity scheduling policies to schedule pods to nodes that meet specific requirements.

### Using the CCE Console

In this example, a GPU node labeled with **gpu=true** has been created in the cluster. You can use this label to schedule pods to this GPU node.

- Step 1** Log in to the CCE console.
- Step 2** Click the cluster name to go to the cluster console, choose **Workloads** in the navigation pane, and click the **Create Workload** in the upper right corner.
- Step 3** In **Advanced Settings**, choose **Scheduling** and select a policy for **Node Affinity**. In this example, **Custom policies** is selected. For details about how to create a workload, see [Creating a Workload](#).



**Table 5-18** Scheduling policies

Parameter	Description	Example
Node Affinity	<ul style="list-style-type: none"> <li>● <b>Not configured:</b> No node affinity policy is configured.</li> <li>● <b>Node Affinity:</b> Specify the nodes where workload pods are to be deployed. If no nodes are specified, the pods will be randomly scheduled according to the default cluster scheduling policy.</li> <li>● <b>Specified Node Pool Scheduling:</b> Specify the node pools where workload pods are to be deployed. If no node pools are specified, the pods will be randomly scheduled according to the default cluster scheduling policy.</li> <li>● <b>Custom policies:</b> allow flexible scheduling of workload pods based on node labels. For details about the supported affinity policies, see <a href="#">Table 5-19</a>. Select a proper policy type and add a policy. For details about the parameters, see <a href="#">Table 5-20</a>.</li> </ul>	Custom policies

- Step 4** Select a proper node affinity rule and click **+** to add a scheduling policy. In this example, a scheduling policy is added in the **Required** area under **Node Affinity**.

This policy specifies that the newly created workload can only be scheduled to a specific node.

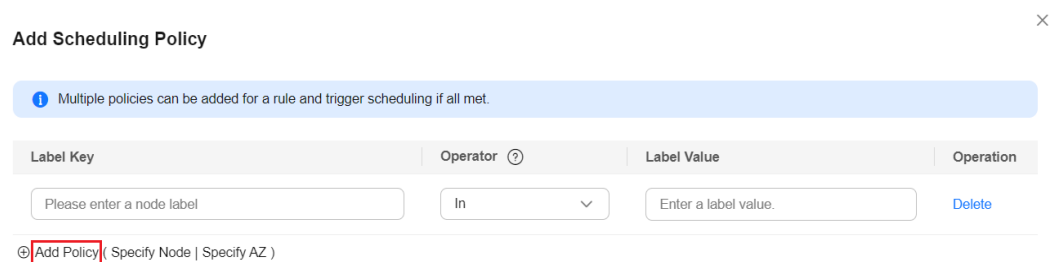
**Table 5-19** Affinity rule

Parameter	Description	Example
Node Affinity	<ul style="list-style-type: none"> <li>• <b>Required:</b> indicates the hard constraint (<b>requiredDuringSchedulingIgnoredDuringExecution</b>) that must be met. If multiple rules that must be met are added, scheduling will be performed when only one rule is met.</li> <li>• <b>Preferred:</b> indicates the soft constraint (<b>preferredDuringSchedulingIgnoredDuringExecution</b>) that need to be met as much as possible. If multiple rules that are preferentially met are added, scheduling will be performed even if one or none of the rules is met.</li> </ul>	Required

**Step 5** In the dialog box that is displayed on the right, click **Add Policy** to configure rules for filtering node labels.

You can also click **Specify Node** or **Specify AZ** to quickly select a node or AZ on the console for scheduling.

Specifying a node or AZ is also implemented through labels. The console frees you from manually entering node labels. The **kubernetes.io/hostname** label is used when you specify a node, and the **failure-domain.beta.kubernetes.io/zone** label is used when you specify an AZ.





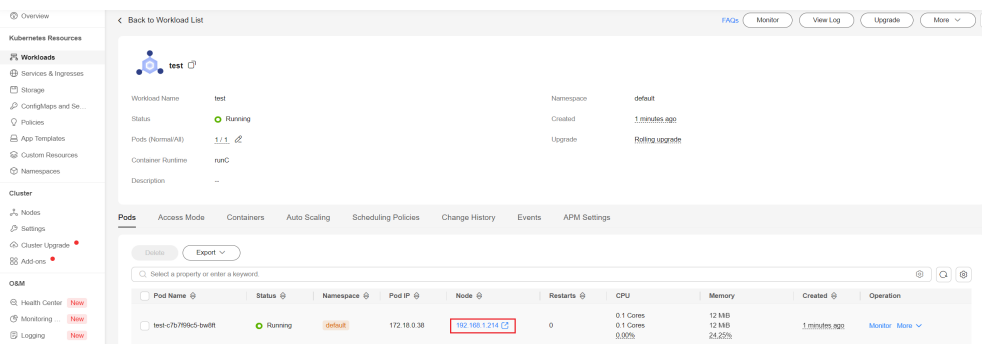
**Table 5-20** Parameters for configuring node affinity scheduling policies

Parameter	Description	Example
Weight	This parameter is available only in a <b>preferred</b> scheduling policy. Weights range from 1 to 100 and are taken into account as an extra scoring factor during scheduling. The scheduler combines the weight with other priority functions of the node to determine the final score and then assigns pods to the node with the highest total score.	None
Label Key	When configuring node affinity, enter the node label to be matched. Both default labels and custom labels are supported.	gpu
Operator	The following operators are supported: <ul style="list-style-type: none"> <li>● <b>In</b>: The label of the affinity or anti-affinity object is in the label value list (<b>values</b> field).</li> <li>● <b>NotIn</b>: The label of the affinity or anti-affinity object is not in the label value list (<b>values</b> field).</li> <li>● <b>Exists</b>: The affinity or anti-affinity object has a specified label key.</li> <li>● <b>DoesNotExist</b>: The affinity or anti-affinity object does not have a specified label key.</li> <li>● <b>Gt</b>: The label value of the scheduling node is greater than what is listed (character string comparison).</li> <li>● <b>Lt</b>: The label value of the scheduling node is less than what is listed (character string comparison).</li> </ul>	In
Label Value	When configuring node affinity, enter the value of the node label.	true

**Step 6** After the scheduling policy is added, click **Create Workload**.

**Step 7** Verify that all pods run on the target node.

1. In the navigation pane, choose **Workloads**.
2. Click the workload name to enter its details page. On the **Pods** tab page, verify that all pods run on the target node, which is labeled with **gpu=true**.



----End

## Using YAML

Workload node affinity rules are implemented using node labels. When a node is created in a CCE cluster, it is automatically assigned certain labels. Here are some examples of commonly used node labels (for more labels, see [Inherent Label of a Node](#)):

- **topology.kubernetes.io/zone**: the AZ where the node is located, which can be used for scheduling in a specified AZ.
- **kubernetes.io/hostname**: hostname of a node, which can be used for specified node scheduling.
- **cce.cloud.com/cce-nodepool**: node pool to which a node belongs, which can be used for scheduling in a specified node pool.

In this example, the rule that **must be met** indicates that the node to be scheduled must be labeled with key **gpu** and value **true**. The rule that needs to be **met as much as possible** indicates that pods are preferentially scheduled to nodes in AZ 1 based on **topology.kubernetes.io/zone**. The following is an example of configuring node affinity:

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: gpu
  labels:
    app: gpu
spec:
  selector:
    matchLabels:
      app: gpu
  replicas: 3
  template:
    metadata:
      labels:
        app: gpu
    spec:
      containers:
        - image: nginx:alpine
          name: gpu
          resources:
            requests:
              cpu: 100m
              memory: 200Mi
            limits:
              cpu: 100m
              memory: 200Mi
      imagePullSecrets:
        - name: default-secret
      affinity: # Configure a scheduling policy.
        nodeAffinity: # Node affinity scheduling
          requiredDuringSchedulingIgnoredDuringExecution: # Scheduling policy that must be met
            nodeSelectorTerms: # Select a node that meets the requirements according to the node label.
              - matchExpressions: # Node label matching rule
                - key: gpu # The key of the node label is gpu.
                  operator: In # The rule is met if a value exists in the value list.
                  values: # The value of the node label is true.
                    - "true"
          preferredDuringSchedulingIgnoredDuringExecution: # Scheduling policy that is met as much as possible
            - weight: 100 # Priority that can be configured when the best-effort policy is used. The value ranges from 1 to 100. A larger value indicates a higher priority.
```

```
preference: # Preferred node label matching rule when the best-effort policy is used
matchExpressions: # Node label matching rule
- key: topology.kubernetes.io/zone # Nodes' AZs
  operator: In # The rule is met if a value exists in the value list.
  values: # The value of the node label is az1.
- "az1"
```

 NOTE

There is no anti-affinity policy for node affinity scheduling. For node anti-affinity, you can filter nodes by label using the **NotIn** and **DoesNotExist** operators.

## 5.4.4 Configuring Workload Affinity or Anti-affinity Scheduling (podAffinity or podAntiAffinity)

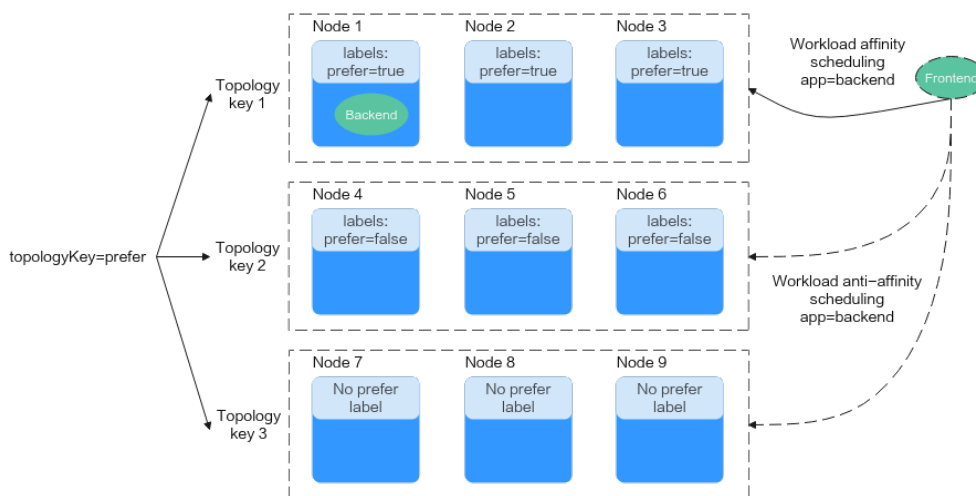
Kubernetes offers workload affinity and anti-affinity scheduling, which allows for flexible scheduling of new workloads on either related or unrelated nodes. This results in improved cluster resource utilization.

For example, frontend workload pods and backend workload pods that frequently communicate with each other can be preferentially scheduled to the same node or AZ to minimize network latency. The process of workload affinity/anti-affinity is as follows:

1. Nodes are classified using node labels based on topology keys (**topologyKey**).  
For example, if **topologyKey** is **prefer**, nodes will be classified using node label **prefer**. Nodes labeled with **prefer=true** will be assigned to topology key 1, while those labeled with **prefer=false** will be assigned to topology key 2. Nodes without a **prefer** label will be assigned to topology key 3.
2. Identify the affinity/anti-affinity workloads based on the workload labels and operators.  
For example, the label selector filters the workloads that have been labeled with **app=backend**.
3. For affinity scheduling, the scheduler chooses the topology key where the target workload is located, while for anti-affinity scheduling, it selects a topology key where the target workload is not present.

In this example, the workload labeled with **app=backend** is assigned to topology key 1. Therefore, workloads that have an affinity with the **app=backend** workload can be scheduled to topology key 1. Conversely, workloads that have an anti-affinity with the **app=backend** workload can only be scheduled to topology key 2 or 3.

Figure 5-9 Workload affinity or anti-affinity scheduling



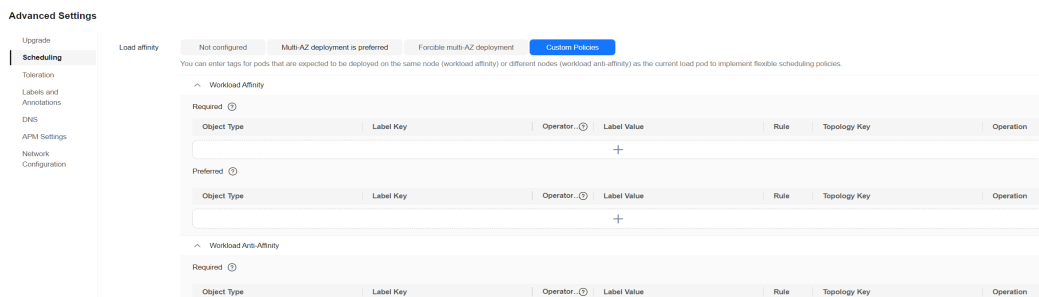
## Configuring Load Affinity/Anti-affinity

There are various ways to configure load affinity or anti-affinity scheduling policies to schedule pods to nodes that meet specific requirements.

### Using the CCE Console


In this example, a backend workload with the **app=backend** label has been created in the cluster. This label can be used for workload affinity or anti-affinity scheduling, allowing the newly created frontend workload labeled with **app=frontend** to be deployed on the same node as the backend workload. Both workloads run in topology key **kubernetes.io/hostname**. When you use **kubernetes.io/hostname** for topology classification, only one node can be assigned to each topology key because each node has a unique hostname. This enables workloads to be scheduled on the same node when workload affinity is enabled.

- Step 1** Log in to the CCE console.
- Step 2** Click the cluster name to go to the cluster console, choose **Workloads** in the navigation pane, and click the **Create Workload** in the upper right corner.
- Step 3** In **Advanced Settings**, choose **Scheduling** and select a policy for **Load Affinity**. In this example, **Custom policies** is selected. For details about how to create a workload, see [Creating a Workload](#).



**Table 5-21** Scheduling policies

Parameter	Description	Example
Load affinity	<ul style="list-style-type: none"> <li>• <b>Not configured:</b> No load affinity policy is configured.</li> <li>• <b>Multi-AZ deployment preferred:</b> Workload pods are <b>preferentially</b> scheduled to nodes in different AZs through pod anti-affinity. The AZs serve as topology keys in this process.</li> <li>• <b>Forcible multi-AZ deployment:</b> Workload pods are <b>forcibly</b> scheduled to nodes in different AZs through pod anti-affinity. The AZs serve as topology keys in this process. When this scheduling policy is used, if there are fewer nodes than pods or node resources are insufficient, the extra pods will fail to run.</li> <li>• <b>Custom policies:</b> allow flexible scheduling of workload pods based on pod labels. For details about the supported scheduling policies, see <a href="#">Table 5-22</a>. Select a proper policy type and add a policy. For details about the parameters, see <a href="#">Table 5-23</a>.</li> </ul>	Custom policies

**Step 4** Select a proper load affinity rule and click  to add a scheduling policy. In this example, a scheduling policy is added in the **Required** area under **Workload Affinity**. This policy specifies that the newly created workload can only be scheduled to a node if a workload with specific labels is already running on that node.

**Table 5-22** Load affinity policies

Policy	Rule Type	Description	Example
Workload affinity	Required	<p>Hard constraint, which corresponds to <b>requiredDuringSchedulingIgnoredDuringExecution</b> in YAML for specifying the conditions that must be met.</p> <p>Select pods that require affinity by label. If such pods already run on a node in the topology key, the scheduler will <b>forcibly</b> schedule the created pods to that topology key.</p> <p><b>NOTE</b> If multiple affinity rules are configured, multiple labels will be used to filter pods that require affinity, and the newly created pods must be affinity with all pods that meet the label filtering conditions. In this way, all pods that meet the label filtering conditions locate in the same topology key for scheduling.</p>	Required

Policy	Rule Type	Description	Example
	Preferred	<p>Soft constraint, which corresponds to <b>preferredDuringSchedulingIgnoredDuringExecution</b> in YAML for specifying the conditions that need to be met as much as possible.</p> <p>Select pods that require affinity by label. If such pods already run on a node in the topology key, the scheduler will <b>preferentially</b> schedule the created pods to that topology key.</p> <p><b>NOTE</b> If multiple affinity rules are configured, multiple labels will be used to filter pods that require affinity, and the newly created pods will be preferentially to be affinity with multiple pods that meet the label filtering conditions. However, even if no pod meets the label filter conditions, a topology key will be selected for scheduling.</p>	
Workload anti-affinity	Required	<p>Hard constraint, which corresponds to <b>requiredDuringSchedulingIgnoredDuringExecution</b> in YAML for specifying the conditions that must be met.</p> <p>Select one or more pods that require anti-affinity by label. If such pods already run on a node in the topology key, the scheduler will <b>not</b> schedule the created pods to that topology key.</p> <p><b>NOTE</b> If multiple anti-affinity rules are configured, multiple labels will be used to filter pods that require anti-affinity, and the newly created pods must be anti-affinity with all pods that meet the label filtering conditions. In this way, all the topology keys where the pods that meet the label filtering conditions locate will not be scheduled.</p>	None

Policy	Rule Type	Description	Example
	Preferred	<p>Soft constraint, which corresponds to <b>preferredDuringSchedulingIgnoredDuringExecution</b> in YAML for specifying the conditions that need to be met as much as possible.</p> <p>Select one or more pods that require anti-affinity by label. If such pods already run on a node in the topology key, the scheduler will <b>preferentially</b> schedule the created pods to other topology keys.</p> <p><b>NOTE</b> If multiple anti-affinity rules are configured, multiple labels will be used to filter pods that require anti-affinity, and the newly created pods will be preferentially to be anti-affinity with multiple pods that meet the label filtering conditions. However, even if all topology keys involve the pods that require anti-affinity, a topology key will be selected for scheduling.</p>	

**Step 5** In the dialog box that is displayed on the right, click **Add Policy** to configure rules for filtering node labels.

**Table 5-23** Parameters for configuring load affinity/anti-affinity scheduling policies

Parameter	Description	Example
Weight	This parameter is available only in a <b>preferred</b> scheduling policy. Weights range from 1 to 100 and are taken into account as an extra scoring factor during scheduling. The scheduler combines the weight with other priority functions of the node to determine the final score and then assigns pods to the node with the highest total score.	None
Namespace	Namespace for which the scheduling policy takes effect.	default

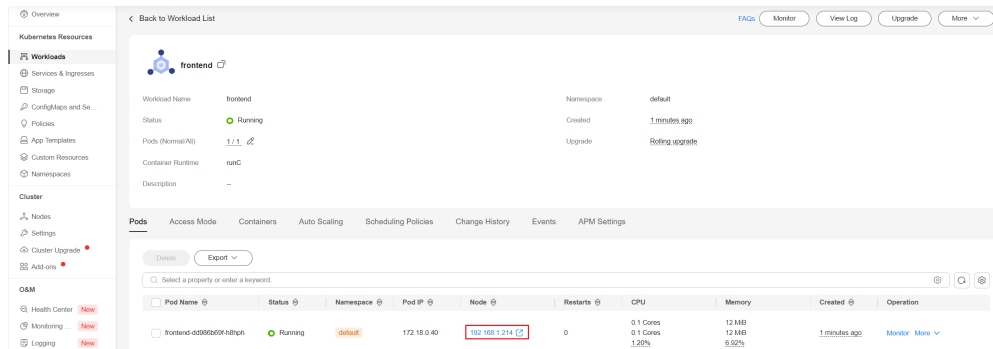
Parameter	Description	Example
Topology Key	<p>A topology key (<b>topologyKey</b>) determines the range of nodes to be scheduled based on node labels, identifies affinity/anti-affinity objects based on the labels and operators, and performs scheduling based on the topology key where the target object is located.</p> <ul style="list-style-type: none"> <li>For example, if the node label is <b>kubernetes.io/hostname</b>, the label value will be a node name. Nodes with different names are assigned to different topology keys. This allows for workload affinity scheduling on a single node, as each topology key contains only one node.</li> <li>If the specified label is <b>kubernetes.io/os</b>, the label value will be a node OS. Nodes running different OSs are assigned to different topology keys. This allows for workload affinity scheduling on multiple nodes, as each topology key contains multiple nodes. For example, if pods that meet the load affinity rule are running on a node in a topology key, all nodes in the topology key can be scheduled.</li> </ul>	kubernete s.io/ hostname
Label Key	<p>When configuring a workload affinity or anti-affinity policy, enter the workload label to be matched.</p> <p>Both default labels and custom labels are supported.</p>	app
Operator	<p>The following operators are supported:</p> <ul style="list-style-type: none"> <li><b>In</b>: The label of the affinity or anti-affinity object is in the label value list (<b>values</b> field).</li> <li><b>NotIn</b>: The label of the affinity or anti-affinity object is not in the label value list (<b>values</b> field).</li> <li><b>Exists</b>: The affinity or anti-affinity object has a specified label key.</li> <li><b>DoesNotExist</b>: The affinity or anti-affinity object does not have a specified label key.</li> </ul>	In
Label Value	<p>When configuring a workload affinity or anti-affinity policy, enter the value of the workload label.</p>	backend

**Step 6** After the scheduling policy is added, click **Create Workload**.

**Step 7** Verify that all pods run on the target node.



1. In the navigation pane, choose **Workloads**.
2. Click the workload name to enter its details page. On the **Pods** tab page, verify that the new pod and the existing backend pod run on the same node.



----End

## Using YAML

- Workload affinity

Kubernetes supports affinity between pods, which allows the frontend and backend pods of an application to be deployed together to minimize access latency.

Assume that the backend pods of an application have been created with label **app=backend**. You can use **.spec.affinity.podAffinity** to configure workload affinity so that the frontend pods (labeled **app=frontend**) and backend pods (labeled **app=backend**) can be deployed together.

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: frontend
  labels:
    app: frontend
spec:
  selector:
    matchLabels:
      app: frontend
  replicas: 3
  template:
    metadata:
      labels:
        app: frontend
    spec:
      containers:
        - image: nginx:alpine
          name: frontend
          resources:
            requests:
              cpu: 100m
              memory: 200Mi
            limits:
              cpu: 100m
              memory: 200Mi
      imagePullSecrets:
        - name: default-secret
      affinity: # Configure a scheduling policy.
        podAffinity: # Workload affinity scheduling rule
          requiredDuringSchedulingIgnoredDuringExecution: # Scheduling policy that must be met
            - topologyKey: prefer # Topology keys are divided based on node labels, among which prefer
              is a custom label.
```

```

labelSelector: # Select workloads that meet the requirements based on workload labels.
matchExpressions: # Workload label matching rule
- key: app # The key of the workload label is app.
operator: In # The rule is met if a value exists in the value list.
values: # Workload label values
- backend
preferredDuringSchedulingIgnoredDuringExecution: # Scheduling policy that is met as much
as possible
- weight: 100 # Priority that can be configured when the best-effort policy is used. The value
ranges from 1 to 100. A larger value indicates a higher priority.
podAffinityTerm: # Affinity configuration when the best-effort policy is used
topologyKey: topology.kubernetes.io/zone # Topology keys are divided based on node
labels by node AZ.
labelSelector:
matchExpressions:
- key: app
operator: In
values:
- backend

```

During workload scheduling in the preceding example, node topology keys are divided based on the **prefer** label using the rule that must be met. If backend pods (labeled **app=backend**) are running on a node in the topology key, frontend pods (labeled **app=frontend**) will also be deployed in that topology key, even if not all nodes in the topology key are running the backend pods. According to the best-effort rule, topology keys are divided based on **topology.kubernetes.io/zone** by node AZ. This ensures that the frontend and backend pods are deployed on nodes within the same AZ as much as possible.

 **NOTE**

For workload affinity, **topologyKey** cannot be left blank when **requiredDuringSchedulingIgnoredDuringExecution** and **preferredDuringSchedulingIgnoredDuringExecution** are used.

**topologyKey** is used to divide topology keys based on the labels of nodes. Nodes with the same labels are grouped into the same topology key. The scheduler then selects the topology key to be scheduled based on the workload label. A topology key can consist of multiple nodes. If a workload that meets a label selection rule runs on a node in a topology key, all nodes in the topology key can be scheduled.

For example, if the **topologyKey** label is set to **topology.kubernetes.io/zone**, nodes' AZs will be used as the topology keys, and workloads will be scheduled by AZ during deployment.

- **Workload anti-affinity**

In some cases, pods need to be deployed separately. This is because deploying them together can negatively impact performance.

Assume that the frontend pods of an application have been created with label **app=frontend**. To ensure that pods are deployed on different nodes and multiple AZs are preferred, you can use **.spec.affinity.podAntiAffinity** to configure workload anti-affinity.

```

apiVersion: apps/v1
kind: Deployment
metadata:
name: frontend
labels:
app: frontend
spec:
selector:
matchLabels:
app: frontend
replicas: 5
template:

```

```
metadata:
  labels:
    app: frontend
spec:
  containers:
  - image: nginx:alpine
    name: frontend
    resources:
      requests:
        cpu: 100m
        memory: 200Mi
      limits:
        cpu: 100m
        memory: 200Mi
  imagePullSecrets:
  - name: default-secret
  affinity:
    podAntiAffinity: # Workload anti-affinity scheduling rule
      requiredDuringSchedulingIgnoredDuringExecution: # Scheduling policy that must be met
        - topologyKey: kubernetes.io/hostname # Topology keys are divided based on node labels.
          labelSelector: # Pod label matching rule
            matchExpressions: # The key of the workload label is app.
              - key: app # The key of the workload label is app.
                operator: In # The rule is met if a value exists in the value list.
                values: # Workload label values
                  - frontend
            preferredDuringSchedulingIgnoredDuringExecution: # Scheduling policy that is met as much
as possible
              - weight: 100 # Priority that can be configured when the best-effort policy is used. The value
ranges from 1 to 100. A larger value indicates a higher priority.
                podAffinityTerm: # Affinity configuration when the best-effort policy is used
                  topologyKey: topology.kubernetes.io/zone # Topology keys are divided based on node
labels.
                    labelSelector:
                      matchExpressions:
                        - key: app
                          operator: In
                          values:
                            - frontend
```

In the preceding example, anti-affinity rules are configured. The rule that must be met indicates that node topology keys are divided based on **kubernetes.io/hostname**. Nodes with the **kubernetes.io/hostname** label have different label values. Therefore, there is only one node in each topology key. If a topology key contains only one node where a **frontend** pod already exists, pods with the same label will not be scheduled to that topology key. According to the best-effort rule, topology keys are divided based on **topology.kubernetes.io/zone** by node AZ. This ensures that the pods are deployed on nodes in different AZs as much as possible.

#### NOTE

For workload anti-affinity, when **requiredDuringSchedulingIgnoredDuringExecution** is used, the default access controller **LimitPodHardAntiAffinityTopology** of Kubernetes requires that **topologyKey** can only be **kubernetes.io/hostname**. To use other custom topology logic, modify or disable the access controller.

## 5.5 Logging In to a Container

### Scenario

If you encounter unexpected problems when using a container, you can log in to the container to debug it.

## Notes and Constraints

When using CloudShell to access a CCE cluster or container, you can open a maximum of 15 instances simultaneously.

## Using kubectl

**Step 1** Use kubectl to access the cluster. For details, see [Connecting to a Cluster Using kubectl](#).

**Step 2** Run the following command to view the created pod:

```
kubectl get pod
```

The example output is as follows:

NAME	READY	STATUS	RESTARTS	AGE
nginx-59d89cb66f-mhljr	1/1	Running	0	11m

**Step 3** Query the container name in the pod.

```
kubectl get po nginx-59d89cb66f-mhljr -o jsonpath='{range .spec.containers[*]}{.name}{end}{"\n"}'
```

The example output is as follows:

```
container-1
```

**Step 4** Run the following command to log in to the **container-1** container in the **nginx-59d89cb66f-mhljr** pod:

```
kubectl exec -it nginx-59d89cb66f-mhljr -c container-1 -- /bin/sh
```

**Step 5** To exit the container, run the **exit** command.

```
----End
```

## 5.6 Managing Workloads

### Scenario

After a workload is created, you can upgrade, log, monitor, roll back, or delete the workload, as well as edit its YAML file.

**Table 5-24** Workload/Job management

Operation	Description
<a href="#">Monitor</a>	You can view the CPU and memory usage of workloads and pods on the CCE console.
<a href="#">View Log</a>	You can view the logs of workloads.
<a href="#">Upgrade</a>	You can replace images or image tags to quickly upgrade Deployments, StatefulSets, and DaemonSets without interrupting services.

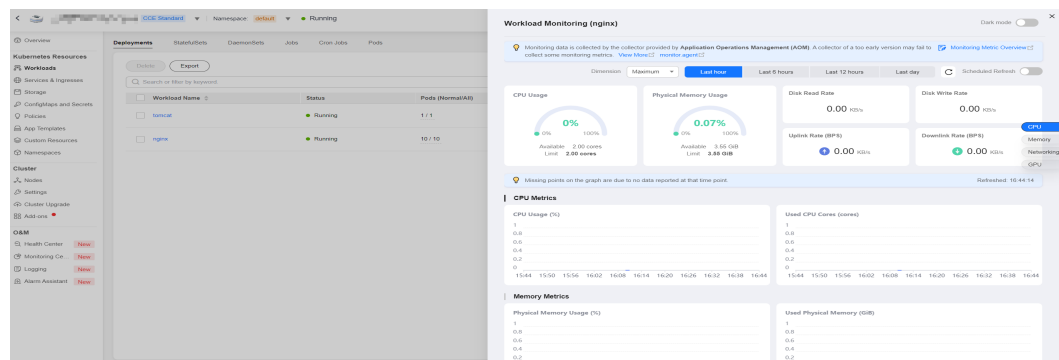
Operation	Description
<b>Edit YAML</b>	You can modify and download YAML files of Deployments, StatefulSets, DaemonSets, CronJobs, and pods on the CCE console. YAML files of jobs can only be viewed, copied, and downloaded.  <b>NOTE</b> If an existing CronJob is modified, the new configuration takes effect for the new pods, and the existing pods continue to run without any change.
<b>Roll Back</b>	Only Deployments can be rolled back.
<b>Redeploy</b>	You can redeploy a workload. After the workload is redeployed, all pods in the workload will be restarted.
<b>Enabling/ Disabling the Upgrade</b>	Only Deployments support this operation.
<b>Manage Label</b>	Labels are attached to workloads as key-value pairs to manage and select workloads. Jobs and Cron Jobs do not support this operation.
<b>Delete</b>	You can delete a workload or job that is no longer needed. Deleted workloads or jobs cannot be recovered.
<b>View Events</b>	You can view event names, event types, number of occurrences, Kubernetes events, first occurrence time, and last occurrence time.
<b>Stop/Start</b>	You can only start or stop a cron job.

## Monitoring a Workload

You can view the CPU and memory usage of Deployments and pods on the CCE console to determine the resource specifications you may need. This section uses a Deployment as an example to describe how to monitor a workload.

- Step 1** Log in to the CCE console, go to an existing cluster, and choose **Workloads** in the navigation pane.
- Step 2** Click the **Deployments** tab and click **Monitor** of the target workload. On the page that is displayed, you can view CPU usage and memory usage of the workload.

Figure 5-10 Viewing monitoring information



**Step 3** Click the workload name. On the **Pods** tab page, click the **Monitor** of the target pod to view its CPU and memory usage.

----End

## Viewing Logs

You can view logs of Deployments, StatefulSets, DaemonSets, and jobs. This section uses a Deployment as an example to describe how to view logs.

### NOTICE

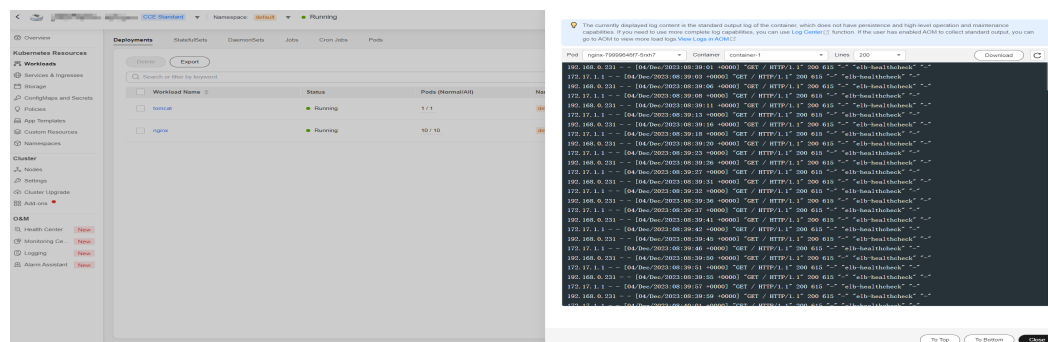
Before viewing logs, ensure that the time of the browser is the same as that on the backend server.

**Step 1** Log in to the CCE console, go to an existing cluster, and choose **Workloads** in the navigation pane.

**Step 2** Click the **Deployments** tab and click **View Log** of the target workload.

In the displayed **View Log** window, you can view logs.

Figure 5-11 Viewing logs of a workload



 **NOTE**

The displayed logs are standard output logs of containers and do not have persistence and advanced O&M capabilities. To use more comprehensive log capabilities, see [Logs](#). If the function of collecting standard output is enabled for the workload (enabled by default), you can go to AOM to view more workload logs. For details, see [Collecting Container Logs Using ICAgent](#).

----End

## Upgrading a Workload

You quickly upgrade Deployments, StatefulSets, and DaemonSets on the CCE console.

This section uses a Deployment as an example to describe how to upgrade a workload.

Before replacing an image or image version, upload the new image to the SWR service.

**Step 1** Log in to the CCE console, go to an existing cluster, and choose **Workloads** in the navigation pane.

**Step 2** Click the **Deployments** tab and click **Upgrade** of the target workload.

 **NOTE**

- Workloads cannot be upgraded in batches.
- Before performing an in-place StatefulSet upgrade, you must manually delete old pods. Otherwise, the upgrade status is always displayed as **Processing**.

**Step 3** Upgrade the workload based on service requirements. The method for setting parameter is the same as that for creating a workload.

**Step 4** After the update is complete, click **Upgrade Workload**, manually confirm the YAML file, and submit the upgrade.

----End

## Editing a YAML file

You can modify and download YAML files of Deployments, StatefulSets, DaemonSets, CronJobs, and pods on the CCE console. YAML files of jobs can only be viewed, copied, and downloaded. This section uses a Deployment as an example to describe how to edit the YAML file.

**Step 1** Log in to the CCE console, go to an existing cluster, and choose **Workloads** in the navigation pane.

**Step 2** Click the **Deployments** tab and choose **More > Edit YAML** in the **Operation** column of the target workload. In the dialog box that is displayed, modify the YAML file.

**Step 3** Click **OK**.

**Step 4** (Optional) In the **Edit YAML** window, click **Download** to download the YAML file.

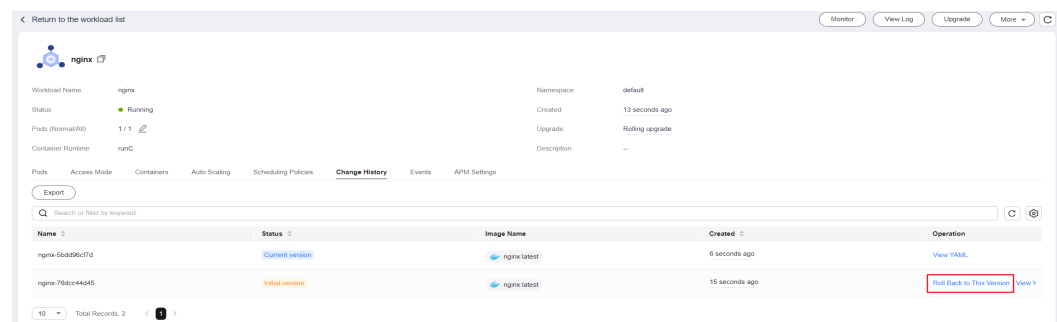
----End

## Rolling Back a Workload (Available Only for Deployments)

CCE records the release history of all Deployments. You can roll back a Deployment to a specified version.

- Step 1** Log in to the CCE console, go to an existing cluster, and choose **Workloads** in the navigation pane.
- Step 2** Click the **Deployments** tab and choose **More > Roll Back** in the **Operation** column of the target workload.
- Step 3** Switch to the **Change History** tab page, click **Roll Back to This Version** of the target version, manually confirm the YAML file, and click **OK**.

**Figure 5-12** Rolling back a workload version



----End

## Redeploying a Workload

After you redeploy a workload, all pods in the workload will be restarted. This section uses Deployments as an example to illustrate how to redeploy a workload.

- Step 1** Log in to the CCE console, go to an existing cluster, and choose **Workloads** in the navigation pane.
- Step 2** Click the **Deployments** tab and choose **More > Redeploy** in the **Operation** column of the target workload.
- Step 3** In the dialog box that is displayed, click **Yes** to redeploy the workload.

----End

## Disabling/Enabling Upgrade (Available Only for Deployments)

Only Deployments support this operation.

- After the upgrade is disabled, the upgrade command can be delivered but will not be applied to the pods.  
If you are performing a rolling upgrade, the rolling upgrade stops after the disabling upgrade command is delivered. In this case, the new and old pods co-exist.
- After the upgrade is enabled, a Deployment can be upgraded or rolled back. Its pods will inherit the latest updates of the Deployment. If they are inconsistent, the pods will be upgraded automatically according to the latest information of the Deployment.



**NOTICE**

Deployments in the disable upgrade state cannot be rolled back.

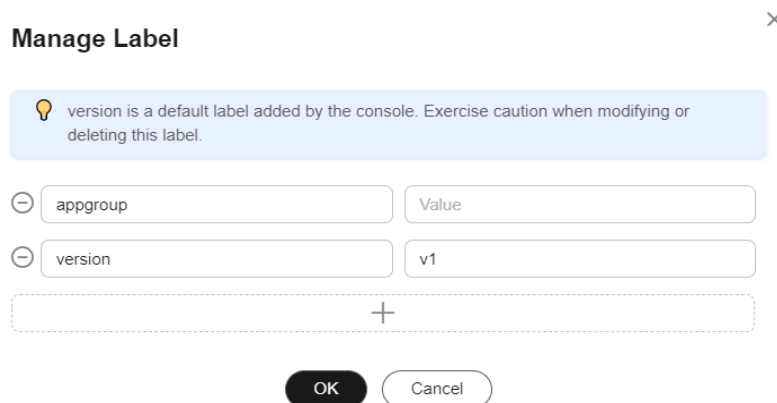
- Step 1** Log in to the CCE console, go to an existing cluster, and choose **Workloads** in the navigation pane.
  - Step 2** Click the **Deployments** tab and choose **More > Disable/Enable Upgrade** in the **Operation** column of the workload.
  - Step 3** In the dialog box that is displayed, click **Yes**.
- End

## Managing Labels

Labels are key-value pairs and can be attached to workloads. You can manage and select workloads by labels. You can add labels to multiple workloads or a specified workload.

- Step 1** Log in to the CCE console, go to an existing cluster, and choose **Workloads** in the navigation pane.
- Step 2** Click the **Deployments** tab and choose **More > Manage Label** in the **Operation** column of the target workload.
- Step 3** Click **+**, enter a key and a value, and click **OK**.

**Figure 5-13** Managing labels



**NOTE**

A key-value pair must contain 1 to 63 characters starting and ending with a letter or digit. Only letters, digits, hyphens (-), underscores (\_), and periods (.) are allowed.

----End

## Deleting a Workload/Job

You can delete a workload or job that is no longer needed. Deleted workloads or jobs cannot be recovered. This section uses a Deployment as an example to describe how to delete a workload.

**Step 1** Log in to the CCE console, go to an existing cluster, and choose **Workloads** in the navigation pane.

**Step 2** In the same row as the workload you will delete, choose **Operation > More > Delete**.

Read the system prompts carefully. A workload cannot be recovered after it is deleted. Exercise caution when performing this operation.

**Step 3** Click **Yes**.

### NOTE

- If the node where the pod is located is unavailable or shut down and the workload cannot be deleted, you can forcibly delete the pod from the pod list on the workload details page.
- Ensure that the storage volumes to be deleted are not used by other workloads. If these volumes are imported or have snapshots, you can only unbind them.

----End

## Events

This section uses a Deployment as an example to describe how to view events of a workload. To view the event of a job or CronJob, click **View Event** in the **Operation** column of the target workload.

**Step 1** Log in to the CCE console, go to an existing cluster, and choose **Workloads** in the navigation pane.

**Step 2** On the **Deployments** tab page, click the target workload. In the **Pods** tab page, click the **View Events** to view the event name, event type, number of occurrences, Kubernetes event, first occurrence time, and last occurrence time.

### NOTE

Event data will be retained for one hour and then automatically deleted.

----End

## 5.7 Pod Security

### 5.7.1 Configuring a Pod Security Policy

A pod security policy (PSP) is a cluster-level resource that controls sensitive security aspects of the pod specification. The **PodSecurityPolicy** object in Kubernetes defines a group of conditions that a pod must comply with to be accepted by the system, as well as the default values of related fields.

By default, the PSP access control component is enabled for clusters of v1.17.17 and a global default PSP named **psp-global** is created. You can modify the default

policy (but not delete it). You can also create a PSP and bind it to the RBAC configuration.

 **NOTE**

- In addition to the global default PSP, the system configures independent PSPs for system components in namespace kube-system. Modifying the psp-global configuration does not affect pod creation in namespace kube-system.
- PodSecurityPolicy was deprecated in Kubernetes v1.21, and removed from Kubernetes in v1.25. You can use pod security admission as a substitute for PodSecurityPolicy. For details, see [Configuring Pod Security Admission](#).

## Modifying the Global Default PSP

Before modifying the global default PSP, ensure that a CCE cluster has been created and connected by using kubectl.

**Step 1** Run the following command:

```
kubectl edit psp psp-global
```

**Step 2** Modify the required parameters, as shown in [Table 5-25](#).

**Table 5-25** PSP configuration

Item	Description
privileged	Starts the privileged container.
hostPID hostIPC	Uses the host namespace.
hostNetwork hostPorts	Uses the host network and port.
volumes	Specifies the type of the mounted volume that can be used.
allowedHostPaths	Specifies the host path to which a hostPath volume can be mounted. The <b>pathPrefix</b> field specifies the host path prefix group to which a hostPath volume can be mounted.
allowedFlexVolumes	Specifies the FlexVolume driver that can be used.
fsGroup	Configures the supplemental group ID used by the mounted volume in the pod.
readOnlyRootFilesystem	Pods can only be started using a read-only root file system.
runAsUser runAsGroup supplementalGroups	Specifies the user ID, primary group ID, and supplemental group ID for starting containers in a pod.

Item	Description
allowPrivilegeEscalation defaultAllowPrivilegeEscalation	Specifies whether <b>allowPrivilegeEscalation</b> can be set to <b>true</b> in a pod. This configuration controls the use of Setuid and whether programs can use additional privileged system calls.
defaultAddCapabilities requiredDropCapabilities allowedCapabilities	Controls the Linux capabilities used in pods.
seLinux	Controls the configuration of seLinux used in pods.
allowedProcMountTypes	Controls the ProcMountTypes that can be used by pods.
annotations	Configures AppArmor or Seccomp used by containers in a pod.
forbiddenSysctls allowedUnsafeSysctls	Controls the configuration of Sysctl used by containers in a pod.

----End

## Example of Enabling Unsafe Sysctls in Pod Security Policy

You can configure allowed-unsafe-sysctls for a node pool. For CCE clusters of **v1.17.17** and later versions, add configurations in **allowedUnsafeSysctls** of the pod security policy to make the configuration take effect. For details, see [Table 5-25](#).

In addition to modifying the global pod security policy, you can add new pod security policies. For example, enable the **net.core.somaxconn** unsafe sysctls. The following is an example of adding a pod security policy:

```
apiVersion: policy/v1beta1
kind: PodSecurityPolicy
metadata:
  annotations:
    seccomp.security.alpha.kubernetes.io/allowedProfileNames: '*'
  name: sysctl-ppsp
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:
  - '*'
---
kind: ClusterRole
apiVersion: rbac.authorization.k8s.io/v1
metadata:
  name: sysctl-ppsp
rules:
- apiGroups:
  - '*'
  resources:
  - podsecuritypolicies
  resourceNames:
  - sysctl-ppsp
  verbs:
  - use
---
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  name: sysctl-ppsp
roleRef:
  kind: ClusterRole
  name: sysctl-ppsp
  apiGroup: rbac.authorization.k8s.io
subjects:
- kind: Group
  name: system:authenticated
  apiGroup: rbac.authorization.k8s.io

```

## Restoring the Original PSP

If you have modified the default pod security policy and want to restore the original pod security policy, perform the following operations.

- Step 1** Create a policy description file named **policy.yaml**. **policy.yaml** is an example file name. You can rename it as required.

### vi policy.yaml

The content of the description file is as follows:

```

apiVersion: policy/v1beta1
kind: PodSecurityPolicy
metadata:
  name: psp-global
  annotations:
    seccomp.security.alpha.kubernetes.io/allowedProfileNames: '*'
spec:
  privileged: true
  allowPrivilegeEscalation: true
  allowedCapabilities:
  - '*'
  volumes:
  - '*'
  hostNetwork: true
  hostPorts:
  - min: 0
    max: 65535
  hostIPC: true
  hostPID: true
  runAsUser:

```

```

    rule: 'RunAsAny'
  seLinux:
    rule: 'RunAsAny'
  supplementalGroups:
    rule: 'RunAsAny'
  fsGroup:
    rule: 'RunAsAny'
---
kind: ClusterRole
apiVersion: rbac.authorization.k8s.io/v1
metadata:
  name: psp-global
rules:
- apiGroups:
  - "*"
  resources:
  - podsecuritypolicies
  resourceNameNames:
  - psp-global
  verbs:
  - use
---
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  name: psp-global
roleRef:
  kind: ClusterRole
  name: psp-global
  apiGroup: rbac.authorization.k8s.io
subjects:
- kind: Group
  name: system:authenticated
  apiGroup: rbac.authorization.k8s.io

```

**Step 2** Run the following command:

```
kubectl apply -f policy.yaml
```

```
----End
```

## 5.7.2 Configuring Pod Security Admission

Before using [pod security admission](#), understand Kubernetes [Pod Security Standards](#). These standards define different isolation levels for pods. They let you define how you want to restrict the behavior of pods in a clear, consistent fashion. Kubernetes offers a built-in pod security admission controller to enforce the pod security standards. Pod security restrictions are applied at the namespace level when pods are created.

The pod security standard defines three security policy levels:

**Table 5-26** Pod security policy levels

Level	Description
privileged	Unrestricted policy, providing the widest possible level of permissions, typically aimed at system- and infrastructure-level workloads managed by privileged, trusted users, such as CNIs and storage drivers.

Level	Description
baseline	Minimally restrictive policy that prevents known privilege escalations, typically targeted at non-critical workloads. This policy disables capabilities such as hostNetwork and hostPID.
restricted	Heavily restricted policy, following current Pod hardening best practices.

**Pod security admission** is applied at the namespace level. The controller restricts the security context and other parameters in the pod or container in the namespace. The privileged policy does not verify the **securityContext** field of the pod and container. The baseline and restricted policies have different requirements on **securityContext**. For details, see [Pod Security Standards](#).

Setting security context: [Configure a Security Context for a Pod or Container](#)

## Pod Security Admission Labels

Kubernetes defines three types of labels for pod security admission (see [Table 5-27](#)). You can set these labels in a namespace to define the pod security standard level to be used. However, do not change the pod security standard level in system namespaces such as kube-system. Otherwise, pods in the system namespace may be faulty.

**Table 5-27** Pod security admission labels

Mode	Target Object	Description
enforce	Pods	Policy violations will cause the pod to be rejected.
audit	Workloads (such as Deployment and job)	Policy violations will trigger the addition of an audit annotation to the event recorded in the audit log, but are otherwise allowed.
warn	Workloads (such as Deployment and job)	Policy violations will trigger a user-facing warning, but are otherwise allowed.

### NOTE

Pods are often created indirectly, by creating a workload object such as a Deployment or job. To help catch violations early, both the audit and warning modes are applied to the workload resources. However, the enforce mode is applied only to the resulting pod objects.

## Enforcing Pod Security Admission with Namespace Labels

You can label namespaces to enforce pod security standards. Assume that a namespace is configured as follows:

```
apiVersion: v1
kind: Namespace
metadata:
  name: my-baseline-namespace
  labels:
    pod-security.kubernetes.io/enforce: privileged
    pod-security.kubernetes.io/enforce-version: v1.25
    pod-security.kubernetes.io/audit: baseline
    pod-security.kubernetes.io/audit-version: v1.25
    pod-security.kubernetes.io/warn: restricted
    pod-security.kubernetes.io/warn-version: v1.25

# The label can be in either of the following formats:
# pod-security.kubernetes.io/<MODE>: <LEVEL>
# pod-security.kubernetes.io/<MODE>-version: <VERSION>
# The audit and warn modes inform you of which security behaviors are violated by the load.
```

Namespace labels indicate which policy level to apply for the mode. For each mode, there are two labels that determine the policy used:

- `pod-security.kubernetes.io/<MODE>: <LEVEL>`
  - `<MODE>`: must be **enforce**, **audit**, or **warn**. For details about the modes, see [Table 5-27](#).
  - `<LEVEL>`: must be **privileged**, **baseline**, or **restricted**. For details about the levels, see [Table 5-26](#).
- `pod-security.kubernetes.io/<MODE>-version: <VERSION>`

Optional, which pins the policy to a given Kubernetes version.

  - `<MODE>`: must be **enforce**, **audit**, or **warn**. For details about the modes, see [Table 5-27](#).
  - `<VERSION>`: Kubernetes version number. For example, v1.25. You can also use **latest**.

If pods are deployed in the preceding namespace, the following security restrictions apply:

1. The verification in the enforce mode is skipped (enforce mode + privileged level).
2. Restrictions related to the baseline policy are verified (audit mode + baseline level). That is, if the pod or container violates the policy, the corresponding event is recorded into the audit log.
3. Restrictions related to the restricted policy are verified (warn mode + restricted level). That is, if the pod or container violates the policy, the user will receive an alarm when creating the pod.

## Migrating from Pod Security Policy to Pod Security Admission

If you use pod security policies in a cluster earlier than v1.25 and need to replace them with pod security admission in a cluster of v1.25 or later, follow the guide in [Migrate from PodSecurityPolicy to the Built-In PodSecurity Admission Controller](#).



---

**NOTICE**

1. Pod security admission supports only three isolation modes, less flexible than pod security policies. If you require more control over specific constraints, you will need to use a Validating Admission Webhook to enforce those policies.
  2. Pod security admission is a non-mutating admission controller, meaning it will not modify pods before validating them. If you were relying on this aspect of PSP, you will need to either modify the security context in your workloads, or use a Mutating Admission Webhook to make those changes.
  3. PSP lets you bind different policies to different service accounts. This approach has many pitfalls and is not recommended, but if you require this feature anyway you will need to use a third-party webhook instead.
  4. Do not apply pod security admission to namespaces where CCE components, such as kube-system, kube-public, and kube-node-lease, are deployed. Otherwise, CCE components and add-on functions will be abnormal.
- 

## Documentation

- [Pod Security Admission](#)
- [Mapping PodSecurityPolicies to Pod Security Standards](#)
- [Enforce Pod Security Standards with Namespace Labels](#)
- [Enforce Pod Security Standards by Configuring the Built-in Admission Controller](#)

# 6 Scheduling

## 6.1 Overview

CCE supports different types of resource scheduling and task scheduling, improving application performance and overall cluster resource utilization. This section describes the main functions of CPU resource scheduling, GPU/NPU heterogeneous resource scheduling, and Volcano scheduling.

### CPU Scheduling

CCE provides CPU policies to allocate complete physical CPU cores to applications, improving application performance and reducing application scheduling latency.

Function	Description	Documentation
CPU policy	When many CPU-intensive pods are running on a node, workloads may be migrated to different CPU cores. Many workloads are not sensitive to this migration and work fine without any intervention. For CPU-sensitive applications, you can use the CPU policy provided by Kubernetes to allocate dedicated cores to applications, improving application performance and reducing application scheduling latency.	<a href="#">CPU Policy</a>
Enhanced CPU policy	Based on the Kubernetes static core binding policy, the enhanced CPU policy (enhanced-static) supports burstable pods (whose CPU requests and limits must be positive integers) and allows them to preferentially use certain CPUs to ensure application stability.	<a href="#">Enhanced CPU Policy</a>

## GPU Scheduling

CCE schedules heterogeneous GPU resources in clusters and allows GPUs to be used in containers.

Function	Description	Documentation
Default GPU scheduling in Kubernetes	This function allows you to specify the number of GPUs that a pod requests. The value can be less than 1 so that multiple pods can share a GPU.	<a href="#">Default GPU Scheduling in Kubernetes</a>

## NPU Scheduling

CCE schedules heterogeneous NPU resources in a cluster to quickly and efficiently perform inference and image recognition.

Function	Description	Documentation
NPU scheduling	NPU scheduling allows you to specify the number of NPUs that a pod requests to provide NPU resources for workloads.	<a href="#">NPU Scheduling</a>

## Volcano Scheduling

Volcano is a Kubernetes-based batch processing platform that supports machine learning, deep learning, bioinformatics, genomics, and other big data applications. It provides general-purpose, high-performance computing capabilities, such as job scheduling, heterogeneous chip management, and job running management.

Function	Description	Documentation
NUMA affinity scheduling	Volcano targets to lift the limitation to make scheduler NUMA topology aware so that: <ul style="list-style-type: none"> <li>Pods are not scheduled to the nodes that NUMA topology does not match.</li> <li>Pods are scheduled to the most suitable node for NUMA topology.</li> </ul>	<a href="#">NUMA Affinity Scheduling</a>

## Cloud Native Hybrid Deployment

The cloud native hybrid deployment solution focuses on the Volcano and Kubernetes ecosystems to help users improve resource utilization and efficiency and reduce costs.

Function	Description	Documentation
Dynamic resource oversubscription	Based on the types of online and offline jobs, Volcano scheduling is used to utilize the resources that are requested but not used in the cluster (the difference between the number of requested resources and the number of used resources) for resource oversubscription and hybrid deployment to improve cluster resource utilization.	<a href="#">Dynamic Resource Oversubscription</a>

## 6.2 CPU Scheduling

### 6.2.1 CPU Policy

#### Application Scenarios

By default, kubelet uses [CFS quotas](#) to enforce pod CPU limits. When a node runs many CPU-bound pods, the workload can move to different CPU cores depending on whether the pod is throttled and which CPU cores are available at scheduling time. Many workloads are not sensitive to this migration and thus work fine without any intervention. Some applications are CPU-sensitive. They are sensitive to:

- CPU throttling
- Context switching
- Processor cache misses
- Cross-socket memory access
- Hyperthreads that are expected to run on the same physical CPU card

If your workloads are sensitive to any of these items, you can use Kubernetes [CPU management policies](#) to allocate dedicated CPU cores (through CPU core binding) to these workloads. This will shorten scheduling latency and improve application performance. The CPU manager preferentially allocates resources on a socket and full physical cores to avoid interference.

A CPU management policy is specified by using kubelet `--cpu-manager-policy`. By default, Kubernetes supports the following policies:

- **none**: the default policy. The **none** policy explicitly enables the existing default CPU affinity scheme, providing no affinity beyond what the OS scheduler does automatically.
- **static**: The **static** policy allows containers in [guaranteed pods](#) with integer CPU requests to be use dedicated CPU resources on nodes through CPU core binding.

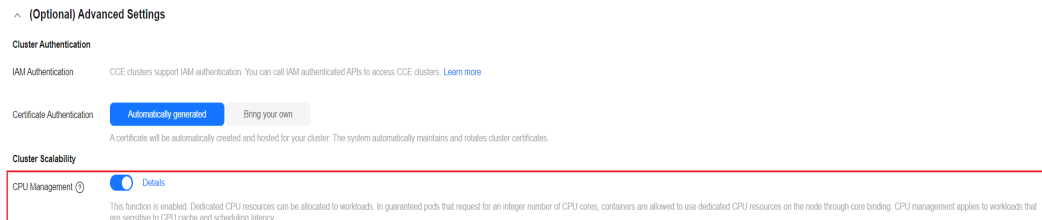
#### Notes and Constraints

The CPU management policy cannot take effect on physical cloud server nodes.

## Enabling CPU Management for the Default Node Pool

When creating a cluster, you can enable CPU management in **Advanced Settings**.

- If this function is enabled, the **static** Kubernetes policy is used, where CPU core binding is used.
- If this function is disabled, the **none** Kubernetes policy is used, where CPU core binding is not used.

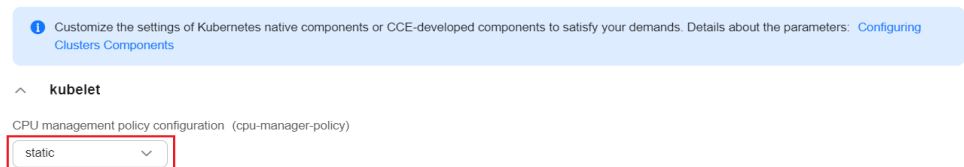


## Enabling CPU Management for a Custom Node Pool

You can configure a CPU management policy in a custom node pool. After the configuration, the kubelet parameter **--cpu-manager-policy** will be automatically modified on the node in the node pool.

1. Log in to the CCE console and click the cluster name to access the cluster console.
2. Choose **Nodes** in the navigation pane and click the **Node Pools** tab on the right. Locate the target node pool and choose **More > Manage**.
3. On the **Manage Configurations** page, change the **cpu-manager-policy** value to **static** in the **kubelet** area.

Manage Configurations (Node Pool XXXXXXXXXX.nodepool-52200)



4. Click **OK**.

## Allowing Pods to Exclusively Use the CPU Resources

Prerequisites:

- Enable the **static** policy on the node. For details, see [Enabling CPU Management for the Default Node Pool](#).
- Both requests and limits must be configured in pods and their values must be the same integer.
- If an init container needs to exclusively use CPUs, set its requests to the same as that of the service container. Otherwise, the service container does not inherit the CPU allocation result of the init container, and the CPU manager reserves more CPU resources than supposed. For more information, see [App Containers can't inherit Init Containers CPUs - CPU Manager Static Policy](#).

You can use **node affinity scheduling** to schedule the configured pods to the nodes where the **static** policy is enabled. In this way, the pods can exclusively use the CPU resources.

Example YAML:

```
kind: Deployment
apiVersion: apps/v1
metadata:
  name: test
spec:
  replicas: 1
  selector:
    matchLabels:
      app: test
  template:
    metadata:
      labels:
        app: test
    spec:
      containers:
        - name: container-1
          image: nginx:alpine
          resources:
            requests:
              cpu: 2          # The value must be an integer and must be the same as that in limits.
              memory: 2048Mi
            limits:
              cpu: 2          # The value must be an integer and must be the same as that in requests.
              memory: 2048Mi
          imagePullSecrets:
            - name: default-secret
```

## Verification

Take a node with 8 vCPUs and 16 GiB of memory as an example. Deploy a workload whose CPU request and limit are both **2** on the node beforehand.

Log in to the node where the workload is running and check the `/var/lib/kubelet/cpu_manager_state` output.

```
cat /var/lib/kubelet/cpu_manager_state
```

The command output is as follows:

```
{"policyName":"static","defaultCpuSet":"0-1,4-7","entries":{"de14506d-0408-411f-bbb9-822866b58ae2":{"container-1":"2-3"},"checksum":3744493798}}
```

- If the **policyName** is **static**, the policy has been configured.
- Value **2-3** indicates the set of CPUs that can be used by containers in the pod.

## 6.2.2 Enhanced CPU Policy

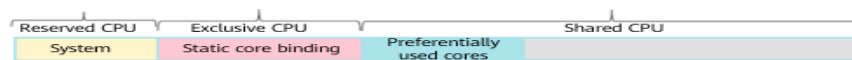
Kubernetes provides two **CPU policies**: none and static.

- **none**: The CPU policy is disabled by default, indicating the existing scheduling behavior.
- **static**: The static CPU core binding policy is enabled. This policy allows pods with certain resource characteristics to be granted enhanced CPU affinity and exclusivity on the node.

Based on the Kubernetes static core binding policy, the enhanced CPU policy (enhanced-static) supports burstable pods (whose CPU requests and limits must be positive integers) and allows them to preferentially use certain CPUs to ensure application stability. Example:

```
...
spec:
  containers:
  - name: nginx
    image: nginx
    resources:
      limits:
        memory: "300Mi"
        cpu: "2"
      requests:
        memory: "200Mi"
        cpu: "1"
```

This feature is built on the optimized CPU scheduling in the kernel. When the CPU usage preferentially used by a container exceeds 85%, the container is automatically allocated to other CPUs with low usage to ensure the response capability of applications.



**NOTE**

- When enhanced CPU policy is enabled, the application performance is better than that of the **none** policy but worse than that of the **static** policy.
- CPU would not be exclusively used by burstable pods, it is still in the shared CPU pool. When the burstable pods are in the low tide, other pods can share this CPU.

## Notes and Constraints

To use this feature, the following conditions must be met:

- The cluster version must be v1.23 or later.
- The node OS is .
- The CPU management policy cannot take effect on physical cloud server nodes.

## Procedure

- 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. In the right pane, click the **Node Pools** tab.
- Step 3** Select a node pool whose OS is and click **Manage** in the **Operation** column.
- Step 4** On the **Manage Components** page, change the **cpu-manager-policy** value to **enhanced-static** in the **kubelet** area.

**Figure 6-1** CPU policy



**Step 5** Click **OK**.

----End

## Verification

Take a node with 8 vCPUs and 32 GB memory as an example. Deploy a workload whose CPU request is 1 and limit is 2 in the cluster in advance.

**Step 1** Log in to a node in the node pool and view the `/var/lib/kubelet/cpu_manager_state` output.

```
cat /var/lib/kubelet/cpu_manager_state
```

Command output:

```
{"policyName":"enhanced-static","defaultCpuSet":"0,2-7","entries":{"6739f6f2-  
ebe5-48ae-945a-986d5d8919b9":{"container-1":"0-7,10001"},"checksum":1638128523}}
```

- If the value of **policyName** is **enhanced-static**, the policy is configured successfully.
- 10000 is used as the base for the CPU ID. In this example, 10001 indicates that the affinity CPU ID used by the container is CPU 1, and 0-7 indicates the set of CPUs that can be used by the container in the pod.

**Step 2** Check the cgroup setting of **cpuset.preferred\_cpus** of the container. The output is the ID of the CPU that is preferentially used.

```
cat /sys/fs/cgroup/cpuset/kubepods/burstable/pod {pod uid} /cpuset.preferred_cpus
```

- `{pod uid}` indicates the pod UID, which can be obtained by running the following command on the host that has been connected to the cluster using `kubectl`:

```
kubectl get po {pod name} -n {namespace} -ojsonpath='{.metadata.uid}'
```

In the preceding command, `{pod name}` and `{namespace}` indicate the pod name and the namespace to which the pod belongs.

- `{Container id}` must be a complete container ID. You can run the following command on the node where the container is running to obtain the container ID:

Docker node pool: In the command, `{pod name}` indicates the pod name.

```
docker ps --no-trunc | grep {pod name} | grep -v cce-pause | awk '{print $1}'
```



containerd node pool: In the command, *{pod name}* indicates the pod name, *{pod id}* indicates the pod ID, and *{container name}* indicates the container name.

```
# Obtain the pod ID.
crictl pods | grep {pod name} | awk '{print $1}'
# Obtain the complete container ID.
crictl ps --no-trunc | grep {pod id} | grep {container name} | awk '{print $1}'
```

A complete example is as follows:

```
cat /sys/fs/cgroup/cpuset/kubepods/burstable/pod6739f6f2-
ebe5-48ae-945a-986d5d8919b9/5ba5603434b95fd22d36fba6a5f1c44eba83c18c2e1de9b52ac9b52e93547a1
3/cpuset.preferred_cpus
```

If the following command output is displayed, CPU 1 is preferentially used.

```
1
```

----End

## 6.3 GPU Scheduling

### 6.3.1 Default GPU Scheduling in Kubernetes

You can use GPUs in CCE containers.

#### Prerequisites

- A GPU node has been created. For details, see [Creating a Node](#).
- The CCE AI Suite (NVIDIA GPU) add-on has been installed. During the installation, select the driver corresponding to the GPU model on the node. For details, see [CCE AI Suite \(NVIDIA GPU\)](#).
- When the default GPU scheduling is used in clusters of v1.27 or earlier, the CCE AI Suite (NVIDIA GPU) add-on mounts the driver directory to **/usr/local/nvidia/lib64**, so you need to add **/usr/local/nvidia/lib64** to the **LD\_LIBRARY\_PATH** environment variable to use GPUs in a container. You can skip this step for clusters of v1.28 or later.

You can add environment variables in any of the following ways:

- Configure the **LD\_LIBRARY\_PATH** environment variable in the Dockerfile used for creating an image. (Recommended)
- Configure the **LD\_LIBRARY\_PATH** environment variable in the image startup command.
- Define the **LD\_LIBRARY\_PATH** environment variable when creating a workload. (Ensure that this variable is not configured in the container. Otherwise, it will be overwritten.)

```
...
env:
  - name: LD_LIBRARY_PATH
    value: /usr/local/nvidia/lib64
...
```

## Using GPUs

Create a workload and request GPUs. You can specify the number of GPUs as follows:

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: gpu-test
  namespace: default
spec:
  replicas: 1
  selector:
    matchLabels:
      app: gpu-test
  template:
    metadata:
      labels:
        app: gpu-test
    spec:
      containers:
      - image: nginx:perl
        name: container-0
        resources:
          requests:
            cpu: 250m
            memory: 512Mi
            nvidia.com/gpu: 1 # Number of requested GPUs
          limits:
            cpu: 250m
            memory: 512Mi
            nvidia.com/gpu: 1 # Maximum number of GPUs that can be used
      imagePullSecrets:
      - name: default-secret
```

**nvidia.com/gpu** specifies the number of GPUs to be requested. The value can be smaller than **1**. For example, **nvidia.com/gpu: 0.5** indicates that multiple pods share a GPU. In this case, all the requested GPU resources come from the same GPU card.

### NOTE

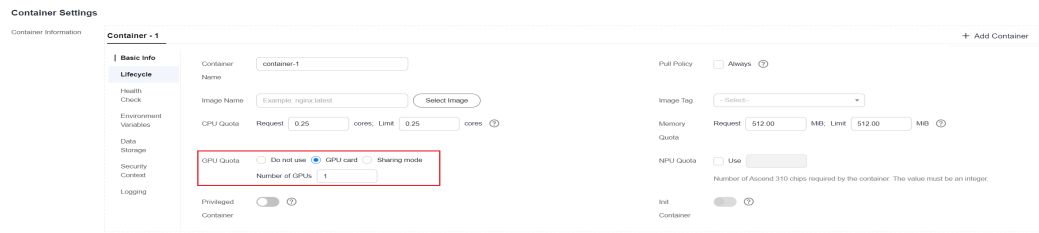
When you use **nvidia.com/gpu** to specify the number of GPUs, the values of requests and limits must be the same.

After **nvidia.com/gpu** is specified, workloads will not be scheduled to nodes without GPUs. If the node is GPU-starved, Kubernetes events similar to the following are reported:

- 0/2 nodes are available: 2 Insufficient nvidia.com/gpu.
- 0/4 nodes are available: 1 InsufficientResourceOnSingleGPU, 3 Insufficient nvidia.com/gpu.

To use GPU resources on the CCE console, you only need to configure the GPU quota when creating a workload.

Figure 6-2 GPU quota



## GPU Node Labels

CCE will label GPU-enabled nodes after they are created. Different types of GPU-enabled nodes have different labels.

```
$ kubectl get node -L accelerator
NAME          STATUS  ROLES  AGE   VERSION          ACCELERATOR
10.100.2.179 Ready  <none> 8m43s v1.19.10-r0-CCE21.11.1.B006-21.11.1.B006 nvidia-t4
```

When using GPUs, you can enable the affinity between pods and nodes based on labels so that the pods can be scheduled to the correct nodes.

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: gpu-test
  namespace: default
spec:
  replicas: 1
  selector:
    matchLabels:
      app: gpu-test
  template:
    metadata:
      labels:
        app: gpu-test
    spec:
      nodeSelector:
        accelerator: nvidia-t4
      containers:
      - image: nginx:perl
        name: container-0
        resources:
          requests:
            cpu: 250m
            memory: 512Mi
            nvidia.com/gpu: 1 # Number of requested GPUs
          limits:
            cpu: 250m
            memory: 512Mi
            nvidia.com/gpu: 1 # Maximum number of GPUs that can be used
        imagePullSecrets:
        - name: default-secret
```

## 6.3.2 GPU Virtualization

### 6.3.2.1 Overview

CCE uses xGPU virtualization technologies to dynamically divide the GPU memory and computing power. A single GPU can be virtualized into up to 20 virtual GPU devices. Virtualization is more flexible than static allocation. You can specify the number of GPUs on the basis of stable service running to improve GPU utilization.

## Advantages

The GPU virtualization function of CCE has the following advantages:

- **Flexible:** The GPU computing ratio and memory size are finely tuned. The allocation granularity for computing is 5% of GPUs, while the memory allocation is in MiB.
- **Isolated:** The memory of a single GPU can be isolated, and both GPU computing and memory can be isolated concurrently.
- **Compatible:** There's no need to recompile services or replace the CUDA library.

## Prerequisites

Item	Supported Version
Cluster version	v1.23.8-r0, v1.25.3-r0, or later
OS	Huawei Cloud EulerOS 2.0
GPU type	T4 and V100
Driver version	470.57.02, 510.47.03, and 535.54.03
Runtime	containerd
Add-on	The following add-ons must be installed in the cluster: <ul style="list-style-type: none"><li>• <a href="#">Volcano Scheduler</a>: 1.10.5 or later</li><li>• <a href="#">CCE AI Suite (NVIDIA GPU)</a>: 2.0.5 or later</li></ul>

## Notes and Constraints

- A single GPU can be virtualized into a maximum of 20 xGPU devices.
- xGPUs cannot be used in init containers.
- GPU virtualization supports two isolation modes: GPU memory isolation and isolation between GPU memory and computing power. A single GPU can schedule only workloads in the same isolation mode.
- Autoscaler cannot be used to automatically scale in or out GPU nodes.
- xGPU isolation does not allow you to request for GPU memory by calling CUDA API `cudaMallocManaged()`, which is also known as using UVM. For more information, see [NVIDIA official documents](#). Use other methods to request for GPU memory, for example, by calling `cudaMalloc()`.
- When a containerized application is initializing, the real-time compute monitored by the `nvidia-smi` may exceed the upper limit of the available compute of the container.

### 6.3.2.2 Preparing xGPU Resources

CCE uses xGPU virtualization technologies to dynamically divide the GPU memory and computing power. A single GPU can be virtualized into up to 20 virtual GPU

devices. This section describes how to implement GPU scheduling and isolation capabilities on GPU nodes.

## Prerequisites

Item	Supported Version
Cluster version	v1.23.8-r0, v1.25.3-r0, or later
OS	Huawei Cloud EulerOS 2.0
GPU type	T4 and V100
Driver version	470.57.02, 510.47.03, and 535.54.03
Runtime	containerd
Add-on	The following add-ons must be installed in the cluster: <ul style="list-style-type: none"> <li><b>Volcano Scheduler:</b> 1.10.5 or later</li> <li><b>CCE AI Suite (NVIDIA GPU):</b> 2.0.5 or later</li> </ul>

## Step 1: Enable GPU Virtualization

Both **CCE AI Suite (NVIDIA GPU)** and **Volcano Scheduler** must be installed in the cluster.

## Step 2: Create a GPU Node

Create nodes that support GPU virtualization in the cluster to use the GPU virtualization function. For details, see [Creating a Node](#) or [Creating a Node Pool](#).

The screenshot shows a table of GPU node specifications. The 'GPU-accelerated' category is selected. The table lists various flavors with their vCPUs, memory, assured/maximum bandwidth, and packets per second (PPS). A red box highlights the 'GPU virtualization' label on the p2.2xlarge.4, p2.4xlarge.4, and p2.8xlarge.4 flavors.

Flavor	vCPUs   Memory	Assured/Maximum Bandwidth	Packets Per Second (PPS)
p3.6xlarge.4	24cores   96GiB	9.0/25.0 Gbit/s	4,000,000 pps
p3.12xlarge.4	48cores   192GiB	18.0/25.0 Gbit/s	7,600,000 pps
p2.2xlarge.4 GPU virtualization	8cores   32GiB	4.0/10.0 Gbit/s	500,000 pps
p2.4xlarge.4 GPU virtualization	16cores   64GiB	8.0/15.0 Gbit/s	1,000,000 pps
p2.8xlarge.4 GPU virtualization	32cores   128GiB	15.0/25.0 Gbit/s	2,000,000 pps
p1.2xlarge.4	8cores   32GiB	1.6/5.0 Gbit/s	400,000 pps
g5c.4xlarge.2	16cores   32GiB	8.0/15.0 Gbit/s	1,000,000 pps
g5c.8xlarge.2	32cores   64GiB	15.0/25.0 Gbit/s	2,000,000 pps
g5c.16xlarge.2	64cores   128GiB	30.0/40.0 Gbit/s	4,000,000 pps

### NOTE

If your cluster already has GPU nodes that meet the [Prerequisites](#), skip this step.

## Step 3 (Optional): Modifying the Volcano Scheduling Policy

The default scheduling policy of Volcano for GPU nodes is **Spread**. If the node configurations are the same, Volcano selects the node with the minimum number

of running containers, so that containers can be evenly allocated to each node. In contrast, the bin packing policy attempts to schedule all containers to one node to avoid resource fragmentation.

If the bin packing policy is required when the GPU virtualization feature is used, you can modify the policy in the advanced settings of the Volcano add-on. The procedure is as follows:

- 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** Find the Volcano add-on on the right and click **Edit**.
- Step 3** On the displayed page, modify the advanced settings.
  1. In the nodeorder add-on, add the **arguments** parameter and set **leastrequested.weight** to **0**. That is, set the priority of the node with the fewest allocated resources to **0**.
  2. Add the bin packing add-on, and specify the weights of xGPU customized resources (**volcano.sh/gpu-core.percentage** and **volcano.sh/gpu-mem.128Mi**).

A complete example is as follows:

```
{
  "colocation_enable": "",
  "default_scheduler_conf": {
    "actions": "allocate, backfill, preempt",
    "tiers": [
      {
        "plugins": [
          {
            "name": "priority"
          },
          {
            "enablePreemptable": false,
            "name": "gang"
          },
          {
            "name": "conformance"
          }
        ]
      },
      {
        "plugins": [
          {
            "enablePreemptable": false,
            "name": "drf"
          },
          {
            "name": "predicates"
          },
          {
            "name": "nodeorder",
            // Set the priority of the node with the fewest allocated resources to 0.
            "arguments": {
              "leastrequested.weight": 0
            }
          }
        ]
      }
    ],
    {
      "plugins": [
        {
          "name": "cce-gpu-topology-predicate"
        }
      ],

```

```
{
  "name": "cce-gpu-topology-priority"
},
{
  "name": "xgpu"
},
// Add the bin packing add-on, and specify the weights of xGPU resources.
{
  "name": "binpack",
  "arguments": {
    "binpack.resources": "volcano.sh/gpu-core.percentage,volcano.sh/gpu-mem.128Mi",
    "binpack.resources.volcano.sh/gpu-mem.128Mi": 10,
    "binpack.resources.volcano.sh/gpu-core.percentage": 10
  }
}
]
},
{
  "plugins": [
    {
      "name": "nodelocalvolume"
    },
    {
      "name": "nodeemptydirvolume"
    },
    {
      "name": "nodeCSIscheduling"
    },
    {
      "name": "networkresource"
    }
  ]
}
],
"tolerations": [
  {
    "effect": "NoExecute",
    "key": "node.kubernetes.io/not-ready",
    "operator": "Exists",
    "tolerationSeconds": 60
  },
  {
    "effect": "NoExecute",
    "key": "node.kubernetes.io/unreachable",
    "operator": "Exists",
    "tolerationSeconds": 60
  }
]
}
```

----End

### 6.3.2.3 Using GPU Virtualization

This section describes how to use the GPU virtualization capability to isolate the computing power from the GPU memory and efficiently use GPU device resources.

#### Prerequisites

- You have [prepared GPU virtualization resources](#).
- If you want to create a cluster using commands, use `kubectl` to access the cluster. For details, see [Connecting to a Cluster Using kubectl](#).

## Notes and Constraints

- A single GPU can be virtualized into a maximum of 20 xGPU devices.
- xGPUs cannot be used in init containers.
- GPU virtualization supports two isolation modes: GPU memory isolation and isolation between GPU memory and computing power. A single GPU can schedule only workloads in the same isolation mode.
- Autoscaler cannot be used to automatically scale in or out GPU nodes.
- xGPU isolation does not allow you to request for GPU memory by calling CUDA API `cudaMallocManaged()`, which is also known as using UVM. For more information, see [NVIDIA official documents](#). Use other methods to request for GPU memory, for example, by calling `cudaMalloc()`.
- When a containerized application is initializing, the real-time compute monitored by the `nvidia-smi` may exceed the upper limit of the available compute of the container.

## Creating a GPU Virtualization Application

### Using the CCE Console

- Step 1** Log in to the CCE console.
- Step 2** Click the cluster name to go to the cluster console, choose **Workloads** in the navigation pane, and click the **Create Workload** in the upper right corner.
- Step 3** Set basic information about the workload.

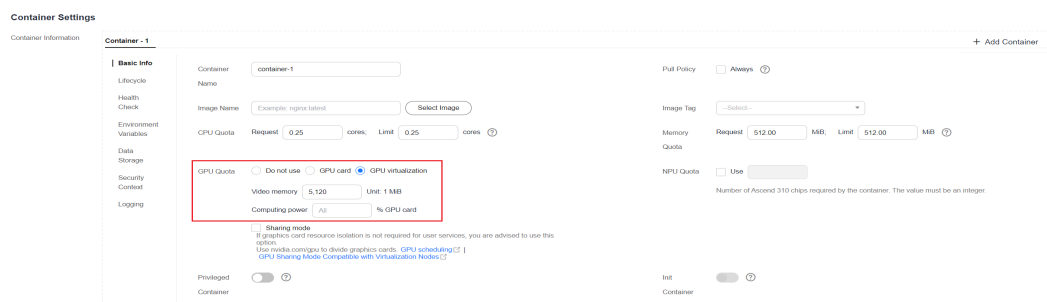
Choose **Container Settings** > **Basic Info** and configure the GPU quota.

- **Video memory:** The unit is MiB. The value must be a positive integer that is a multiple of 128. If the value exceeds the memory of a single GPU, scheduling cannot be performed.
- **Computing power:** The unit is %. The value must be a multiple of 5 and cannot exceed 100.

#### NOTE

- If the GPU memory is set to the capacity upper limit of a single GPU or the computing power is set to 100%, the entire GPU will be used.
- When GPU virtualization is used, the workload scheduler defaults to Volcano and cannot be changed.

**Figure 6-3** Configuring the xGPU quota





This section describes how to use GPU virtualization. For details about other parameters, see [Workloads](#).

After completing the setting, click **Create**.

**Step 4** After a workload is created, you can try to verify the isolation capability of GPU virtualization.

1. Log in to the pod and check the total GPU memory allocated to the pod.

```
kubectl exec -it gpu-app -- nvidia-smi
```

Expected output:

```
Wed Apr 12 07:54:59 2023
+-----+
| NVIDIA-SMI 470.141.03   Driver Version: 470.141.03   CUDA Version: 11.4   |
+-----+-----+
| GPU Name      Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|      Memory-Usage | GPU-Util  Compute M. |
|               |              | MIG M. |
+-----+-----+-----+-----+
|  0  Tesla V100-SXM2...  Off | 00000000:21:01.0 Off |             0 |
| N/A   27C   P0   37W / 300W | 4912MiB / 5120MiB |    0%    Default |
|               |              | N/A |
+-----+-----+-----+-----+

+-----+
| Processes:
| GPU  GI  CI       PID   Type   Process name          GPU Memory |
|   ID ID             |           |         | Usage           |
+-----+-----+-----+-----+
|  0   N/A  N/A     760445    C   python                4835MiB |
+-----+-----+-----+-----+
```

The expected output indicates that the total GPU memory allocated to the pod is 5120 MiB, and 4912 MiB is used.

2. Run the following command on the node to check the isolation status of the GPU memory:

```
nvidia-smi
```

Expected output:

```
Wed Apr 12 09:31:10 2023
+-----+
| NVIDIA-SMI 470.141.03   Driver Version: 470.141.03   CUDA Version: 11.4   |
+-----+-----+
| GPU Name      Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|      Memory-Usage | GPU-Util  Compute M. |
|               |              | MIG M. |
+-----+-----+-----+-----+
|  0  Tesla V100-SXM2...  Off | 00000000:21:01.0 Off |             0 |
| N/A   27C   P0   37W / 300W | 4957MiB / 16160MiB |    0%    Default |
|               |              | N/A |
+-----+-----+-----+-----+

+-----+
| Processes:
| GPU  GI  CI       PID   Type   Process name          GPU Memory |
|   ID ID             |           |         | Usage           |
+-----+-----+-----+-----+
|  0   N/A  N/A     760445    C   python                4835MiB |
+-----+-----+-----+-----+
```

The expected output indicates that the total GPU memory on the node is 16160 MiB, and the example pod uses 4957 MiB.

----End

**Using kubectl**

**Step 1** Use kubectl to access the cluster.

**Step 2** Create an application that uses GPU virtualization.

 **NOTE**

The GPU memory isolation and isolation between GPU memory and computing power are supported. The computing power cannot be isolated only. **volcano.sh** and **gpu-core.percentage** cannot be set separately.

Create a **gpu-app.yaml** file with the following content.

- Isolate GPU memory only:

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: gpu-app
  labels:
    app: gpu-app
spec:
  replicas: 1
  selector:
    matchLabels:
      app: gpu-app
  template:
    metadata:
      labels:
        app: gpu-app
    spec:
      containers:
        - name: container-1
          image: <your_image_address> # Replace it with your image address.
      resources:
        limits:
          volcano.sh/gpu-mem.128Mi: 40 # GPU memory allocated to the pod. The value is a
          multiple of 128 MiB (40 x 128 = 5120 MiB).
          imagePullSecrets:
            - name: default-secret
          schedulerName: volcano
```

- Isolate the GPU memory from computing power:

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: gpu-app
  labels:
    app: gpu-app
spec:
  replicas: 1
  selector:
    matchLabels:
      app: gpu-app
  template:
    metadata:
      labels:
        app: gpu-app
    spec:
      containers:
        - name: container-1
          image: <your_image_address> # Replace it with your image address.
      resources:
        limits:
          volcano.sh/gpu-mem.128Mi: 40 # GPU memory allocated to the pod. The value is a
          multiple of 128 MiB (40 x 128 = 5120 MiB).
          volcano.sh/gpu-core.percentage: 25 # Computing power allocated to the pod
          imagePullSecrets:
            - name: default-secret
          schedulerName: volcano
```

**Table 6-1** Key parameters

Parameter	Mandatory	Description
volcano.sh/gpu-mem.128Mi	No	The value is a positive integer that is a multiple of 128 in the unit of MiB. If the value exceeds the memory of a single GPU, scheduling cannot be performed.
volcano.sh/gpu-core.percentage	No	The unit of the computing power is %. The value must be a multiple of 5 and cannot exceed 100.

 **NOTE**

- If the GPU memory is set to the capacity upper limit of a single GPU or the computing power is set to 100%, the entire GPU will be used.
- When GPU virtualization is used, the workload scheduler defaults to Volcano and cannot be changed.

**Step 3** Run the following command to create an application:

```
kubectl apply -f gpu-app.yaml
```

**Step 4** Verify the isolation capability of GPU virtualization.

1. Log in to the pod and check the total GPU memory allocated to the pod.

```
kubectl exec -it gpu-app -- nvidia-smi
```

Expected output:

```
Wed Apr 12 07:54:59 2023
+-----+
| NVIDIA-SMI 470.141.03   Driver Version: 470.141.03   CUDA Version: 11.4   |
+-----+-----+
| GPU Name      Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|      Memory-Usage | GPU-Util  Compute M. |
|               |              | MIG M. |
+-----+-----+-----+-----+
| 0  Tesla V100-SXM2...  Off | 00000000:21:01:0  Off |          0 |
|N/A   27C    P0   37W / 300W | 4912MiB / 5120MiB |    0%    Default |
|               |              | N/A |
+-----+-----+-----+-----+
+-----+
| Processes:
| GPU  GI  CI       PID   Type   Process name          GPU Memory |
|   ID ID          |          |            | Usage           |
+-----+-----+-----+-----+
+-----+
```

The expected output indicates that the total GPU memory allocated to the pod is 5120 MiB, and 4912 MiB is used.

2. Run the following command on the node to check the isolation status of the GPU memory:

```
nvidia-smi
```

Expected output:

```
Wed Apr 12 09:31:10 2023
+-----+
```

```

| NVIDIA-SMI 470.141.03   Driver Version: 470.141.03   CUDA Version: 11.4   |
+-----+-----+-----+-----+-----+-----+
| GPU  Name      Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|  Memory-Usage | GPU-Util  Compute M. |
|                                           MIG M. |
+-----+-----+-----+-----+-----+-----+
|  0   Tesla V100-SXM2...  Off   | 00000000:21:01.0 Off  |      0 |
| N/A   27C   P0   37W / 300W | 4957MiB / 16160MiB |    0%   Default |
|                                           N/A   |
+-----+-----+-----+-----+-----+
|
+-----+-----+-----+-----+-----+
| Processes:
| GPU  GI  CI       PID   Type   Process name          GPU Memory |
| ID   ID                          Usage          |
+-----+-----+-----+-----+-----+
|  0   N/A  N/A   760445    C    python                4835MiB |
+-----+-----+-----+-----+-----+

```

The expected output indicates that the total GPU memory on the node is 16160 MiB, and the example pod uses 4957 MiB.

----End

### 6.3.2.4 Supporting Kubernetes' Default GPU Scheduling

After GPU virtualization is enabled, the target GPU node does not support the workloads that use [Kubernetes' default GPU scheduling](#) by default, which are workloads using `nvidia.com/gpu` resources. If there are workloads using `nvidia.com/gpu` resources in your cluster, you can enable the GPU node to support GPU sharing in the `gpu-device-plugin` configuration so that the GPU node can support Kubernetes' default GPU scheduling.

- Enabling compatibility and setting `nvidia.com/gpu` to a decimal fraction (for example, **0.5**) allow GPU virtualization to provide the specified `nvidia.com/gpu` quota for GPU memory isolation in workloads. Containers will receive GPU memory allocated based on the quota, such as 8 GiB (0.5 x 16 GiB). The GPU memory value must be a multiple of 128 MiB, or it will be rounded down to the nearest integer automatically. If `nvidia.com/gpu` resources have been used in the workload before compatibility is enabled, the resources will be from the entire GPU but not GPU virtualization.
- Enabling compatibility is equivalent to enabling GPU memory isolation when using the `nvidia.com/gpu` quota. The quota can be shared with workloads in GPU memory isolation mode, but not with those in compute and GPU memory isolation mode. In addition, [Notes and Constraints](#) on GPU virtualization must be followed.
- If compatibility is disabled, the `nvidia.com/gpu` quota specified in the workload only affects scheduling and is not restricted by GPU memory isolation. For example, even if you set the `nvidia.com/gpu` quota to **0.5**, you can still access the entire GPU memory within the container. In addition, workloads using `nvidia.com/gpu` resources and workloads using virtualized GPU memory cannot be scheduled to the same node.
- If you deselect **Virtualization nodes are compatible with GPU sharing mode**, running workloads will not be affected, but workloads may fail to be scheduled. For example, if compatibility is disabled, workloads that use `nvidia.com/gpu` resources will remain in GPU memory isolation mode. As a result, the GPU cannot schedule workloads in compute and GPU memory

isolation mode. To reschedule, you must remove the workloads that use **nvidia.com/gpu** resources.

## Notes and Constraints

To support Kubernetes' default GPU scheduling on GPU nodes, the CCE AI Suite (NVIDIA GPU) add-on must be of v2.0.10 or later, and the Volcano Scheduler add-on must be of v1.10.5 or later.

## Procedure

**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 **CCE AI Suite (NVIDIA GPU)** on the right and click **Install**.

If the add-on has been installed, click **Edit**.

**Step 3** Configure the add-on. For details, see [Installing the add-on](#).

After GPU virtualization is enabled, you can configure the **nvidia.com/gpu** field to enable or disable the function of supporting Kubernetes' default GPU scheduling.

**Step 4** Click **Install**.

----End

## Configuration Example

**Step 1** Use `kubectl` to access the cluster.

**Step 2** Create a workload that uses `nvidia.com/gpu` resources.

Create a **gpu-app.yaml** file. The following shows an example:

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: gpu-app
  namespace: default
spec:
  replicas: 1
  selector:
    matchLabels:
      app: gpu-app
  template:
    metadata:
      labels:
        app: gpu-app
    spec:
      schedulerName: volcano
      containers:
        image: <your_image_address> # Replace it with your image address.
        name: container-0
        resources:
          requests:
            cpu: 250m
            memory: 512Mi
            nvidia.com/gpu: 0.1 # Number of requested GPUs
          limits:
            cpu: 250m
            memory: 512Mi
            nvidia.com/gpu: 0.1 # Maximum number of GPUs that can be used
```

```
imagePullSecrets:
- name: default-secret
```

**Step 3** Run the following command to create an application:

```
kubectl apply -f gpu-app.yaml
```

**Step 4** Log in to the pod and check the total GPU memory allocated to the pod.

```
kubectl exec -it gpu-app -- nvidia-smi
```

Expected output:

```
Thu Jul 27 07:53:49 2023
+-----+
| NVIDIA-SMI 470.57.02    Driver Version: 470.57.02    CUDA Version: 11.4    |
+-----+
| GPU  Name        Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf    Pwr:Usage/Cap|      Memory-Usage | GPU-Util  Compute M. |
|                                           MIG M. |
+-----+-----+
|  0  NVIDIA A30           Off   | 00000000:00:0D:0  Off   |      0 |
| N/A  47C   P0     34W / 165W |  0MiB / 2304MiB |  0%      Default |
|                                           Disabled |
+-----+-----+
+-----+
| Processes:
| GPU  GI  CI       PID   Type   Process name          GPU Memory |
|   ID  ID                 |                   Usage      |
+-----+-----+
| No running processes found
+-----+
```

The output shows that the total GPU memory that can be used by the pod is 2304 MiB.

In this example, the total GPU memory on the GPU node is 24258 MiB, but the number 2425.8 (24258 x 0.1) is not an integer multiple of 128 MiB. Therefore, the value 2425.8 is rounded down to 18 times of 128 MiB (18 x 128 MiB = 2304 MiB).

----End

### 6.3.3 Monitoring GPU Metrics

You can use Prometheus and Grafana to observe GPU metrics. This section uses Prometheus as an example to describe how to view the GPU memory usage of a cluster.

The process is as follows:

1. [Accessing Prometheus](#)

(Optional) Bind a LoadBalancer Service to Prometheus so that Prometheus can be accessed from external networks.

2. [Monitoring GPU Metrics](#)

After a GPU workload is deployed in the cluster, GPU metrics will be automatically reported.

3. [Accessing Grafana](#)

View Prometheus monitoring data on Grafana, a visualization panel.

## Prerequisites

- The [Cloud Native Cluster Monitoring](#) add-on has been installed in the cluster.
- The [CCE AI Suite \(NVIDIA GPU\)](#) add-on has been installed in the cluster, and the add-on version is 2.0.10 or later.
- To monitor GPU virtualization metrics, ensure [Volcano Scheduler](#) has been installed in the cluster and the add-on version is 1.10.5 or later.

## Accessing Prometheus

After the Prometheus add-on is installed, you can deploy workloads and Services. The Prometheus server will be deployed as a StatefulSet in the **monitoring** namespace.

You can create a public network [LoadBalancer Service](#) so that Prometheus can be accessed from an external network.

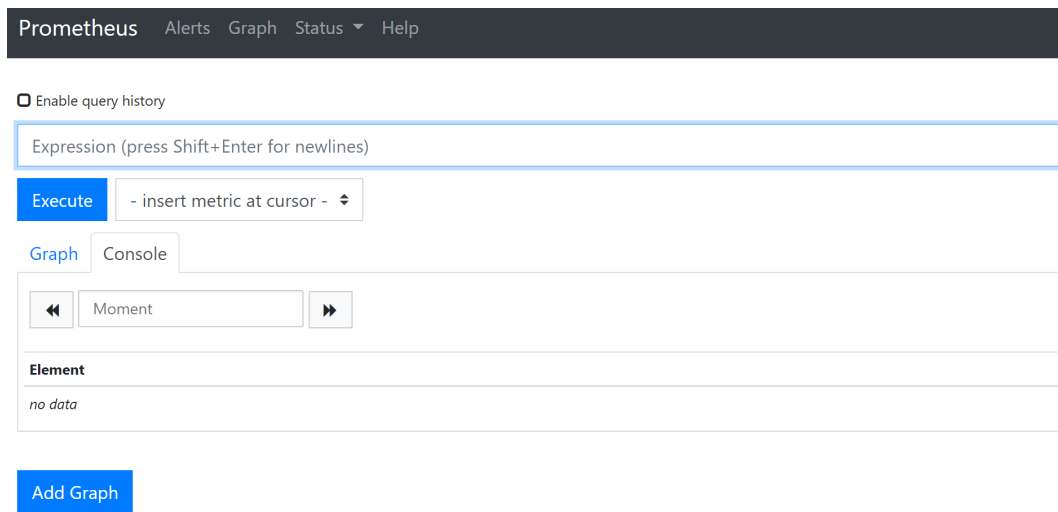
**Step 1** Log in to the CCE console and click the name of the cluster with Prometheus installed to access the cluster console. In the navigation pane, choose **Services & Ingresses**.

**Step 2** Click **Create from YAML** in the upper right corner to create a public network LoadBalancer Service.

```
apiVersion: v1
kind: Service
metadata:
  name: prom-lb # Service name, which is customizable.
  namespace: monitoring
  labels:
    app: prometheus
    component: server
  annotations:
    kubernetes.io/elb.id: 038ff*** # Replace it with the ID of the public network load balancer in the VPC
    that the cluster belongs to.
spec:
  ports:
    - name: cce-service-0
      protocol: TCP
      port: 88 # Service port, which is customizable.
      targetPort: 9090 # Default Prometheus port. Retain the default value.
  selector: # The label selector can be adjusted based on the label of a Prometheus server
    instance.
    app.kubernetes.io/name: prometheus
    prometheus: server
  type: LoadBalancer
```

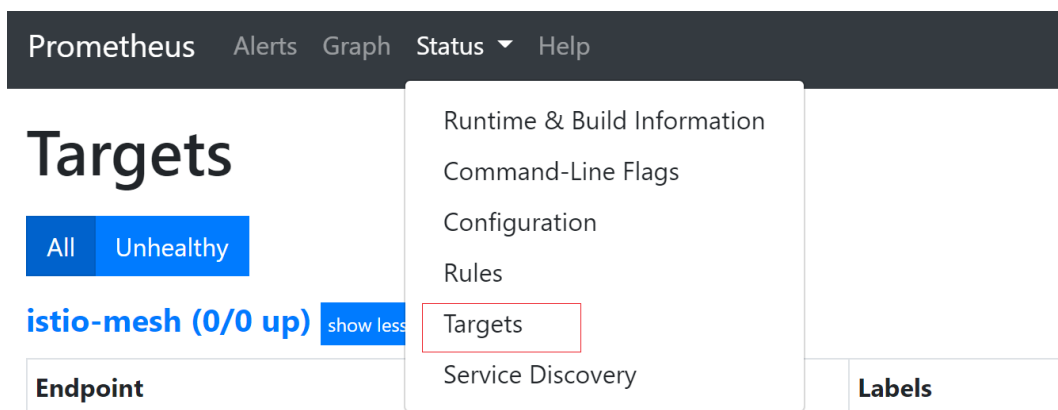
**Step 3** After the Service is created, visit **Public IP address of the load balancer.Service port** to access Prometheus.

**Figure 6-4** Accessing Prometheus



**Step 4** Choose **Status > Targets** to view the targets monitored by Prometheus.

**Figure 6-5** Viewing monitored targets

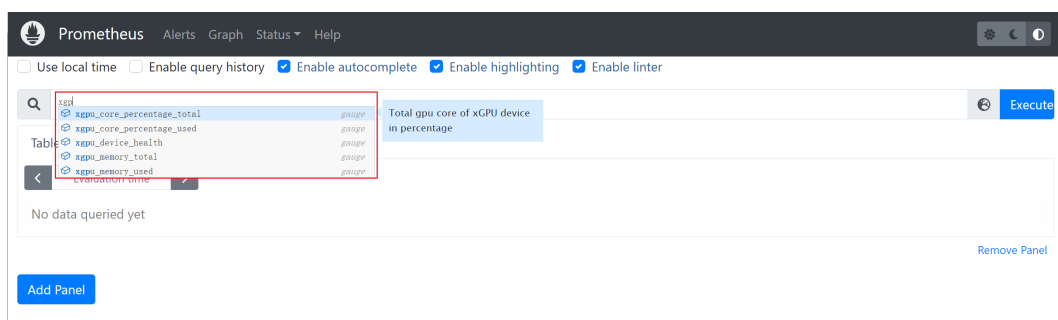


----End

## Monitoring GPU Metrics

Create a GPU workload. After the workload runs properly, access Prometheus and view GPU metrics on the **Graph** page.

**Figure 6-6** Viewing GPU metrics





**Table 6-2** Basic GPU monitoring metrics

Type	Metric	Monitoring Level	Description
Utilization	cce_gpu_utilization	GPU cards	GPU compute usage
	cce_gpu_memory_utilization	GPU cards	GPU memory usage
	cce_gpu_encoder_utilization	GPU cards	GPU encoding usage
	cce_gpu_decoder_utilization	GPU cards	GPU decoding usage
	cce_gpu_utilization_process	GPU processes	GPU compute usage of each process
	cce_gpu_memory_utilization_process	GPU processes	GPU memory usage of each process
	cce_gpu_encoder_utilization_process	GPU processes	GPU encoding usage of each process
	cce_gpu_decoder_utilization_process	GPU processes	GPU decoding usage of each process
Memory	cce_gpu_memory_used	GPU cards	Used GPU memory
	cce_gpu_memory_total	GPU cards	Total GPU memory
	cce_gpu_memory_free	GPU cards	Free GPU memory
	cce_gpu_bar1_memory_used	GPU cards	Used GPU BAR1 memory
	cce_gpu_bar1_memory_total	GPU cards	Total GPU BAR1 memory
Frequency	cce_gpu_clock	GPU cards	GPU clock frequency
	cce_gpu_memory_clock	GPU cards	GPU memory frequency
	cce_gpu_graphics_clock	GPU cards	GPU frequency
	cce_gpu_video_clock	GPU cards	GPU video processor frequency
Physical status	cce_gpu_temperature	GPU cards	GPU temperature
	cce_gpu_power_usage	GPU cards	GPU power

Type	Metric	Monitoring Level	Description
	cce_gpu_total_energy_consumption	GPU cards	Total GPU energy consumption
Bandwidth	cce_gpu_pcie_link_bandwidth	GPU cards	GPU PCIe bandwidth
	cce_gpu_nvlink_bandwidth	GPU cards	GPU NVLink bandwidth
	cce_gpu_pcie_throughput_rx	GPU cards	GPU PCIe RX bandwidth
	cce_gpu_pcie_throughput_tx	GPU cards	GPU PCIe TX bandwidth
	cce_gpu_nvlink_utilization_counter_rx	GPU cards	GPU NVLink RX bandwidth
	cce_gpu_nvlink_utilization_counter_tx	GPU cards	GPU NVLink TX bandwidth
Memory isolation page	cce_gpu_retired_pages_sbe	GPU cards	Number of isolated GPU memory pages with single-bit errors
	cce_gpu_retired_pages_dbe	GPU cards	Number of isolated GPU memory pages with dual-bit errors

**Table 6-3** xGPU metrics

Metric	Monitoring Level	Description
xgpu_memory_total	GPU processes	Total xGPU memory
xgpu_memory_used	GPU processes	Used xGPU memory
xgpu_core_percentage_total	GPU processes	Total xGPU cores
xgpu_core_percentage_used	GPU processes	Used xGPU cores

Metric	Monitoring Level	Description
gpu_schedule_policy	GPU cards	xGPU scheduling policy. Options: <ul style="list-style-type: none"> <li>• <b>0</b>: xGPU memory is isolated and cores are shared.</li> <li>• <b>1</b>: Both xGPU memory and cores are isolated.</li> <li>• <b>2</b>: default mode, indicating that the current card is not used by any xGPU device for allocation.</li> </ul>
xgpu_device_health	GPU cards	Health status of an xGPU device. Options: <ul style="list-style-type: none"> <li>• <b>0</b>: The xGPU device is healthy.</li> <li>• <b>1</b>: The xGPU device is not healthy.</li> </ul>

## Accessing Grafana

The Prometheus add-on has had [Grafana](#) (an open-source visualization tool) installed and interconnected. You can create a public network [LoadBalancer Service](#) so that you can access Grafana from the public network and view Prometheus monitoring data on Grafana.

Click the access address to access Grafana and select a proper dashboard to view the aggregated content.

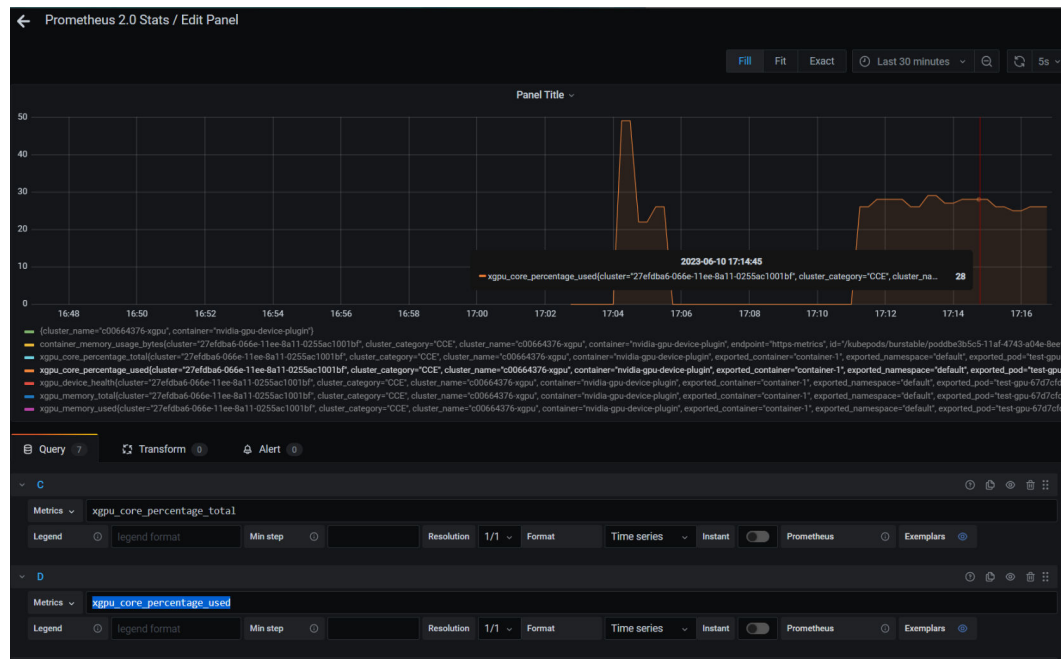
**Step 1** Log in to the CCE console and click the name of the cluster with Prometheus installed to access the cluster console. In the navigation pane, choose **Services & Ingresses**.

**Step 2** Click **Create from YAML** in the upper right corner to create a public network LoadBalancer Service for Grafana.

```
apiVersion: v1
kind: Service
metadata:
  name: grafana-lb # Service name, which is customizable
  namespace: monitoring
  labels:
    app: grafana
  annotations:
    kubernetes.io/elb.id: 038ff*** # Replace it with the ID of the public network load balancer in the VPC
    to which the cluster belongs.
spec:
  ports:
    - name: cce-service-0
      protocol: TCP
      port: 80 # Service port, which is customizable
      targetPort: 3000 # Default Grafana port. Retain the default value.
  selector:
    app: grafana
  type: LoadBalancer
```

**Step 3** After the Service is created, visit **Public IP address of the load balancer.Service port** to access Grafana and select a proper dashboard to view xGPU resources.

Figure 6-7 Viewing xGPU resources



----End

### 6.3.4 Configuring Workload Scaling Based on GPU Monitoring Metrics

If there are GPU nodes in a cluster, you can view the GPU resource usage of the nodes through GPU metrics, such as the GPU usage and used GPU memory. After obtaining GPU monitoring metrics, you can configure auto scaling policies based on the GPU metrics of applications to adaptively adjust the number of nodes for the applications when services fluctuate.

#### Prerequisites

- A cluster is available, and there are GPU nodes and GPU related services running in the cluster.
- The **CCE AI Suite (NVIDIA GPU)** add-on has been installed in the cluster, and the add-on metrics API is working properly. You can log in to the GPU node and run the following command:  

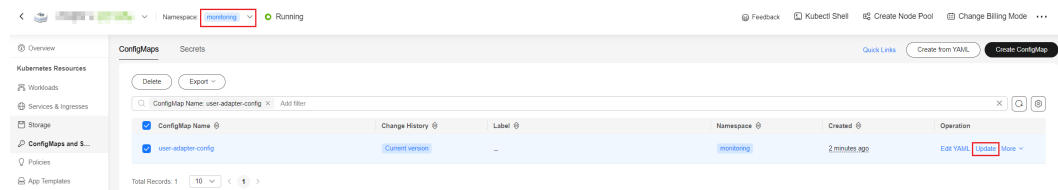
```
curl {Pod IP}:2112/metrics
```

In the preceding command, *{Pod IP}* indicates the pod IP address of the GPU add-on. If the metric result is returned, the GPU add-on is running properly.
- The **Cloud Native Cluster Monitoring** add-on of v3.9.5 or later has been installed in the cluster, and it is deployed in local data storage mode.

#### Collecting GPU Metrics

- Step 1** Log in to the CCE console and click the cluster name to access the cluster console. In the navigation pane, choose **ConfigMaps and Secrets**.
- Step 2** Select the **monitoring** namespace. On the **ConfigMaps** tab, locate the row containing **user-adapter-config** and click **Update**.

**Figure 6-8** Updating a ConfigMap



**Step 3** On the **Update ConfigMap** page, click **Edit** in the **Operation** column of the **config.yaml** file in the **Data** pane. Then, add a custom metric collection rule under the **rules** field. Click **OK**.

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 rule for collecting **cce\_gpu\_memory\_utilization**. For details about more GPU metrics, see [Monitoring GPU Metrics](#).

```
rules:
- seriesQuery: '{__name__=~"cce_gpu_memory_utilization",container!="",namespace!="",pod!=""}'
  seriesFilters: []
  resources:
    overrides:
      namespace:
        resource: namespace
      pod:
        resource: pod
  metricsQuery: sum(last_over_time(<<.Series>>{<<.LabelMatchers>>}[1m])) by (<&lt.GroupBy>>)
```

**Figure 6-9** Customizing a collection rule

✕

### Update ConfigMap

Name:

Namespace: monitoring

Description:  0/255 ↕

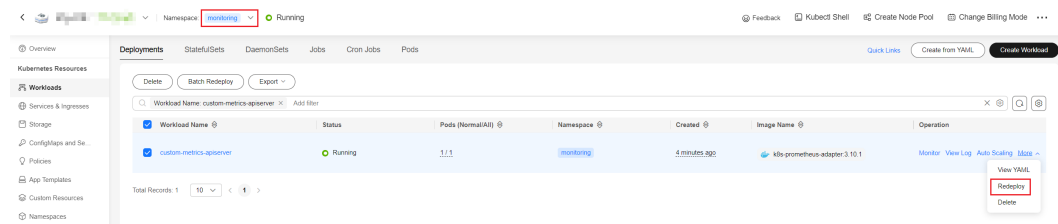
Data	Key	Value	Operation
	config.yaml	rules: [] resourceRules: cpu: containerQu...	<span style="border: 1px solid red; padding: 2px 5px; color: red;">Edit</span> <span style="padding: 2px 5px; color: red;">Delete</span>
+ <input style="width: 100%; border: 1px dashed #ccc;" type="text"/>			

Label:  =  Confirm

Version Management:  Enabling this function allows ConfigMap data to be backed up before modification, so that you can easily manage and roll back data.

**Step 4** Redeploy the **custom-metrics-apiserver** workload in the **monitoring** namespace.

**Figure 6-10** Redeploying custom-metrics-apiserver



**Step 5** After the restart, check whether the metrics in the target pod are normal (replace the namespace and service pod names).

```
# Obtain metrics.
$ kubectl get --raw "/apis/custom.metrics.k8s.io/v1beta1"

{"kind":"APIResourceList","apiVersion":"v1","groupVersion":"custom.metrics.k8s.io/v1beta1","resources":
[{"name":"pods/
cce_gpu_memory_utilization","singularName":"","namespaced":true,"kind":"MetricValueList","verbs":["get"]},
{"name":"namespaces/
cce_gpu_memory_utilization","singularName":"","namespaced":false,"kind":"MetricValueList","verbs":
["get"]}]}

# Obtain workload metric values.
$ kubectl get --raw "/apis/custom.metrics.k8s.io/v1beta1/namespaces/default/pods/test-gpu-
hpa-68667fdd94-grmd2/cce_gpu_memory_utilization"

{"kind":"MetricValueList","apiVersion":"custom.metrics.k8s.io/v1beta1","metadata":{"},"items":
[{"describedObject":{"kind":"Pod","namespace":"default","name":"test-gpu-hpa-68667fdd94-
grmd2","apiVersion":"/
v1"},"metricName":"cce_gpu_memory_utilization","timestamp":"2024-01-10T08:36:44Z","value":"20","selecto
r":null}]}

----End
```

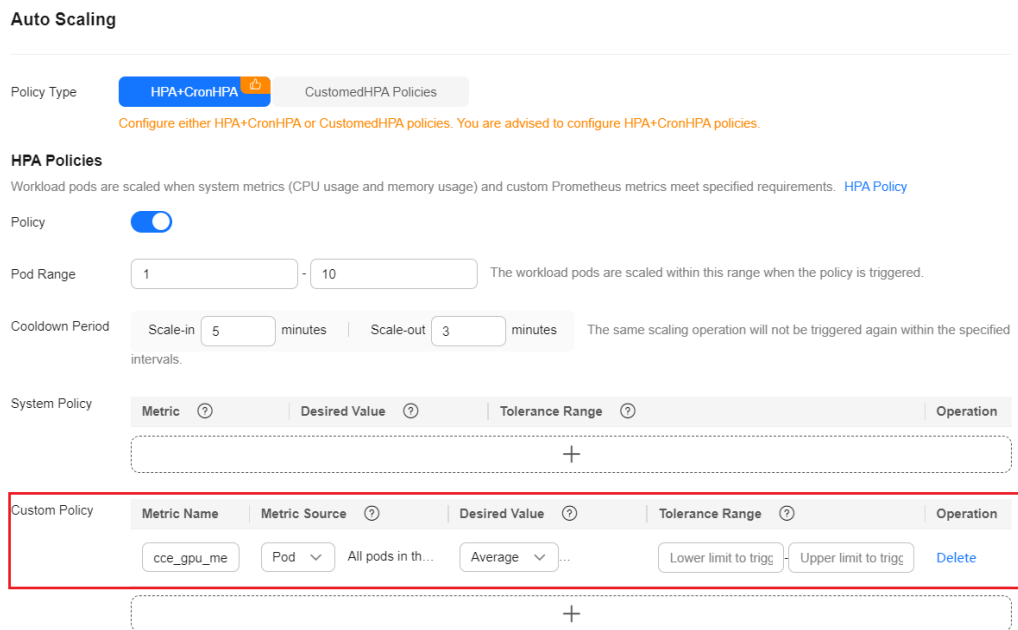
## Creating an Auto Scaling Policy

**Step 1** Choose **Workloads** in the navigation pane. Locate the target workload and choose **Auto Scaling** in the **Operation** column.

**Step 2** Set **Policy Type** to **HPA+CronHPA** and enable HPA.

You can select GPU monitoring parameters in **Custom Policy** to create an auto scaling policy. The following is an example.

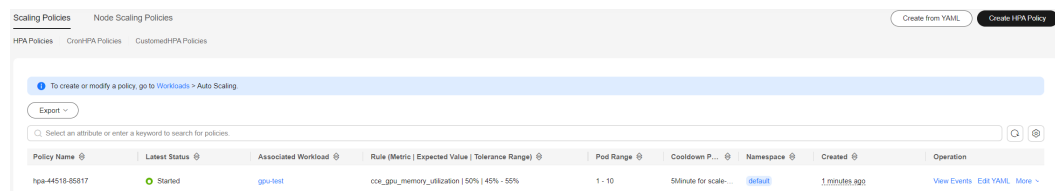
**Figure 6-11** Selecting custom metrics



In this example, **cce\_gpu\_memory\_utilization** (GPU memory usage) is used as the scaling metric. For details about how to configure other HPA parameters, see [Creating an HPA Policy](#).

**Step 3** Return to the **Scaling Policies** page and check whether the HPA policy has been created.

**Figure 6-12** HPA policy created



----End

## 6.4 NPU Scheduling

You can use NPUs in CCE containers.

### Prerequisites

- An NPU node has been created. For details, see [Creating a Node](#).
- The huawei-npu has been installed. For details, see [CCE AI Suite \(Ascend NPU\)](#).

### Using NPUs

Create a workload and request NPUs. You can specify the number of NPUs as follows:

```
kind: Deployment
apiVersion: apps/v1
metadata:
  name: npu-test
  namespace: default
spec:
  replicas: 1
  selector:
    matchLabels:
      app: npu-test
  template:
    metadata:
      labels:
        app: npu-test
    spec:
      containers:
        - name: container-0
          image: nginx:perl
          resources:
            limits:
              cpu: 250m
              huawei.com/ascend-310: '1'
              memory: 512Mi
            requests:
              cpu: 250m
              huawei.com/ascend-310: '1'
              memory: 512Mi
          imagePullSecrets:
            - name: default-secret
```

Specify the number of NPUs to be requested in **huawei.com/ascend-310**.

#### NOTE

When you use **huawei.com/ascend-310** to specify the number of NPUs, the values of requests and limits must be the same.

After **huawei.com/ascend-310** is specified, workloads will be scheduled only to nodes with NPUs. If NPUs are insufficient, a Kubernetes event similar to "0/2 nodes are available: 2 Insufficient huawei.com/ascend-310." will be reported.

To use NPUs on the CCE console, select the NPU quota and specify the number of NPUs to be used when creating a workload.

## NPU Node Labels

CCE will label NPU-enabled nodes that are ready to use.

```
$ kubectl get node -L accelerator/huawei-npu
NAME          STATUS  ROLES  AGE   VERSION                                HUAWEI-NPU
10.100.2.59   Ready  <none> 2m18s v1.19.10-r0-CCE21.11.1.B006-21.11.1.B006 ascend-310
```

When using NPUs, you can enable the affinity between pods and nodes based on labels so that the pods can be scheduled to the correct nodes.

```
kind: Deployment
apiVersion: apps/v1
metadata:
  name: npu-test
  namespace: default
spec:
  replicas: 1
  selector:
    matchLabels:
      app: npu-test
  template:
    metadata:
```



```
labels:
  app: npu-test
spec:
  nodeSelector:
    accelerator/huawei-npu: ascend-310
  containers:
  - name: container-0
    image: nginx:perl
    resources:
      limits:
        cpu: 250m
        huawei.com/ascend-310: '1'
        memory: 512Mi
      requests:
        cpu: 250m
        huawei.com/ascend-310: '1'
        memory: 512Mi
    imagePullSecrets:
    - name: default-secret
```

## 6.5 Volcano Scheduling

### 6.5.1 Scheduling Workloads

**Volcano** is a Kubernetes-based batch processing platform with high-performance general computing capabilities like task scheduling engine, heterogeneous chip management, and task running management. It provides end users with computing frameworks from multiple domains such as AI, big data, gene, and rendering. It also offers job scheduling, job management, and queue management for computing applications.

Kubernetes typically uses its default scheduler to schedule workloads. To use Volcano, specify Volcano for your workloads. For details about the Kubernetes scheduler, see [Specify schedulers for pods](#).

### Notes and Constraints

If you schedule a lot of workloads, Volcano will generate a significant number of logs. To prevent the node hosting Volcano from running out of disk space, use Volcano with LTS.

### Using Volcano

When using Volcano to schedule workloads, you only need to configure **schedulerName** in the **spec** field of the pod and set the parameter to **volcano**. The following is an example:

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx
  labels:
    app: nginx
spec:
  replicas: 4
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
```

```

annotations:
  # Submit the job to the q1 queue.
  scheduling.volcano.sh/queue-name: "q1"
  volcano.sh/preemptable: "true"
labels:
  app: nginx
spec:
  # Specify Volcano as the scheduler.
  schedulerName: volcano
containers:
  - name: nginx
    image: nginx
    imagePullPolicy: IfNotPresent
resources:
  limits:
    cpu: 1
    memory: 100Mi
  requests:
    cpu: 1
    memory: 100Mi
ports:
  - containerPort: 80
    
```

Additionally, Volcano supports the workload queues and preemption, which can be implemented through pod annotations. The following table lists the supported annotations.

**Table 6-4** Pod annotations supported by Volcano

Pod Annotations	Description
scheduling.volcano.sh/queue-name: "<queue-name>"	Specifies the queue to which the workload belongs. <queue-name> indicates the queue name.
volcano.sh/preemptable: "true"	Indicates whether a job can be preempted. If this function is enabled, the job can be preempted. Options: <ul style="list-style-type: none"> <li>● <b>true</b>: Preemption is enabled. This option is enabled by default.</li> <li>● <b>false</b>: Preemption is disabled.</li> </ul>

You can obtain pod details to check whether the pod is scheduled by Volcano and the allocated queue.

```
kubectl describe pod <pod_name>
```

Command output:

```

Spec:
  Min Member: 1
  Min Resources:
    Cpu: 100m
    Memory: 100Mi
  Queue: q1
Status:
  Conditions:
    Last Transition Time: 2023-05-30T01:54:43Z
    Reason: tasks in gang are ready to be scheduled
    
```

```

Status:      True
Transition ID: 70be1d7d-3532-41e0-8324-c7644026b38f
Type:        Scheduled
Phase:       Running
Events:
Type  Reason  Age          From  Message
----  -
Normal Scheduled 0s (x3 over 2s) volcano pod group is ready
    
```

## 6.5.2 Resource Usage-based Scheduling

### 6.5.2.1 Bin Packing

Bin packing is an optimization algorithm that aims to properly allocate resources to each job and get the jobs done using the minimum amount of resources. 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.

#### Prerequisites

- A cluster of v1.19 or later is available. For details, see [Buying a CCE Standard/Turbo Cluster](#).
- The Volcano add-on has been installed. For details, see [Volcano Scheduler](#).

#### Features

Bin packing aims to fill as many existing nodes as possible (try not to allocate blank nodes). In the concrete implementation, the bin packing algorithm scores the nodes that can be delivered, and a higher score indicates a higher resource utilization rate of nodes. Bin packing intends to centrally schedule application workloads onto some nodes in a cluster, which facilitates auto scaling of cluster nodes.

The bin packing add-on works with other scheduling add-ons of the scheduler to score nodes. You can customize the overall weight of the add-on and the weight of each resource to modify the influence in the node score. When using bin packing to calculate node scores, the scheduler considers extended resources such as CPUs, memory, and GPUs requested by pods, and calculates the scores based on the weights configured for each resource.

#### How It Works

A node is scored based on the overall weight of the bin packing add-on and the weight of each resource. Each type of resource requested by the to-be-scheduled pods is scored. Take CPUs as an example, the CPU score is calculated using the following formula:

**CPU.weight x (Requested + Used)/Allocatable**

A larger CPU weight leads to a higher score. A higher resource usage of a node leads to a higher node score. The same rule applies to memory and GPU resources. The parameters in the formula for scoring a resource are as follows:

- **CPU.weight:** customized CPU weight

- **Requested:** CPU resources requested by the pods to be scheduled
- **Used:** CPU resources that have been used on the current node
- **Allocatable:** total CPU resources available on the current node

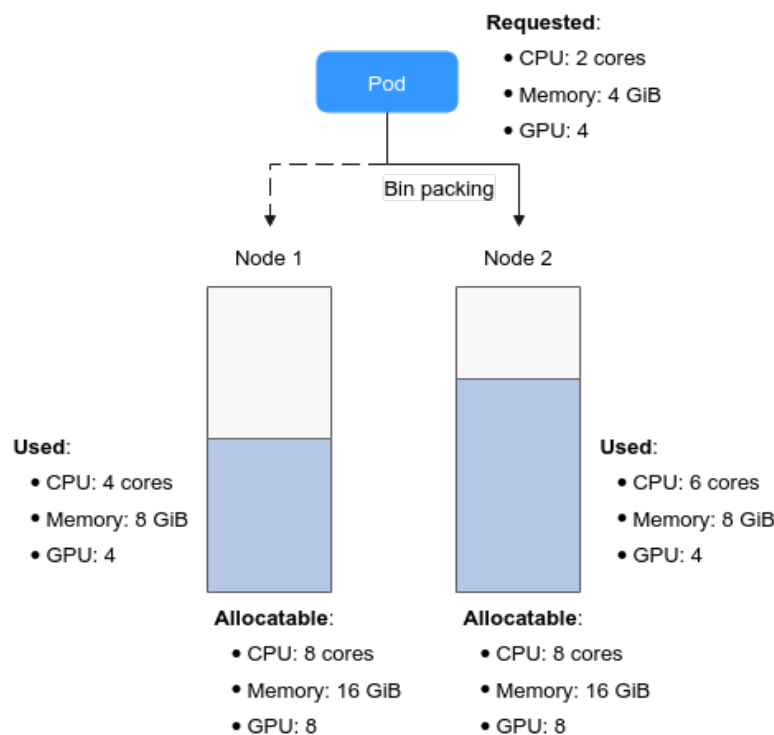
The bin packing add-on calculates the score of a node using the following formula:

$$\text{Binpack.weight} \times (\text{CPU.score} + \text{Memory.score} + \text{GPU.score}) / (\text{CPU.weight} + \text{Memory.weight} + \text{GPU.weight}) \times 100$$

A larger bin packing weight leads to a higher score. A larger resource weight leads to a greater influence in the node score. The parameters in the formula for scoring a node are as follows:

- **Binpack.weight:** bin packing weight
- **CPU.score:** calculated CPU score; **CPU.weight:** customized CPU weight
- **Memory.score:** calculated memory score; **Memory.weight:** customized memory weight
- **GPU.score:** calculated GPU score; **GPU.weight:** customized GPU weight

Figure 6-13 Bin packing example



In this figure, there are two nodes in the cluster. When pods need to be scheduled, the bin packing policy scores the two nodes separately.

For example, **CPU.weight** is set to **1**, **Memory.weight** is set to **1**, **GPU.weight** is set to **2**, and **binpack.weight** is set to **5** in a cluster.

1. Scoring for node 1

The score of each resource type is calculated using the following formulas:

- CPU:  $\text{CPU.weight} \times (\text{Requested} + \text{Used}) / \text{Allocatable} = 1 \times (2 + 4) / 8 = 0.75$
- Memory:  $\text{Memory.weight} \times (\text{Requested} + \text{Used}) / \text{Allocatable} = 1 \times (4 + 8) / 16 = 0.75$
- GPU:  $\text{GPU.weight} \times (\text{Requested} + \text{Used}) / \text{Allocatable} = 2 \times (4 + 4) / 8 = 2$

The total score of each node is calculated using the following formula:  
 $\text{Binpack.weight} \times (\text{CPU.score} + \text{Memory.score} + \text{GPU.score}) / (\text{CPU.weight} + \text{Memory.weight} + \text{GPU.weight}) \times 100$

If **binpack.weight** is set to **5**, the score of node 1 is calculated as follows:  $5 \times (0.75 + 0.75 + 2) / (1 + 1 + 2) \times 100 = 437.5$

2. Scoring for node 2

- CPU:  $\text{CPU.weight} \times (\text{Requested} + \text{Used}) / \text{Allocatable} = 1 \times (2 + 6) / 8 = 1$
- Memory:  $\text{Memory.weight} \times (\text{Requested} + \text{Used}) / \text{Allocatable} = 1 \times (4 + 8) / 16 = 0.75$
- GPU:  $\text{GPU.weight} \times (\text{Requested} + \text{Used}) / \text{Allocatable} = 2 \times (4 + 4) / 8 = 2$

Score of node 2:  $5 \times (1 + 0.75 + 2) / (1 + 1 + 2) \times 100 = 468.75$

The calculation results show that the score of node 2 is greater than that of node 1. According to the bin packing policy, new pods will be preferentially scheduled to node 2.

## Configuring Bin Packing

After Volcano is installed, bin packing takes effect by default. If the default configuration cannot meet your requirements, you can customize the weight of bin packing and the weight of each resource on the **Scheduling** page. To do so, perform the following operations:

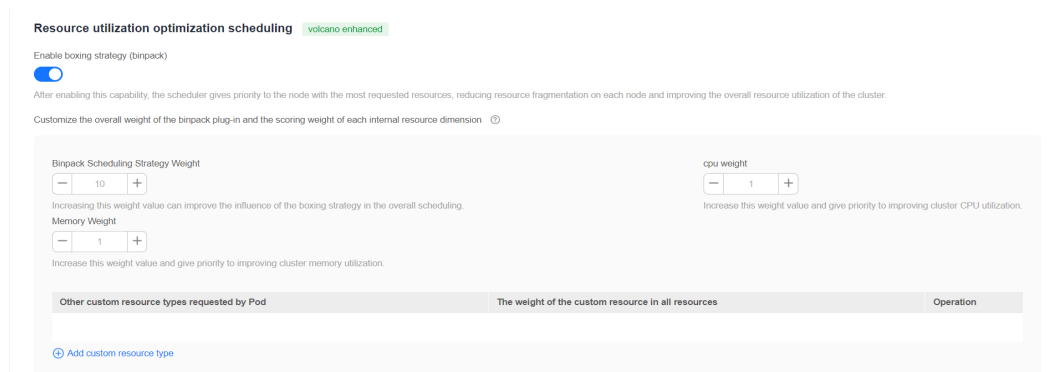
- Step 1** Log in to the CCE console.
- Step 2** Click the cluster name to access the cluster console. Choose **Settings** in the navigation pane. In the right pane, click the **Scheduling** tab.
- Step 3** In the **Resource utilization optimization scheduling** configuration, enable bin packing.

**Table 6-5** Bin packing weights

Item	Description	Default Value
Binpack Scheduling Strategy Weight	A larger value indicates a higher weight of the bin packing policy in overall scheduling.	10
CPU Weight	A larger value indicates a higher cluster CPU usage.	1
Memory Weight	A larger value indicates a higher cluster memory usage.	1

Item	Description	Default Value
Custom Resource Type	Other custom resource types requested by pods, for example, <b>nvidia.com/gpu</b> . A larger value indicates a higher usage of the specified cluster resource.	None

**Figure 6-14** Resource utilization optimization scheduling



**Step 4** Click **Confirm**.

----End

### 6.5.2.2 Descheduling

Scheduling in a cluster is the process of binding pending pods to nodes, and is performed by a component called kube-scheduler or Volcano Scheduler. The scheduler uses a series of algorithms to compute the optimal node for running pods. However, Kubernetes clusters are dynamic and their state changes over time. For example, if a node needs to be maintained, all pods on the node will be evicted to other nodes. After the maintenance is complete, the evicted pods will not automatically return back to the node because descheduling will not be triggered once a pod is bound to a node. Due to these changes, the load of a cluster may be unbalanced after the cluster runs for a period of time.

CCE has resolved this issue by using Volcano Scheduler to evict pods that do not comply with the configured policy so that pods can be rescheduled. In this way, the cluster load is balanced and resource fragmentation is minimized.

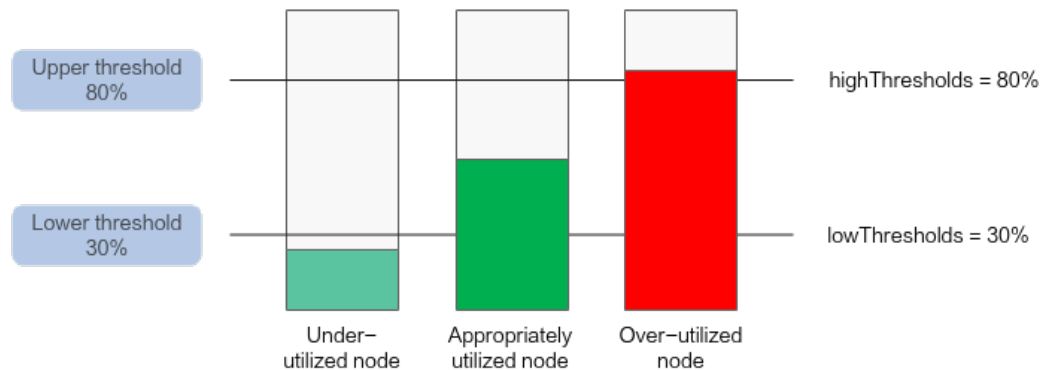
## Features

### Load-aware Descheduling

During Kubernetes cluster management, over-utilized nodes are due to high CPU or memory usage, which affects the stable running of pods on these nodes and increases the probability of node faults. To dynamically balance the resource usage between nodes in a cluster, a cluster resource view is required based on node monitoring metrics. During cluster management, real-time monitoring can be used to detect issues such as high resource usage on a node, node faults, and excessive number of pods on a node so that the system can take measures

promptly, for example, by migrating some pods from an over-utilized node to under-utilized nodes.

**Figure 6-15** Load-aware descheduling



When using this add-on, ensure the **highThresholds** value is greater than the **lowThresholds** value. Otherwise, the descheduler cannot work.

- **Appropriately utilized node:** a node whose resource usage is greater than or equal to 30% and less than or equal to 80%. The resource usage of appropriately utilized nodes is within the expected range.
- **Over-utilized node:** a node whose resource usage is higher than 80%. Some pods will be evicted from over-utilized nodes to reduce its resource usage to be less than or equal to 80%. The descheduler will schedule the evicted pods to under-utilized nodes.
- **Under-utilized node:** a node whose resource usage is lower than 30%.

### HighNodeUtilization

This policy finds nodes that are under-utilized and evicts pods from the nodes in the hope that these pods will be scheduled compactly into fewer nodes. This policy must be used with the bin packing policy of Volcano Scheduler or the MostAllocated policy of the kube-scheduler scheduler. Thresholds can be configured for CPU and memory.

### Prerequisites

- A cluster of v1.19.16 or later is available. For details, see [Buying a CCE Standard/Turbo Cluster](#).
- Volcano of v1.11.5 or later has been installed in the cluster. For details, see [Volcano Scheduler](#).

### Notes and Constraints

- Pods need to be rescheduled using a scheduler, and no scheduler can label pods or nodes. Therefore, an evicted pod might be rescheduled to the original node.
- Descheduling does not support anti-affinity between pods. An evicted pod is in anti-affinity relationship with other running pods. Therefore, the scheduler may still schedule the pod back to the node where the pod was evicted from.
- When configuring load-aware descheduling, you are required to enable load-aware scheduling on Volcano Scheduler. When configuring

HighNodeUtilization, you are required to enable bin packing on Volcano Scheduler.

## Configuring a Load-aware Descheduling Policy

When configuring a load-aware descheduling policy, do as follows to enable load-aware scheduling on Volcano Scheduler:

- Step 1** Log in to the CCE console and click the cluster name to access the cluster console. In the navigation pane, choose **Settings** and click the **Scheduling** tab on the right side of the page. Then, enable load-aware scheduling.
- Step 2** On the **Scheduling** tab page, select **Volcano scheduler**, find the expert mode, and click **Refresh**.

### Select Cluster Scheduler

Default cluster scheduler (default-scheduler) ⓘ

Kube-scheduler scheduler

**Volcano scheduler**

Volcano is compatible with kube-scheduler scheduling capabilities and provides incremental scheduling capabilities.

🌱 Expert mode: allows you to create custom scheduling policies by using a configuration file if the options available on the console do not meet your service needs. [Learn More](#) ⓘ [Refresh](#)

- Step 3** Configure a load-aware descheduling policy. The following shows a configuration example in JSON format for Volcano v1.11.21 or later:

```
{
  "colocation_enable": "",
  "default_scheduler_conf": {
    "actions": "allocate, backfill, preempt",
    "tiers": [
      {
        "plugins": [
          {
            "name": "priority"
          },
          {
            "enablePreemptable": false,
            "name": "gang"
          },
          {
            "name": "conformance"
          }
        ]
      },
      {
        "plugins": [
          {
            "enablePreemptable": false,
            "name": "drf"
          },
          {
            "name": "predicates"
          },
          {
            "name": "nodeorder"
          },
          {
            "name": "usage",
            "enablePredicate": true,
            "arguments": {
              "usage.weight": 5,
              "cpu.weight": 1,

```



```

        "memory.weight": 1,
        "thresholds": {
          "cpu": 80,
          "mem": 80
        }
      }
    ]
  },
  {
    "plugins": [
      {
        "name": "cce-gpu-topology-predicate"
      },
      {
        "name": "cce-gpu-topology-priority"
      },
      {
        "name": "cce-gpu"
      }
    ]
  },
  {
    "plugins": [
      {
        "name": "nodelocalvolume"
      },
      {
        "name": "nodeemptydirvolume"
      },
      {
        "name": "nodeCSIScheduling"
      },
      {
        "name": "networkresource"
      }
    ]
  }
],
},
"deschedulerPolicy": {
  "profiles": [
    {
      "name": "ProfileName",
      "pluginConfig": [
        {
          "args": {
            "ignorePvcPods": true,
            "nodeFit": true,
            "priorityThreshold": {
              "value": 100
            }
          },
          "name": "DefaultEvictor"
        },
        {
          "args": {
            "evictableNamespaces": {
              "exclude": ["kube-system"]
            },
            "metrics": {
              "type": "prometheus_adaptor"
            },
            "targetThresholds": {
              "cpu": 80,
              "memory": 85
            },
            "thresholds": {
              "cpu": 30,

```

```

        "memory": 30
      }
    },
    "name": "LoadAware"
  }
],
"plugins": {
  "balance": {
    "enabled": ["LoadAware"]
  }
}
}
]
},
"descheduler_enable": "true",
"deschedulingInterval": "10m"
}

```

**Table 6-6** Key parameters of a cluster descheduling policy

Parameter	Description
descheduler_enable	Whether to enable a cluster descheduling policy. <ul style="list-style-type: none"> <li>• <b>true</b>: The cluster descheduling policy is enabled.</li> <li>• <b>false</b>: The cluster descheduling policy is disabled.</li> </ul>
deschedulingInterval	Descheduling period.
deschedulerPolicy	Cluster descheduling policy. For details, see <a href="#">Table 6-7</a> .

**Table 6-7** deschedulerPolicy parameters

Parameter	Description
profiles. [].plugins.balance.enabled.[]	Descheduling policy for a cluster. <b>LoadAware</b> : a load-aware descheduling policy is used.
profiles. [].pluginConfig. [].name	Configuration of a load-aware descheduling policy. Options: <ul style="list-style-type: none"> <li>• <b>DefaultEvictor</b>: default eviction policy</li> <li>• <b>LoadAware</b>: a load-aware descheduling policy</li> </ul>

Parameter	Description
<p>profiles. [].pluginConfig. [].args</p>	<p>Descheduling policy configuration of a cluster.</p> <ul style="list-style-type: none"> <li>● Configurations for the <b>DefaultEvictor</b> policy: <ul style="list-style-type: none"> <li>– <b>ignorePvcPods</b>: whether PVC pods should be ignored or evicted. Value <b>true</b> indicates that the pods are ignored, and value <b>false</b> indicates that the pods are evicted. This configuration does not differentiate PVC types (local PVs, SFS, or EVS).</li> <li>– <b>nodeFit</b>: whether to consider the existing scheduling configurations such as node affinity and taint on the node during descheduling. Value <b>true</b> indicates that the existing scheduling configurations will be considered, and value <b>false</b> indicates that those will be ignored.</li> <li>– <b>priorityThreshold</b>: priority setting. If the priority of a pod is greater than or equal to the value of this parameter, the pod will not be evicted. Example:  <pre>{   "value": 100 }</pre> </li> </ul> </li> <li>● Configurations for the <b>LoadAware</b> policy: <ul style="list-style-type: none"> <li>– <b>evictableNamespaces</b>: namespaces where the eviction policy takes effect. The default value is the namespaces other than kube-system. Example:  <pre>{   "exclude": ["kube-system"] }</pre> </li> <li>– <b>metrics</b>: how monitoring data is obtained. Either the Custom Metrics API (prometheus_adaptor) or Prometheus can be used. For Volcano 1.11.17 and later versions, use Custom Metrics API to obtain monitoring data. The following is an example:  <pre>{   "type": "prometheus_adaptor" }</pre>  For Volcano 1.11.5 to 1.11.16, use Prometheus to obtain monitoring data. You need to enter the IP address of the Prometheus server. The following is an example:  <pre>{   "address": "http://10.247.119.103:9090",   "type": "prometheus" }</pre> </li> <li>– <b>targetThresholds</b>: threshold for evicting pods from a node. When the CPU or memory usage of a node is greater than the threshold, the pods on the node will be evicted. Example:  <pre>{   "cpu": 60,</pre> </li> </ul> </li> </ul>

Parameter	Description
	<pre>"memory": 65 }</pre> <ul style="list-style-type: none"> <li>- <b>thresholds</b>: threshold for a node to run pods. When the CPU or memory usage of a node is less than the threshold, the node allows evicted pods to run. Example: <pre>{   "cpu": 30,   "memory": 30 }</pre> </li> </ul>

**Step 4** Click **OK**.

----End

## Configuring a HighNodeUtilization Policy

When configuring a HighNodeUtilization policy, do as follows to enable the bin packing policy on Volcano Scheduler:

**Step 1** Log in to the CCE console and click the cluster name to access the cluster console. In the navigation pane, choose **Settings** and click the **Scheduling** tab on the right side of the page. Then, enable bin packing. For details, see [Bin Packing](#).

**Step 2** On the **Scheduling** tab page, select **Volcano scheduler**, find the expert mode, and click **Refresh**.

### Select Cluster Scheduler

Default cluster scheduler (default-scheduler) ⓘ

Kube-scheduler scheduler

**Volcano scheduler**

Volcano is compatible with kube-scheduler scheduling capabilities and provides incremental scheduling capabilities.

🌱 Expert mode: allows you to create custom scheduling policies by using a configuration file if the options available on the console do not meet your service needs. [Learn More](#) ⓘ **Refresh**

**Step 3** Configure a resource defragmentation policy. The following shows a configuration example in JSON format:

```
{
  "colocation_enable": "",
  "default_scheduler_conf": {
    "actions": "allocate, backfill, preempt",
    "tiers": [
      {
        "plugins": [
          {
            "name": "priority"
          },
          {
            "enablePreemptable": false,
            "name": "gang"
          },
          {
            "name": "conformance"
          },
          {
            "arguments": {
```

```

        "binpack.weight": 5
      },
      "name": "binpack"
    }
  ]
},
{
  "plugins": [
    {
      "enablePreemptable": false,
      "name": "drf"
    },
    {
      "name": "predicates"
    },
    {
      "name": "nodeorder"
    }
  ]
},
{
  "plugins": [
    {
      "name": "cce-gpu-topology-predicate"
    },
    {
      "name": "cce-gpu-topology-priority"
    },
    {
      "name": "cce-gpu"
    }
  ]
},
{
  "plugins": [
    {
      "name": "nodelocalvolume"
    },
    {
      "name": "nodeemptydirvolume"
    },
    {
      "name": "nodeCSIscheduling"
    },
    {
      "name": "networkresource"
    }
  ]
}
],
"deschedulerPolicy": {
  "profiles": [
    {
      "name": "ProfileName",
      "pluginConfig": [
        {
          "args": {
            "ignorePvcPods": true,
            "nodeFit": true,
            "priorityThreshold": {
              "value": 100
            }
          }
        },
        {
          "name": "DefaultEvictor"
        }
      ],
      "args": {
        "evictableNamespaces": {

```

```

        "exclude": ["kube-system"]
      },
      "thresholds": {
        "cpu": 25,
        "memory": 25
      }
    },
    "name": "HighNodeUtilization"
  }
],
"plugins": {
  "balance": {
    "enabled": ["HighNodeUtilization"]
  }
}
]
},
"descheduler_enable": "true",
"deschedulingInterval": "10m"
}

```

**Table 6-8** Key parameters of a cluster descheduling policy

Parameter	Description
descheduler_enable	Whether to enable a cluster descheduling policy. <ul style="list-style-type: none"> <li>• <b>true</b>: The cluster descheduling policy is enabled.</li> <li>• <b>false</b>: The cluster descheduling policy is disabled.</li> </ul>
deschedulingInterval	Descheduling period.
deschedulerPolicy	Cluster descheduling policy. For details, see <a href="#">Table 6-9</a> .

**Table 6-9** deschedulerPolicy parameters

Parameter	Description
profiles. [].plugins.balance.enabled.[]	Descheduling policy for a cluster. <b>HighNodeUtilization</b> : the policy for minimizing CPU and memory fragments is used.
profiles. [].pluginConfig. [].name	Configuration of a load-aware descheduling policy. Options: <ul style="list-style-type: none"> <li>• <b>DefaultEvictor</b>: default eviction policy</li> <li>• <b>HighNodeUtilization</b>: policy for minimizing CPU and memory fragments</li> </ul>

Parameter	Description
<p>profiles. [] .pluginConfig. [] .args</p>	<p>Descheduling policy configuration of a cluster.</p> <ul style="list-style-type: none"> <li>● Configurations for the <b>DefaultEvictor</b> policy: <ul style="list-style-type: none"> <li>- <b>ignorePvcPods</b>: whether PVC pods should be ignored or evicted. Value <b>true</b> indicates that the pods are ignored, and value <b>false</b> indicates that the pods are evicted. This configuration does not differentiate PVC types (local PVs, SFS, or EVS).</li> <li>- <b>nodeFit</b>: whether to consider the existing scheduling configurations such as node affinity and taint on the node during descheduling. Value <b>true</b> indicates that the existing scheduling configurations will be considered, and value <b>false</b> indicates that those will be ignored.</li> <li>- <b>priorityThreshold</b>: priority setting. If the priority of a pod is greater than or equal to the value of this parameter, the pod will not be evicted. Example: <pre data-bbox="774 907 1428 985" style="background-color: #f0f0f0;">{   "value": 100 }</pre> </li> </ul> </li> <li>● Configurations for the <b>HighNodeUtilization</b> policy: <ul style="list-style-type: none"> <li>- <b>evictableNamespaces</b>: namespaces where the eviction policy takes effect. The default value is the namespaces other than kube-system. Example: <pre data-bbox="774 1176 1428 1254" style="background-color: #f0f0f0;">{   "exclude": ["kube-system"] }</pre> </li> <li>- <b>thresholds</b>: threshold for evicting pods from a node. When the CPU or memory usage of a node is less than the threshold, the pods on the node will be evicted. Example: <pre data-bbox="774 1400 1428 1489" style="background-color: #f0f0f0;">{   "cpu": 25,   "memory": 25 }</pre> </li> </ul> </li> </ul>

**Step 4** Click **OK**.

----End

## Use Cases

### HighNodeUtilization

1. Check the nodes in a cluster. It is found that some nodes are under-utilized.

192.168.44.152 (Private)	1 / 40	0.51%	0%
192.168.54.65 (Private)	6 / 40	26.53%	33.93%

2. Edit the Volcano parameters to enable the descheduler and set the CPU and memory usage thresholds to 25. When the CPU and memory usage of a node is less than 25%, pods on the node will be evicted.

```

Advanced Settings
{
  "exclude": [
    "kube-system"
  ],
  "thresholds": {
    "cpu": 25,
    "memory": 25
  },
  "name": "HighNodeUtilization"
},
"plugins": {
  "balance": [
    "enabled": ["HighNodeUtilization"]
  ]
}
},
"descheduler_enable": "true",
"deschedulingInterval": "10m"
}

```

3. After the policy takes effect, pods on the node with IP address 192.168.44.152 will be migrated to the node with IP address 192.168.54.65 for minimized resource fragments.

192.168.44.152 (Private)	0 / 40	0%	0%
192.168.54.65 (Private)	7 / 40	27.04%	33.93%

## Common Issues

If an input parameter is incorrect, for example, the entered value is beyond the accepted value range or in an incorrect format, an event will be generated. In this case, modify the parameter setting as prompted.

Kubern...	Event ...	Occurr...	Event Name	Kubernetes Event	First Occurred	Last Occurred
sigs.k8s.io.des...	Alarm	1	Abnormal	descheduler run err due to parameter e...	Nov 02, 2023 11:44:5...	Nov 03, 2023 10:20:0...

### 6.5.2.3 Node Pool Affinity

In scenarios such as node pool replacement and rolling node upgrade, an old resource pool needs to be replaced with a new one. To prevent the node pool replacement from affecting services, enable soft affinity to schedule service pods to the new node pool. Soft affinity scheduling tries to schedule newly created pods or rescheduled pods to the new node pool. If the pods cannot be scheduled to the new node pool, for example, due to insufficient resources, the pods can also be scheduled to the old node pool. Since a node pool replacement should not affect



services, the node affinity configuration is not declared in service workloads. Use soft affinity in cluster scheduling to schedule pods to new node pools when a pool replacement is triggered.

Volcano aims to soft schedule service pods to specified nodes when node soft affinity is not configured on service workloads.

## Scheduling Priority

Soft affinity scheduling of a node pool is implemented based on labels in the node pool. Each node in the node pool is scored to select the optimal one for pod scheduling.

The rule is to schedule pods to nodes with specified labels as far as possible.

The scoring formula is as follows:

Node score = Weight x MaxNodeScore x haveLabel

Parameters:

- **Weight:** weight of the soft affinity add-on in the node pool.
- **MaxNodeScore:** maximum score (100) of a node.
- **haveLabel:** whether the labels configured in the add-on are available on a node. If yes, the value is **1**. If no, the value is **0**.

## Prerequisites

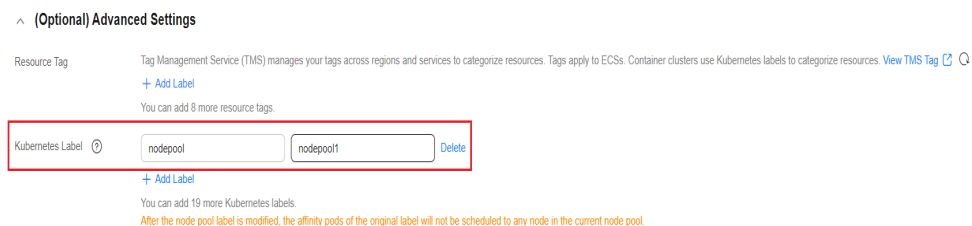
- A cluster of v1.19.16 or later is available. For details, see [Buying a CCE Standard/Turbo Cluster](#).
- Volcano of v1.11.5 or later has been installed in the cluster. For details, see [Volcano Scheduler](#).

## Configuring Soft Affinity Scheduling for Volcano Node Pools

**Step 1** Configure labels for affinity scheduling in the node pool.

1. Log in to the CCE console.
2. Click the cluster name to access the cluster console. Choose **Nodes** in the navigation pane. In the right pane, click the **Node Pools** tab.
3. Click **Update** of the target node pool. On the page that is displayed, configure labels in the **Kubernetes Label** area.

The following is an example.



**Step 2** On the **Scheduling** tab page, select **Volcano scheduler**, find the expert mode, and click **Refresh**.


### Select Cluster Scheduler

Default cluster scheduler (default-scheduler) [?](#)

Kube-scheduler scheduler

**Volcano scheduler**

Volcano is compatible with kube-scheduler scheduling capabilities and provides incremental scheduling capabilities.

 Expert mode: allows you to create custom scheduling policies by using a configuration file if the options available on the console do not meet your service needs. [Learn More](#) [Refresh](#)

**Step 3** Configure Volcano scheduler parameters. The following shows a configuration example in JSON format:

```

...
"default_scheduler_conf": {
  "actions": "allocate, backfill, preempt",
  "tiers": [
    {
      "plugins": [
        {
          "name": "priority"
        },
        {
          "name": "gang"
        },
        {
          "name": "conformance"
        }
      ]
    },
    {
      "plugins": [
        {
          "name": "drf"
        },
        {
          "name": "predicates"
        },
        {
          "name": "nodeorder"
        }
      ]
    },
    {
      "plugins": [
        {
          "name": "cce-gpu-topology-predicate"
        },
        {
          "name": "cce-gpu-topology-priority"
        },
        {
          "name": "cce-gpu"
        },
        {
          // Enable node pool affinity scheduling.
          "name": "nodepoolaffinity",
          // Configure the affinity scheduling weight and labels of the node pool.
          "arguments": {
            "nodepoolaffinity.weight": 10000,
            "nodepoolaffinity.label": "nodepool1=nodepool1"
          }
        }
      ]
    }
  ],
  {
    "plugins": [

```

```
        "name": "nodelocalvolume"
      },
      {
        "name": "nodeemptydirvolume"
      },
      {
        "name": "nodeCSIscheduling"
      },
      {
        "name": "networkresource"
      }
    ]
  }
},
],
...
```

**Step 4** Click **OK**.

----End

### 6.5.2.4 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.

#### Prerequisites

- A cluster of v1.21 or later is available. For details, see [Buying a CCE Standard/Turbo Cluster](#).
- The Volcano add-on of v1.11.14 or later has been installed. For details, see [Volcano Scheduler](#).
- The Cloud Native Cluster Monitoring add-on (kube-prometheus-stack) has been installed and it runs in local data storage mode. For details, see [Cloud Native Cluster Monitoring](#).

#### Features

The native Kubernetes scheduler schedules resources only based on requested resources. However, the actual resource usage of a pod differs greatly from the requested or limited value of the requested resources, which is the cause of cluster load imbalance.

1. The actual resource usage of certain nodes in a cluster is far lower than the resource allocation rate, but no more pods are scheduled onto the nodes, leading to resource waste.
2. The scheduler failed to detect that some nodes in the cluster are overloaded, which could significantly affect the stability of the service.

Volcano resolves the preceding issues based on actual loads. If there are plenty of resources, pods are preferentially scheduled to nodes with the lightest load to balance the load on each node in the cluster.

The status, workload traffic, and requests of a cluster change dynamically, and the resource usage of nodes changes in real time. To prevent extreme load imbalance in a cluster after pod scheduling, Volcano provides load-aware hotspot

descheduling for the optimal load balancing of cluster nodes. For details about hotspot descheduling, see [Descheduling](#).

## How It Works

Load-aware scheduling is implemented using both Volcano and the CCE cloud native monitoring add-on (kube-prometheus-stack). After load-aware scheduling is enabled, metrics such as CPU and memory loads are defined by following Prometheus adapter rules. Then, the kube-prometheus-stack add-on collects and saves the actual CPU and memory loads of each node based on the defined metric rules. Volcano scores and sorts nodes based on the metric values provided by the kube-prometheus-stack add-on and preferentially schedules pods to the node with the lightest load.

Load-aware scheduling scores each node using the weighted average of the CPU and memory metrics as well as the load-aware scheduling policy and preferentially selects the node with the highest score for scheduling. You can customize the weights of the CPU, memory, and load-aware scheduling policy on the **Scheduling** tab by choosing **Settings** in the navigation pane of the target cluster.

The formula for scoring a node is as follows:  $\text{Weight of the load-aware scheduling policy} \times [(1 - \text{CPU usage}) \times \text{CPU weight} + (1 - \text{Memory usage}) \times \text{Memory weight}] / (\text{CPU weight} + \text{Memory weight})$

- CPU usage: average CPU usage of all nodes in the target cluster in the last 10 minutes (The collection frequency can be modified in the Prometheus adapter rule.)
- Memory usage: average memory usage of all nodes in the target cluster in the last 10 minutes

## Configuring Load-aware Scheduling

**Step 1** Obtain resource metrics by calling the metrics API and add custom metric collection rules.

After the kube-prometheus-stack add-on is installed, enable the function of automatically obtaining resource metrics through the metrics API. For details, see [Providing Resource Metrics Through the Metrics API](#).

Add custom metric collection rules. For details, see [Creating an HPA Policy Using Custom Metrics](#). In the following example rules, the rules in red are new ones and those in black are existing ones:

```
rules:
- seriesQuery: '{__name__=~"node_cpu_seconds_total"}'
  resources:
    overrides:
      instance:
        resource: node
    name:
      matches: node_cpu_seconds_total
      as: node_cpu_usage_avg
    metricsQuery: avg_over_time((1 - avg (irate(<<.Series>>{mode="idle"}[5m]))) by (instance))[10m:30s])
- seriesQuery: '{__name__=~"node_memory_MemTotal_bytes"}'
  resources:
    overrides:
      instance:
        resource: node
```

```
name:
  matches: node_memory_MemTotal_bytes
  as: node_memory_usage_avg
  metricsQuery: avg_over_time(((1-node_memory_MemAvailable_bytes/<<.Series>>))[10m:30s])
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
window: 1m
```

- **Rules for collecting the average CPU usage**

- **node\_cpu\_usage\_avg**: average CPU usage of nodes. The name of this metric cannot be changed.
- **metricsQuery: avg\_over\_time((1 - avg (irate(<<.Series>>{mode="idle"}[5m])) by (instance))[10m:30s]):** statement for obtaining nodes' average CPU usage.

**metricsQuery** indicates to obtain the average CPU usage of all nodes in the target cluster in the last 10 minutes. To change the period, for example, to the last 5 or 30 minutes, change **10m** in red to **5m** or **30m**.

- **Rules for collecting the average memory usage**

- **node\_memory\_usage\_avg**: average memory usage of nodes. The name of this metric cannot be changed.
- **metricsQuery: avg\_over\_time(((1- node\_memory\_MemAvailable\_bytes/<<.Series>>))[10m:30s]):** statement for obtaining nodes' average memory usage.

**metricsQuery** indicates to obtain the average memory usage of all nodes in the target cluster in the last 10 minutes. To change the period, for example, to the last 5 or 30 minutes, change **10m** in red to **5m** or **30m**.

## Step 2 Enable load-aware scheduling.

After Volcano is installed, you can enable or disable load-aware scheduling on the **Scheduling** page by choose **Settings** in the navigation pane. This function is disabled by default.

1. Log in to the CCE console.

2. Click the cluster name to access the cluster console. Choose **Settings** in the navigation pane. In the right pane, click the **Scheduling** tab.
3. In the **Resource utilization optimization scheduling** area, modify the load-aware scheduling settings.

 **NOTE**

For optimal load-aware scheduling, disable bin packing because this policy preferentially schedules pods to the node with the maximal resources allocated based on pods' requested resources. This affects load-aware scheduling to some extent. For details about the combination of multiple policies, see [Configuration Cases for Resource Usage-based Scheduling](#).

Parameter	Description	Default Value
Load-Aware Scheduling Policy Weight	A larger value indicates a higher weight of the load-aware policy in overall scheduling.	5
CPU Weight	A larger value indicates CPU resources will be preferentially balanced.	1
Memory Weight	A larger value indicates memory resources will be preferentially balanced.	1
Actual load threshold effective mode	<ul style="list-style-type: none"> <li>- Soft constraint: When the actual CPU or memory load of a node reaches the threshold, new tasks will be preferentially allocated to underutilized nodes, but this node can still be scheduled.</li> <li>- Hard constraint: When the actual CPU or memory load of a node reaches the threshold, no new tasks can be scheduled to this node.</li> </ul>	Hard constraint
Actual CPU Load Threshold	When a node's CPU usage goes beyond this threshold, pods will be preferentially or forcibly scheduled to other nodes based on how the load threshold takes effect.	80
Actual Memory Load Threshold	When a node's memory usage goes beyond this threshold, pods will be preferentially or forcibly scheduled to other nodes based on how the load threshold takes effect.	80

----End

### 6.5.2.5 Configuration Cases for Resource Usage-based Scheduling

#### Overview

Volcano scheduling involves node filtering and scoring, which is used to filter the nodes meeting scheduling conditions and score the filtered nodes to find the one

with the highest score for scheduling. Volcano provides multiple scheduling policies for node scoring. The weight of each scheduling policy can be adjusted based on service scenarios to enhance or reduce the impact of the policy on node scoring.

## Scheduling Policies for Node Scoring

The following table lists the scheduling policies supported by Volcano for node scoring.

Scheduling Policy	Parameter	Description	Reference
Bin packing	binpack.weight	After this function is enabled, the parameter defaults to <b>10</b> .	<a href="#">Bin Packing</a>
kube-scheduler node sorting (nodeorder)	nodeaffinity.weight	Pods are scheduled based on node affinity. The parameter defaults to <b>2</b> .	By default, this function is enabled.
	podaffinity.weight	Pods are scheduled based on pod affinity. The parameter defaults to <b>2</b> .	
	leastrequested.weight	Pods are scheduled to the node with the least requested resources. The parameter defaults to <b>1</b> .	
	balancedresource.weight	Pods are scheduled to the node with balanced resource allocation. The parameter defaults to <b>1</b> .	
	mostrequested.weight	Pods are scheduled to the node with the most requested resources. The parameter defaults to <b>0</b> .	
	tainttoleration.weight	Pods are scheduled to the node with a high taint tolerance. The parameter defaults to <b>3</b> .	
	imagelocality.weight	Pods are scheduled to the node where the required images exist. The parameter defaults to <b>1</b> .	
	selectorspread.weight	Pods are evenly scheduled to different nodes. The parameter defaults to <b>0</b> .	

Scheduling Policy	Parameter	Description	Reference
	podtopologyspread.weight	Pods are scheduled based on the pod topology. The parameter defaults to <b>2</b> .	
NUMA affinity scheduling (numa-aware)	weight	After this function is enabled, the parameter defaults to <b>1</b> .	<a href="#">NUMA Affinity Scheduling</a>
Load-aware scheduling (usage)	weight	After this function is enabled, the parameter defaults to <b>5</b> .	<a href="#">Load-aware Scheduling</a>
Node pool affinity scheduling (nodepoolaffinity)	nodepoolaffinity.weight	After this function is enabled, the parameter defaults to <b>10000</b> .	<a href="#">Node Pool Affinity</a>

## How Can I Improve Cluster Resource Utilization by Reducing Node Resource Fragments?

There are both heavy- and low-resource jobs running in a cluster. It is hoped that the low-resource job preferentially uses resource fragments on each node so that idle nodes can be allocated to the high-resource job. This prevents job scheduling failures caused by insufficient node resources.

To resolve the preceding issue, enable bin packing and use the default policy weight **10**. For details, see [Bin Packing](#).

Recommended configurations:

- To preferentially reduce CPU fragments in the cluster, increase the CPU weight to **5** and retain the memory weight to **1** in the bin packing policy.
- To preferentially reduce memory fragments in the cluster, increase the memory weight to **5** and retain the CPU weight to **1** in the bin packing policy.
- To preferentially reduce GPU fragments in the cluster, customize the GPU resource type, set the GPU weight to **10**, and retain both the CPU weight and memory weight to **1** in the bin packing policy.

## How Can I Balance the Actual CPU and Memory Loads on Nodes?

When a workload is running, the CPU and memory resources used may differ greatly from what was initially requested. To avoid any issues caused by overloading a single node, it is hoped that the scheduler preferentially schedules pods to the nodes with lighter loads based on nodes' CPU and memory usage in the cluster. This balances loads between nodes and ensures the stability of both applications and nodes.



### Configuration case 1

1. Enable load-aware scheduling and use the default policy weight **5**. For details, see [Load-aware Scheduling](#).
2. Disable bin packing. For details, see [Bin Packing](#).

Recommended configurations:

- To preferentially balance the CPU load of each node, increase the CPU weight of the policy to **5** and retain the memory weight to **1**.
- To preferentially balance the memory load of each node, increase the memory weight of the policy to **5** and retain the CPU weight to **1**.
- To use both the CPU and memory usage and the CPU and memory thresholds, do as follows:
  - Hard constraints:
    - After the CPU usage of a node exceeds its CPU threshold, do not schedule new loads to the node.
    - After the memory usage of a node exceeds its memory threshold, do not schedule new loads to the node.
  - Soft constraints:
    - After the CPU usage of a node exceeds its CPU threshold, do not schedule new loads to the node as far as possible.
    - After the memory usage of a node exceeds its memory threshold, do not schedule new loads to the node as far as possible.
  - To balance the load of each node in a cluster while maximizing the cluster resource utilization, enable soft constraints for the CPU and memory thresholds and use the default value **80** for both the CPU and memory thresholds.
  - To ensure workload stability and reduce the CPU and memory usage of heavy-load nodes, enable hard constraints for the CPU and memory thresholds and set the CPU and memory thresholds a value ranging from 60 to 80.

### Configuration case 2

The status, workload traffic, and requests of a cluster change dynamically, and the resource usage of nodes changes in real time. Node imbalancing may recur after pod scheduling. Use both load-aware scheduling and descheduling for the optimal load balancing of cluster nodes. For details about hotspot descheduling, see [Descheduling](#).

1. Enable load-aware scheduling and use the default policy weight **5**. For details, see [Load-aware Scheduling](#).
2. Enable descheduling and configure the load-aware descheduling policy. For details, see [Descheduling](#).
3. Disable bin packing. For details, see [Bin Packing](#).

Recommended configurations:

- Configure the load-aware descheduling policy as follows:

- **targetThreshold** for evicting pods from heavy-load nodes: Set the CPU threshold to **75** and memory threshold to **70**.
- **thresholds** for accepting pods on light-load nodes: Set both the CPU and memory thresholds to **30**.
- Ensure the actual CPU or memory threshold is between the CPU or memory threshold of the heaviest-load node and that of the lightest-load node.
  - Actual CPU threshold: **65**
  - Actual memory threshold: **60**

## 6.5.3 Priority-based Scheduling

### 6.5.3.1 Priority-based Scheduling

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.

#### Prerequisites

- A cluster of v1.19 or later is available. For details, see [Buying a CCE Standard/Turbo Cluster](#).
- The Volcano add-on has been installed. For details, see [Volcano Scheduler](#).

#### Overview

The services running in a cluster are diversified, including core services, non-core services, online services, and offline services. You can configure priorities for different services based on service importance and SLA requirements. For example, configure a high priority for core services and online services so that such services preferentially obtain cluster resources.

**Table 6-10** lists the priority-based scheduling supported by CCE clusters.

**Table 6-10** Priority-based scheduling

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.

## Configuring Priority-based Scheduling Policies

- Step 1** Log in to the CCE console.
- Step 2** Click the cluster name to access the cluster console. Choose **Settings** in the navigation pane. In the right pane, click the **Scheduling** tab.
- Step 3** In the **Business priority scheduling** area, configure priority-based scheduling.

- **Scheduling based on priority:** 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.

**Step 4** After the configuration, you can use **PriorityClasses** to schedule the pods of workloads or Volcano jobs based priorities.

1. Create one or more **PriorityClasses**.

```
apiVersion: scheduling.k8s.io/v1
kind: PriorityClass
metadata:
  name: high-priority
value: 1000000
globalDefault: false
description: ""
```

2. Create a workload or Volcano job and specify its PriorityClass name.

– **Workload**

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: high-test
labels:
  app: high-test
spec:
  replicas: 5
  selector:
    matchLabels:
      app: test
  template:
    metadata:
      labels:
        app: test
    spec:
      priorityClassName: high-priority
      schedulerName: volcano
      containers:
      - name: test
        image: busybox
        imagePullPolicy: IfNotPresent
        command: ['sh', '-c', 'echo "Hello, Kubernetes!" && sleep 3600']
      resources:
        requests:
          cpu: 500m
        limits:
          cpu: 500m
```

– **Volcano job**

```
apiVersion: batch.volcano.sh/v1alpha1
kind: Job
metadata:
  name: vcjob
spec:
  schedulerName: volcano
  minAvailable: 4
  priorityClassName: high-priority
  tasks:
  - replicas: 4
    name: "test"
    template:
      spec:
        containers:
        - image: alpine
          command: ["/bin/sh", "-c", "sleep 1000"]
          imagePullPolicy: IfNotPresent
          name: running
        resources:
```

```
requests:
  cpu: "1"
restartPolicy: OnFailure
```

----End

## Example of Priority-based Scheduling

For example, there are two idle nodes and several workloads with three priorities (high-priority, medium-priority, and low-priority). Run the high-priority workload to exhaust all cluster resources, and issue the medium-priority and low-priority workloads. Then, the two types of workloads are pending due to insufficient resources. When the high-priority workload ends, the pods of the medium-priority workload will be scheduled ahead of the pods of the low-priority workload according to the priority-based scheduling setting.

**Step 1** Add three **PriorityClasses** (high-priority, med-priority, and low-priority) in **priority.yaml**.

Example configuration of **priority.yaml**:

```
apiVersion: scheduling.k8s.io/v1
kind: PriorityClass
metadata:
  name: high-priority
  value: 100
globalDefault: false
description: "This priority class should be used for volcano job only."
---
apiVersion: scheduling.k8s.io/v1
kind: PriorityClass
metadata:
  name: med-priority
  value: 50
globalDefault: false
description: "This priority class should be used for volcano job only."
---
apiVersion: scheduling.k8s.io/v1
kind: PriorityClass
metadata:
  name: low-priority
  value: 10
globalDefault: false
description: "This priority class should be used for volcano job only."
```

Create PriorityClasses.

```
kubectl apply -f priority.yaml
```

**Step 2** Check PriorityClasses.

```
kubectl get PriorityClass
```

Command output:

NAME	VALUE	GLOBAL-DEFAULT	AGE
high-priority	100	false	97s
low-priority	10	false	97s
med-priority	50	false	97s
system-cluster-critical	2000000000	false	6d6h
system-node-critical	2000001000	false	6d6h

**Step 3** Create a high-priority workload named **high-priority-job** to exhaust all cluster resources.

**high-priority-job.yaml**

```
apiVersion: batch.volcano.sh/v1alpha1
kind: Job
```

```

metadata:
  name: priority-high
spec:
  schedulerName: volcano
  minAvailable: 4
  priorityClassName: high-priority
  tasks:
    - replicas: 4
      name: "test"
      template:
        spec:
          containers:
            - image: alpine
              command: ["/bin/sh", "-c", "sleep 1000"]
              imagePullPolicy: IfNotPresent
              name: running
          resources:
            requests:
              cpu: "1"
          restartPolicy: OnFailure
  
```

Run the following command to issue the job:

```
kubectl apply -f high_priority_job.yaml
```

Run the **kubectl get pod** command to check pod statuses:

NAME	READY	STATUS	RESTARTS	AGE
priority-high-test-0	1/1	Running	0	3s
priority-high-test-1	1/1	Running	0	3s
priority-high-test-2	1/1	Running	0	3s
priority-high-test-3	1/1	Running	0	3s

The command output shows that all cluster resources have been used up.

**Step 4** Create a medium-priority workload **med-priority-job** and a low-priority workload **low-priority-job**.

#### med-priority-job.yaml

```

apiVersion: batch.volcano.sh/v1alpha1
kind: Job
metadata:
  name: priority-medium
spec:
  schedulerName: volcano
  minAvailable: 4
  priorityClassName: med-priority
  tasks:
    - replicas: 4
      name: "test"
      template:
        spec:
          containers:
            - image: alpine
              command: ["/bin/sh", "-c", "sleep 1000"]
              imagePullPolicy: IfNotPresent
              name: running
          resources:
            requests:
              cpu: "1"
          restartPolicy: OnFailure
  
```

#### low-priority-job.yaml

```

apiVersion: batch.volcano.sh/v1alpha1
kind: Job
metadata:
  name: priority-low
  
```

```
spec:
  schedulerName: volcano
  minAvailable: 4
  priorityClassName: low-priority
  tasks:
  - replicas: 4
    name: "test"
    template:
      spec:
        containers:
        - image: alpine
          command: ["/bin/sh", "-c", "sleep 1000"]
          imagePullPolicy: IfNotPresent
          name: running
          resources:
            requests:
              cpu: "1"
            restartPolicy: OnFailure
```

Run the following commands to issue the jobs:

```
kubectl apply -f med_priority_job.yaml
kubectl apply -f low_priority_job.yaml
```

Run the **kubectl get pod** command to check the statuses of the pods for the newly created workloads. The command output shows that the pods are pending due to insufficient resources:

NAME	READY	STATUS	RESTARTS	AGE
priority-high-test-0	1/1	Running	0	3m29s
priority-high-test-1	1/1	Running	0	3m29s
priority-high-test-2	1/1	Running	0	3m29s
priority-high-test-3	1/1	Running	0	3m29s
priority-low-test-0	0/1	Pending	0	2m26s
priority-low-test-1	0/1	Pending	0	2m26s
priority-low-test-2	0/1	Pending	0	2m26s
priority-low-test-3	0/1	Pending	0	2m26s
priority-medium-test-0	0/1	Pending	0	2m36s
priority-medium-test-1	0/1	Pending	0	2m36s
priority-medium-test-2	0/1	Pending	0	2m36s
priority-medium-test-3	0/1	Pending	0	2m36s

**Step 5** Delete the **high\_priority\_job** workload to release resources and check whether the pods of the **med-priority-job** workload will be preferentially scheduled.

Run the **kubectl delete -f high\_priority\_job.yaml** command to release cluster resources and check pod scheduling.

NAME	READY	STATUS	RESTARTS	AGE
priority-low-test-0	0/1	Pending	0	5m18s
priority-low-test-1	0/1	Pending	0	5m18s
priority-low-test-2	0/1	Pending	0	5m18s
priority-low-test-3	0/1	Pending	0	5m18s
priority-medium-test-0	1/1	Running	0	5m28s
priority-medium-test-1	1/1	Running	0	5m28s
priority-medium-test-2	1/1	Running	0	5m28s
priority-medium-test-3	1/1	Running	0	5m28s

----End

## 6.5.4 AI Performance-based Scheduling

### 6.5.4.1 DRF

Dominant Resource Fairness (DRF) is a scheduling algorithm based on the dominant resource of a container group. DRF scheduling can be used to enhance

the service throughput of a cluster, shorten the overall service execution time, and improve service running performance. It is suitable for batch AI training and big data jobs.

## Prerequisites

- A cluster of v1.19 or later is available. For details, see [Buying a CCE Standard/Turbo Cluster](#).
- The Volcano add-on has been installed. For details, see [Volcano Scheduler](#).

## How It Works

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:

1. Resources are allocated in order of increasing demand.
2. No source gets a resource share larger than its demand.
3. Sources with unsatisfied demands get an equal share of the resource.

The max-min fairness algorithm applies to the single resource scenario, where all jobs are requesting the same resources. However, in actual situations, multiple resources are involved. For example, CPU, memory, and GPU resources are requested for allocation. DRF can be used to resolve the preceding issue. DRF can be considered as a general version of the max-min fairness algorithm and supports fair allocation of multiple types of resources so that the dominant resource of each user meets the max-min fairness requirement.

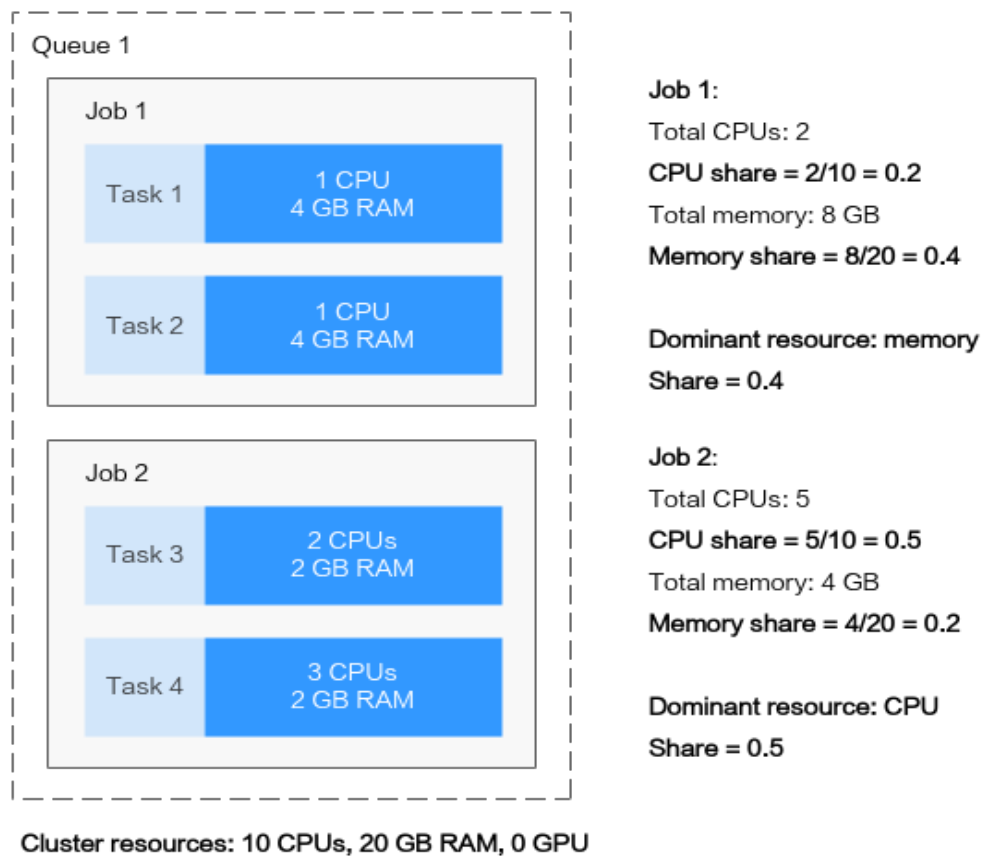
The share value of each job resource is calculated using the following formula:

**Share = Total requested resources/Cluster resources**

If a job involves multiple resources, the resource with the largest share value is the dominant resource. The share value of the dominant resource will be used in priority-based scheduling.

For example, there are two workloads, job 1 and job 2. The following figure shows the resources requested by the two jobs. After DRF calculation, the dominant resource of job 1 is memory, and its share value is 0.4; the dominant resource of job 2 is CPU, and its share value is 0.5. Since the dominant resource share of job 1 is less than that of job 2, job 1 takes precedence over job 2 in scheduling according to the max-min fairness policy.

**Figure 6-16** DRF scheduling



## Configuring DRF

After Volcano is installed, you can enable or disable DRF scheduling on the **Scheduling** page. This function is enabled by default.

**Step 1** Log in to the CCE console.

**Step 2** Click the cluster name to access the cluster console. Choose **Settings** in the navigation pane. In the right pane, click the **Scheduling** tab.

**Step 3** In the **AI task performance enhanced scheduling** pane, select whether to enable DRF.

This function helps you enhance the service throughput of the cluster and improve service running performance.

**Step 4** Click **Confirm**.

----End

### 6.5.4.2 Gang

Gang scheduling is a scheduling algorithm that schedules correlated processes or threads to run simultaneously on different processors. It 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. Gang 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.

## Prerequisites

- A cluster of v1.19 or later is available. For details, see [Buying a CCE Standard/Turbo Cluster](#).
- The Volcano add-on has been installed. For details, see [Volcano Scheduler](#).

## How It Works

The Gang scheduling policy is one of the core scheduling algorithms of Volcano. It 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. 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.

The Gang scheduling algorithm based on container groups is well suitable for scenarios where multi-process collaboration is required. AI 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 master and slave 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.

## Configuring Gang

After Volcano is installed, you can enable or disable Gang scheduling on the **Scheduling** page. This function is enabled by default.

**Step 1** Log in to the CCE console.

**Step 2** Click the cluster name to access the cluster console. Choose **Settings** in the navigation pane. In the right pane, click the **Scheduling** tab.

**Step 3** In the **AI task performance enhanced scheduling** pane, select whether to enable Gang.

This function helps you enhance the service throughput of the cluster and improve service running performance.

**Step 4** Click **Confirm**.

**Step 5** After the configuration, use Gang scheduling in workloads or Volcano jobs.

- Create a workload using Gang scheduling.
  - a. Create a pod group and specify **minMember** and **minResources** as follows:

```
apiVersion: scheduling.volcano.sh/v1beta1
kind: PodGroup
```

```

metadata:
  name: pg-test1
spec:
  minMember: 3
  minResources:
    cpu: 3
    memory: 3Gi

```

- **minMember:** specifies the minimum requirement on the number of pods for running a workload. When the number of pods in the current pod group meets the requirement, these pods can be centrally scheduled.
  - **minResources:** specifies the minimum requirement on resources for running a workload. When the available resources in a cluster meet the requirement, the group of pods can be centrally scheduled.
- b. When creating a workload, use **schedulerName** to specify Volcano Scheduler and **annotation** to specify the pod group in which Volcano Scheduler runs.

```

apiVersion: apps/v1
kind: Deployment
metadata:
  name: podgroup-test
  labels:
    app: podgroup-test
spec:
  replicas: 6
  selector:
    matchLabels:
      app: podgroup-test
  template:
    metadata:
      annotations:
        scheduling.k8s.io/group-name: pg-test1
      labels:
        app: podgroup-test
    spec:
      schedulerName: volcano
      containers:
      - name: test
        image: busybox
        imagePullPolicy: IfNotPresent
        command: ['sh', '-c', 'echo "Hello, Kubernetes!" && sleep 3600']
        resources:
          requests:
            cpu: 500m
          limits:
            cpu: 500m

```

- **schedulerName:** Set this parameter to **volcano**, indicating that Volcano will be used to schedule pods for the workload.
  - **scheduling.k8s.io/group-name:** specifies the pod group created in the previous step, for example, **pg-test1**.
- Create a Volcano job using Gang scheduling.  
When creating a Volcano job, you only need to configure **minAvailable** and set **schedulerName** to **volcano**. Volcano Scheduler will automatically create a pod group and manage it. The following shows an example:

```

apiVersion: batch.volcano.sh/v1alpha1
kind: Job
metadata:
  name: vcjob
spec:

```

```
schedulerName: volcano
minAvailable: 2
tasks:
- replicas: 4
  name: "test"
  template:
    spec:
      containers:
      - image: alpine
        command: ["/bin/sh", "-c", "sleep 1000"]
        imagePullPolicy: IfNotPresent
        name: running
        resources:
          requests:
            cpu: "1"
        restartPolicy: OnFailure
```

----End

## 6.5.5 NUMA Affinity Scheduling

In non-uniform memory access (NUMA) architecture, a NUMA node is a fundamental component that includes a processor and local memory. These nodes are physically separate but interconnected through a high-speed bus to form a complete system. To boost system performance, NUMA nodes allow for quicker access to local memory. However, accessing memory across multiple NUMA nodes within a node can cause delays. To enhance memory access efficiency and overall performance, it is crucial to optimize task scheduling and memory allocation.

When working with high-performance computing (HPC), real-time applications, or memory-intensive workloads that require frequent communication between CPUs, accessing nodes across NUMA in a cloud native environment can lead to decreased system performance due to increased latency and overhead. Volcano's NUMA affinity scheduling 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.

For more information, see <https://github.com/volcano-sh/volcano/blob/master/docs/design/numa-aware.md>.

### Prerequisites

- A CCE standard or Turbo cluster is available. For details, see [Buying a CCE Standard/Turbo Cluster](#).
- The Volcano add-on has been installed in the cluster. For details, see [Volcano Scheduler](#).

### Pod Scheduling Process

After a topology policy is configured for pods, Volcano predicts the nodes that match the policy. For details about how to configure a pod topology policy, see [Example of NUMA Affinity Scheduling](#). The scheduling process is as follows:

1. Volcano filters nodes with the same policy based on the topology policy configured for pods. The topology policy provided by Volcano is the same as that provided by the [topology manager](#).
2. Among the nodes where the same policy applies, Volcano selects the nodes whose CPU topology meets the policy requirements for scheduling.

Pod Topology Policy	How to Filter Nodes During Pod Scheduling	
	1. Filter nodes that meet the topology policy set for the pod.	2. Further filter the node whose CPU topology meets the policy.
none	<p>Nodes with the following topology policies will not be filtered during scheduling:</p> <ul style="list-style-type: none"> <li>• <b>none</b>: schedulable</li> <li>• <b>best-effort</b>: schedulable</li> <li>• <b>restricted</b>: schedulable</li> <li>• <b>single-numa-node</b>: schedulable</li> </ul>	None
best-effort	<p>Nodes with the <b>best-effort</b> topology policy will be filtered.</p> <ul style="list-style-type: none"> <li>• <b>none</b>: unschedulable</li> <li>• <b>best-effort</b>: schedulable</li> <li>• <b>restricted</b>: unschedulable</li> <li>• <b>single-numa-node</b>: unschedulable</li> </ul>	<p>Best-effort scheduling: Pods are preferentially scheduled to a single NUMA node. If a single NUMA node cannot meet the requested CPU cores, the pods can be scheduled to multiple NUMA nodes.</p>
restricted	<p>Nodes with the <b>restricted</b> topology policy will be filtered.</p> <ul style="list-style-type: none"> <li>• <b>none</b>: unschedulable</li> <li>• <b>best-effort</b>: unschedulable</li> <li>• <b>restricted</b>: schedulable</li> <li>• <b>single-numa-node</b>: unschedulable</li> </ul>	<p>Restricted scheduling:</p> <ul style="list-style-type: none"> <li>• If the upper CPU limit of a single NUMA node is greater than or equal to the requested CPU cores, pods can only be scheduled to a single NUMA node. If the remaining CPU cores of a single NUMA node are insufficient, the pods cannot be scheduled.</li> <li>• If the upper CPU limit of a single NUMA node is less than the requested CPU cores, pods can be scheduled to multiple NUMA nodes.</li> </ul>
single-numa-node	<p>Nodes with the <b>single-numa-node</b> topology policy will be filtered.</p> <ul style="list-style-type: none"> <li>• <b>none</b>: unschedulable</li> <li>• <b>best-effort</b>: unschedulable</li> <li>• <b>restricted</b>: unschedulable</li> <li>• <b>single-numa-node</b>: schedulable</li> </ul>	Pods can only be scheduled to a single NUMA node.

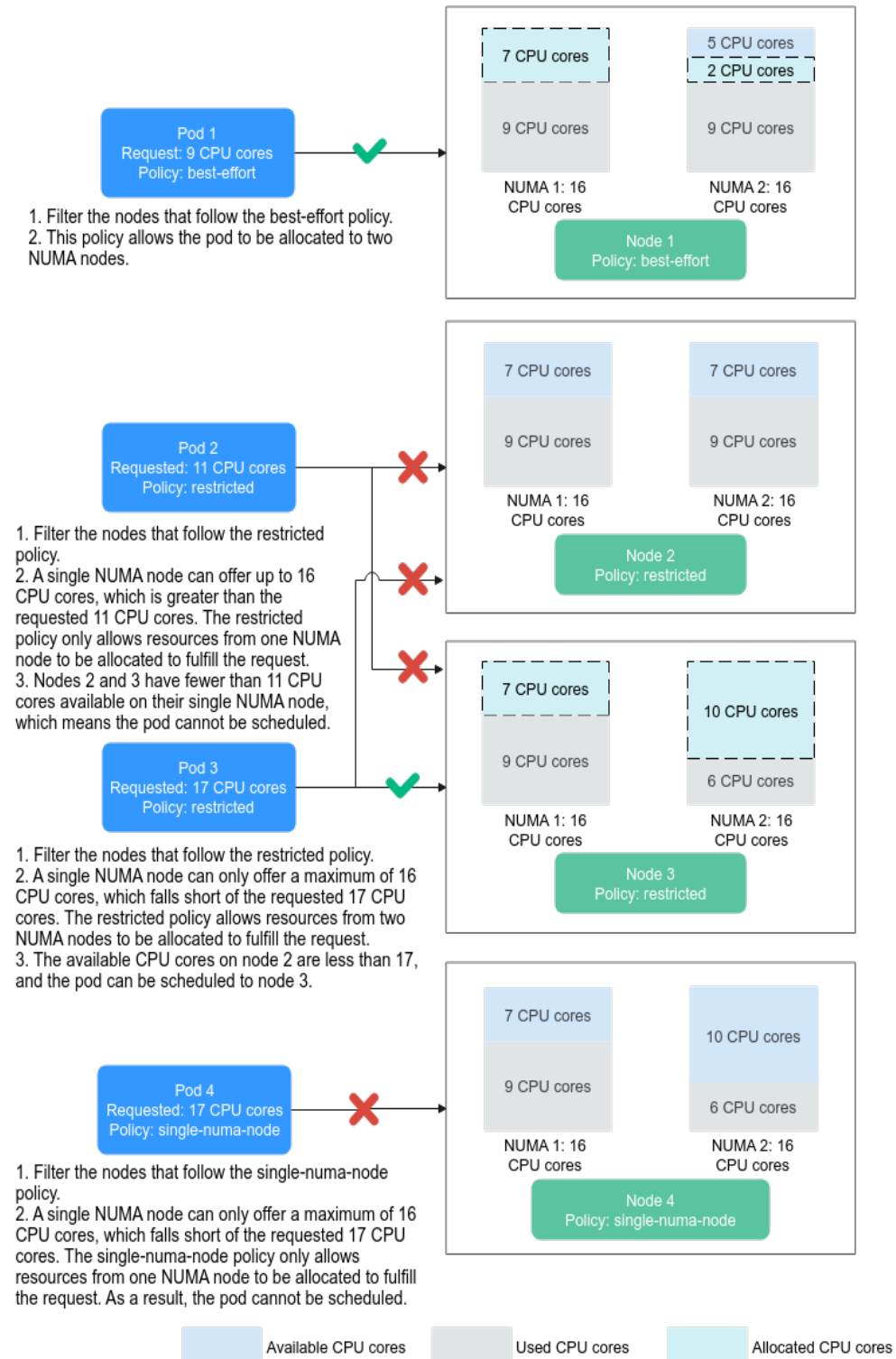
For example, two NUMA nodes provide resources, each with a total of 32 CPU cores. The following table lists resource allocation.

Worker Node	Node Topology Policy	Total CPU Cores on NUMA Node 1		Total CPU Cores on NUMA Node 2	
		Total CPU Cores	Available CPU Cores	Total CPU Cores	Available CPU Cores
Node 1	best-effort	16	7	16	7
Node 2	restricted	16	7	16	7
Node 3	restricted	16	7	16	10
Node 4	single- numa- node	16	7	16	10

**Figure 6-17** shows the scheduling of a pod after a topology policy is configured.

- When 9 CPU cores are requested by a pod and the **best-effort** topology policy is used, Volcano selects node 1 whose topology policy is also **best-effort**, and this policy allows the pod to be scheduled to multiple NUMA nodes. Therefore, the requested 9 CPU cores will be allocated to two NUMA nodes, and the pod can be scheduled to node 1.
- When 11 CPU cores are requested by a pod and the **restricted** topology policy is used, Volcano selects nodes 2 and 3 whose topology policy is also **restricted**, and each node provides at least 11 CPU cores. However, the remaining CPU cores on node 2 or 3 are less than the requested. Therefore, the pod cannot be scheduled.
- When 17 CPU cores are requested by a pod and the **restricted** topology policy is used, Volcano selects nodes 2 and 3 whose topology policy is also **restricted**, this policy allows the pod to be scheduled to multiple NUMA nodes, and the upper CPU limit of both the nodes is less than 17. Then, the pod can be scheduled to node 3.
- When 17 CPU cores are requested by a pod and the **single-numa-node** topology policy is used, Volcano selects nodes whose topology policy is also **single-numa-node**. However, no node can provide a total of 17 CPU cores. Therefore, the pod cannot be scheduled.

**Figure 6-17** Comparison of NUMA scheduling policies



## Scheduling Priority

A topology policy aims to schedule pods to the optimal node. In this example, each node is scored to sort out the optimal node.

Principle: Schedule pods to the worker nodes that require the fewest NUMA nodes.

The scoring formula is as follows:

$$\text{score} = \text{weight} \times (100 - 100 \times \text{numaNodeNum} / \text{maxNumaNodeNum})$$

Parameters:

- **weight**: the weight of NUMA Aware Plugin.
- **numaNodeNum**: the number of NUMA nodes required for running the pod on worker nodes.
- **maxNumaNodeNum**: the maximum number of NUMA nodes required for running the pod among all worker nodes.

For example, three nodes meet the CPU topology policy for a pod and the weight of NUMA Aware Plugin is set to **10**.

- Node A: One NUMA node provides the CPU resources required by the pod (numaNodeNum = 1).
- Node B: Two NUMA nodes provide the CPU resources required by the pod (numaNodeNum = 2).
- Node C: Four NUMA nodes provide the CPU resources required by the pod (numaNodeNum = 4).

According to the preceding formula, **maxNumaNodeNum** is **4**.


- score (Node A) = 10 × (100 - 100 × 1/4) = 750
- score (Node B) = 10 × (100 - 100 × 2/4) = 500
- score (Node C) = 10 × (100 - 100 × 4/4) = 0

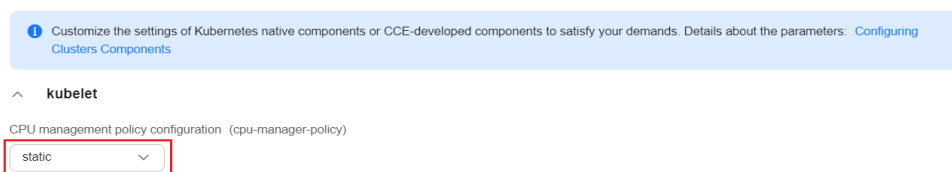
Therefore, the optimal node is Node A.

## Enabling NUMA Affinity Scheduling for Volcano

**Step 1** Enable static CPU management in the node pool. For details, see [Enabling CPU Management for a Custom Node Pool](#).

1. Log in to the CCE console and click the cluster name to access the cluster console.
2. Choose **Nodes** in the navigation pane and click the **Node Pools** tab on the right. Locate the target node pool and choose **More > Manage**.
3. On the **Manage Configurations** page, change the **cpu-manager-policy** value to **static** in the **kubelet** area.

Manage Configurations (Node Pool , nodepool-52200)



4. Click **OK**.

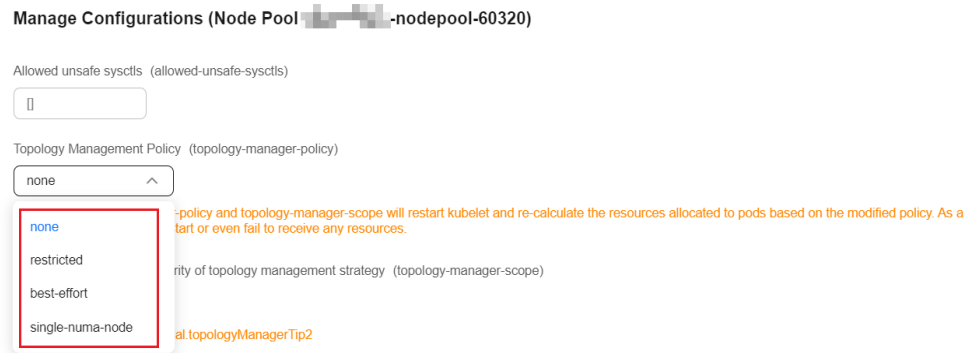
**Step 2** Configure a CPU topology policy in the node pool.

1. Log in to the CCE console and click the cluster name to access the cluster console. In the navigation pane, choose **Nodes**. On the right of the page, click

the **Node Pools** tab and choose **More > Manage** in the **Operation** column of the target node pool.

2. Change the kubelet **Topology Management Policy (topology-manager-policy)** value to the required CPU topology policy.

Valid topology policies include **none**, **best-effort**, **restricted**, and **single-numa-node**. For details, see [Pod Scheduling Process](#).



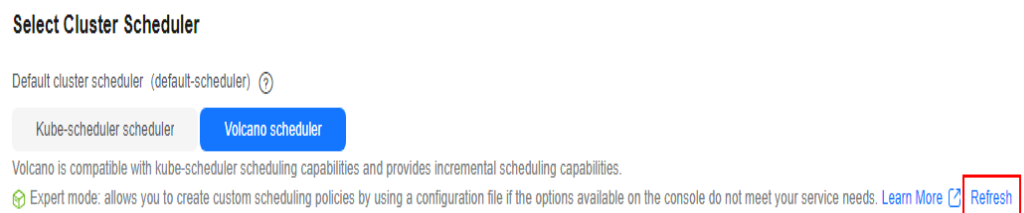
### Step 3 Enable the numa-aware add-on and the **resource\_exporter** function.

#### Volcano 1.7.1 or later

1. Log in to the CCE console and click the cluster name to access the cluster console. Choose **Add-ons** in the navigation pane, locate **Volcano Scheduler** on the right, and click **Edit**.
2. In the **Extended Functions** area, enable **NUMA Topology Scheduling** and click **OK**.

#### Volcano earlier than 1.7.1

1. Log in to the CCE console and click the cluster name to access the cluster console. In the navigation pane, choose **Settings** and click the **Scheduling** tab. In the **Select Cluster Scheduler** area, select **Volcano scheduler**, find the expert mode, and click **Refresh**.



2. Enable **resource\_exporter\_enable** to collect node NUMA information. The following is an example in JSON format:

```
{
  "plugins": {
    "eas_service": {
      "availability_zone_id": "",
      "driver_id": "",
      "enable": "false",
      "endpoint": "",
      "flavor_id": "",
      "network_type": "",
      "network_virtual_subnet_id": "",
      "pool_id": "",
      "project_id": ""
    }
  }
}
```



```
    "secret_name": "eas-service-secret"
  }
},
"resource_exporter_enable": "true"
}
```

After this function is enabled, you can view the NUMA topology information of the current node.

```
kubectl get numatopo
NAME      AGE
node-1    4h8m
node-2    4h8m
node-3    4h8m
```

3. Enable the Volcano numa-aware algorithm add-on.

### **kubectl edit cm -n kube-system volcano-scheduler-configmap**

```
kind: ConfigMap
apiVersion: v1
metadata:
  name: volcano-scheduler-configmap
  namespace: kube-system
data:
  default-scheduler.conf: |-
    actions: "allocate, backfill, preempt"
    tiers:
    - plugins:
      - name: priority
      - name: gang
      - name: conformance
    - plugins:
      - name: overcommit
      - name: drf
      - name: predicates
      - name: nodeorder
    - plugins:
      - name: cce-gpu-topology-predicate
      - name: cce-gpu-topology-priority
      - name: cce-gpu
    - plugins:
      - name: nodelocalvolume
      - name: nodeemptydirvolume
      - name: nodeCSIScheduling
      - name: networkresource
    arguments:
      NetworkType: vpc-router
    - name: numa-aware # add it to enable numa-aware plugin
    arguments:
      weight: 10 # the weight of the NUMA Aware Plugin
```

----End

## Example of NUMA Affinity Scheduling

The following describes how to choose NUMA nodes for scheduling pods according to pod scheduling policies. For details, see [Pod Scheduling Process](#).

- **single-numa-node:** When pods are scheduled, nodes in the node pool with the **single-numa-node** topology management policy will be chosen, and a single NUMA node must provide the CPU cores. If none of the nodes in the pool meet these requirements, the pod cannot be scheduled.
- **restricted:** When pods are scheduled, nodes in the node pool with the **restricted** topology management policy will be chosen, and a set of NUMA nodes on the same node must provide the CPU cores. If none of the nodes in the pool meet these requirements, the pod cannot be scheduled.

- **best-effort:** When pods are scheduled, nodes in the node pool with the **best-effort** topology management policy will be chosen, and a single NUMA node needs to provide the CPU cores. If none of the nodes in the pool meet these requirements, the pod will be scheduled to the most suitable node.

**Step 1** Refer to the following examples for configuration.

1. Example 1: Configure NUMA affinity for a Deployment.

```
kind: Deployment
apiVersion: apps/v1
metadata:
  name: numa-tset
spec:
  replicas: 1
  selector:
    matchLabels:
      app: numa-tset
  template:
    metadata:
      labels:
        app: numa-tset
      annotations:
        volcano.sh/numa-topology-policy: single-numa-node # Configure the topology policy.
    spec:
      containers:
        - name: container-1
          image: nginx:alpine
          resources:
            requests:
              cpu: 2 # The value must be an integer and must be the same as that in limits.
              memory: 2048Mi
            limits:
              cpu: 2 # The value must be an integer and must be the same as that in requests.
              memory: 2048Mi
          imagePullSecrets:
            - name: default-secret
```

2. Example 2: Create a Volcano job and enable NUMA affinity for it.

```
apiVersion: batch.volcano.sh/v1alpha1
kind: Job
metadata:
  name: vj-test
spec:
  schedulerName: volcano
  minAvailable: 1
  tasks:
    - replicas: 1
      name: "test"
      topologyPolicy: best-effort # set the topology policy for task
      template:
        spec:
          containers:
            - image: alpine
              command: ["/bin/sh", "-c", "sleep 1000"]
              imagePullPolicy: IfNotPresent
              name: running
              resources:
                limits:
                  cpu: 20
                  memory: "100Mi"
              restartPolicy: OnFailure
```

**Step 2** Analyze NUMA scheduling.

The following table shows example NUMA nodes.

Worker Node	Topology Manager Policy	Allocatable CPU Cores on NUMA Node 0	Allocatable CPU Cores on NUMA Node 1
Node 1	single-numa-node	16	16
Node 2	best-effort	16	16
Node 3	best-effort	20	20

In the preceding examples,

- In example 1, 2 CPU cores are requested by a pod, and the **single-numa-node** topology policy is used. Therefore, the pod will be scheduled to node 1 with the same policy.
- In example 2, 20 CPU cores are requested by a pod, and the **best-effort** topology policy is used. The pod will be scheduled to node 3 because it can allocate all the requested 20 CPU cores onto one NUMA node, while node 2 can do so on two NUMA nodes.

----End

## Checking NUMA Node Usage

Run the **lscpu** command to check the CPU usage of the current node.

```
# Check the CPU usage of the current node.
lscpu
...
CPU(s):          32
NUMA node(s):    2
NUMA node0 CPU(s): 0-15
NUMA node1 CPU(s): 16-31
```

Then, check the NUMA node usage.

```
# Check the CPU allocation of the current node.
cat /var/lib/kubelet/cpu_manager_state
{"policyName":"static","defaultCpuSet":"0,10-15,25-31","entries":{"777870b5-c64f-42f5-9296-688b9dc212ba":{"container-1":"16-24"},"fb15e10a-b6a5-4aaa-8fcd-76c1aa64e6fd":{"container-1":"1-9"}}, "checksum":318470969}
```

The preceding example shows that two containers are running on the node. One container uses CPU cores 1 to 9 of NUMA node 0, and the other container uses CPU cores 16 to 24 of NUMA node 1.

## Common Issues

### Schedule pods failed.

If Volcano is set as the scheduler and only NUMA is enabled without configuring CPU management during pod scheduling, job scheduling may fail. To fix this issue, do as follows:

- Before using NUMA affinity scheduling, make sure that Volcano has been deployed and is running properly.

- When using NUMA affinity scheduling:
  - a. Set the CPU management policy (**cpu-manager-policy**) of the node pool to **static**.
  - b. Correctly configure the topology management policy (**topology-manager-policy**) in the node pool.
  - c. Configure a correct topology policy for pods to filter nodes with the same topology policy in the node pool. For details, see [Example of NUMA Affinity Scheduling](#).
  - d. Configure the Volcano scheduler to schedule application pods. For details, see [Scheduling Workloads](#). Make sure that the CPU requests for all containers within the pods are integers (measured in cores) and that the requests and limits are identical.

## 6.5.6 Application Scaling Priority Policies

With application scaling priority policies, you can manage resources more efficiently by customizing the scaling order of pods across different node types. 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.

The application scaling priority policy includes the following two 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.

### Notes and Constraints

- The cluster version must be 1.23.11 or later, 1.25.6 or later, or 1.27.3 or later.
- The Volcano Scheduler add-on (1.12.1 or later) must be installed in a cluster, and the application scaling priority policy function must be enabled.
- By default, the scaling priority takes effect for Deployments (including [ReplicaSet](#)).
- To use the scale-out scheduling priority policies, you need to set **spec.schedulerName** of a workload to **volcano** or set the default cluster scheduler to **volcano**. The application scaling priority policy function applies not to workloads with no resource limit and requested resources configured.
- If the default priority policy is used, Volcano Scheduler schedules workloads based on the priorities of yearly/monthly nodes, pay-per-use nodes, and virtual-kubelet nodes (scaling pods to CCI). However, the priorities cannot be fully implemented, because Volcano Scheduler takes scheduling results into account from multiple dimensions rather than just one.
- Volcano Scheduler must balance scheduling performance with scheduling results. When there are a large number of schedulable nodes in a cluster, it selects only some of them for scheduling to ensure scheduling performance and will not find the best global scheduling solution. For details, see [Scheduler Performance Tuning](#). This behavior conflicts with the scaling

priority policies. But you can make Volcano Scheduler select all nodes for scheduling by [adjusting the proportion of nodes that can be scheduled by Volcano Scheduler](#).

## Overview

After the application scaling priority policy is enabled, the **Balancer** and **BalancerPolicyTemplate** CRDs are added to a cluster, and the default scaling priority policy is created. For details, see [Applying the Default Application Scaling Priority Policy](#). Volcano Scheduler obtains the priority of each node based on the **BalancerPolicyTemplate** CR to control the pod scheduling priority during application scale-out. In addition, it configures the priority during application scale-in based on both **Balancer** and **BalancerPolicyTemplate** CRs.

- The **BalancerPolicyTemplate** CRDs are used to define priority policies. For example, in the default scaling priority policy, the **BalancerPolicyTemplate** CR assigns the highest priority to yearly/monthly nodes, followed by pay-per-use nodes, and the lowest priority to virtual-kubelet nodes (scaling pods to CCI) by default.

The **BalancerPolicyTemplate** CRs cannot be updated.

- The **Balancer** CRDs are used to declare the application scope of scaling priorities. When creating a **Balancer** CR, you can specify a workload in a namespace, a specific Deployment, or a specific ReplicaSet as the application scope.

A **Balancer** CR corresponds to a **BalancerPolicyTemplate** CR. They work together to determine which priority policies are applied to specific workloads.

In Volcano Scheduler's default scaling priority policy, the **BalancerPolicyTemplate** CR classifies yearly/monthly nodes, pay-per-use nodes, and virtual-kubelet nodes (scaling pods to CCI) into different priorities. Volcano Scheduler takes these priorities into account during scale-out and preferentially schedules new pods to the yearly/monthly nodes with higher priorities.

Volcano Scheduler applies annotations to pods within the application scope specified by the **Balancer** CR based on the priorities set by the **BalancerPolicyTemplate** CR. It may add the following annotations to a pod that meets the conditions:

- **openvessel.io/workload-balancer-score**: indicates a pod's score, which is higher if the pod is on a high-priority node.
- **autoscaling.volcano.sh/dominated-by-balancer**: specifies the **Balancer** CR that controls the current pod. Pods with low scores are preferentially scaled in.

### NOTE

If the existing pods already have the community supported [controller.kubernetes.io/pod-deletion-cost](#) annotation added, scale-in will be performed based on the priority defined by this annotation. If two pods have the same value for this annotation, the **openvessel.io/workload-balancer-score** annotation will be used to determine which pod to scale-in.

## Configuring an Application Scaling Priority Policy

- Step 1** Install Volcano Scheduler in a cluster and enable the application scaling priority policy. The default scaling priority policy will be created in the cluster.

1. Obtain a default **Balancer CR**.

```
# kubectl get balancer default-balancer -oyaml

apiVersion: autoscaling.volcano.sh/v1alpha1
kind: Balancer
metadata:
  name: default-balancer
spec:
  balancerPolicyTemplateName: default-balancerpolicytemplate
  targets:
  - namespaceSelector:
      matchExpressions:
      - key: kubernetes.io/metadata.name
        operator: Exists
    weight: 10
```

2. Obtain a default **BalancerPolicyTemplate CR**.

```
# kubectl get balancerpolicytemplate default-balancerpolicytemplate -oyaml

apiVersion: autoscaling.volcano.sh/v1alpha1
kind: BalancerPolicyTemplate
metadata:
  name: default-balancerpolicytemplate
spec:
  policy:
    policyName: Priority
    priorities:
      priorityGroups:
      - priority: 10
        requirements:
        - key: node.cce.io/billing-mode
          operator: In
          values:
          - post-paid
      - priority: 100
        requirements:
        - key: node.cce.io/billing-mode
          operator: In
          values:
          - pre-paid
      - priority: 1
        requirements:
        - key: kubernetes.io/role
          operator: In
          values:
          - virtual-kubelet
          - bursting
```

For details about the parameters, see [Applying the Default Application Scaling Priority Policy](#).

**Step 2** Deploy a workload and set the number of pods to 1.

Pods of the current workload are preferentially scheduled to yearly/monthly nodes.

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: balancer-test
  namespace: default
  labels:
    virtual-kubelet.io/burst-to-cci: 'auto' #If the resources of a cluster are not enough, pods in this cluster
    can be deployed on CCI.
spec:
  replicas: 1
  selector:
    matchLabels:
      app: balancer-test
```

```
template:
  metadata:
    labels:
      app: balancer-test
  spec:
    containers:
    - image: nginx:latest
      imagePullPolicy: IfNotPresent
      name: container-1
    resources:
      limits:
        cpu: 250m
        memory: 512Mi
      requests:
        cpu: 250m
        memory: 512Mi
    schedulerName: volcano
```

**Step 3** Increase the number of workload pods to 5.

Pods of the current workload are preferentially scheduled to yearly/monthly nodes. If there are not enough yearly/monthly nodes, these pods will be preferentially scheduled to pay-per-use nodes. If there are not enough pay-per-use nodes, these pods will be scheduled to virtual-kubelet nodes (scaling pods to CCI).

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: balancer-test
  namespace: default
  labels:
    virtual-kubelet.io/burst-to-cci: 'auto' #If the resources of a cluster are not enough, pods in this cluster
    can be deployed on CCI.
spec:
  replicas: 5
  selector:
    matchLabels:
      app: balancer-test
  template:
    metadata:
      labels:
        app: balancer-test
    spec:
      containers:
      - image: nginx:latest
        imagePullPolicy: IfNotPresent
        name: container-1
      resources:
        limits:
          cpu: 250m
          memory: 512Mi
        requests:
          cpu: 250m
          memory: 512Mi
      schedulerName: volcano
```

**Step 4** View the scores of pods.

1. Pods on a yearly/monthly node:

```
apiVersion: v1
kind: Pod
metadata:
  annotations:
    autoscaling.volcano.sh/dominated-by-balancer: default-balancer #The Balancer CR named default-balancer controls the scaling priority of the current pods.
    openvessel.io/workload-balancer-score: "100" #Priority of the current yearly/monthly node, which also indicates the pods' score
  ...
  nodeName: 192.168.20.100 #A yearly/monthly node
```

2. Pods on a pay-per-use node:

```
apiVersion: v1
kind: Pod
metadata:
  annotations:
    autoscaling.volcano.sh/dominated-by-balancer: default-balancer #The Balancer CR named default-balancer controls the scaling priority of the current pods.
    openvessel.io/workload-balancer-score "10" #Priority of the current pay-per-use node, which also indicates the pods' score
    ...
  nodeName: 192.168.20.196 #A pay-per-use node
```

3. Pods on a virtual-kubelet node (scaling pods to CCI):

```
apiVersion: v1
kind: Pod
metadata:
  annotations:
    autoscaling.volcano.sh/dominated-by-balancer: default-balancer #The Balancer CR named default-balancer controls the scaling priority of the current pods.
    openvessel.io/workload-balancer-score: "1" #Priority of the current virtual-kubelet node, which also indicates the pods' score
    ...
  nodeName: virtual-kubelet #A virtual-kubelet node
```

**Step 5** Gradually reduce the number of the workload pods.

Pods on virtual-kubelet nodes (scaling pods to CCI) are deleted first, followed by pods on pay-per-use nodes and those on yearly/monthly nodes.

----End

## Applying the Default Application Scaling Priority Policy

When the default application scaling priority policy is used, the following default CRs are present in a cluster:

- **A Balancer CR:**

```
apiVersion: autoscaling.volcano.sh/v1alpha1
kind: Balancer
metadata:
  name: default-balancer
spec:
  balancerPolicyTemplateName: default-balancerpolicytemplate
  targets:
  - namespaceSelector:
    matchExpressions:
      - key: kubernetes.io/metadata.name
        operator: Exists
    weight: 10
```

**Table 6-11** Key parameters of a Balancer CR

Field	Description	Type	Remarks
metadata.name	Name	String	This field is mandatory.
spec.balancerPolicyTemplateName	Name of the priority policy	String	This field is mandatory. The value is the name of the corresponding <b>BalancerPolicyTemplate</b> CR in the cluster.



Field	Description	Type	Remarks
spec.targets	Application scope of the priority policy	Slice	<p>This field is mandatory. Example:</p> <ul style="list-style-type: none"> <li>Applying to applications in the <b>default</b> namespace:  <pre>spec:   targets:   - namespaceSelector:       matchLabels:         kubernetes.io/metadata.name: default</pre> </li> <li>Applying to applications in multiple namespaces like <b>default</b>, <b>other</b>, and <b>another</b>:  <pre>spec:   targets:   - namespaceSelector:       matchExpressions:       - key: kubernetes.io/metadata.name         operator: In         values:         - default         - other         - another</pre> </li> <li>Applying to applications in all namespaces:  <pre>spec:   targets:   - namespaceSelector:       matchExpressions:       - key: kubernetes.io/metadata.name         operator: Exists</pre> </li> <li>Only applying to Deployments which are named in the format of <b>XXX-XXX-XXX</b>:  <pre>spec:   targets:   - objectSelectors:       - name: xxx-xxx-xxx         kind: Deployment</pre> </li> <li>Only applying to Deployments which are named in the format of <b>XXX-XXX-XXX</b> and are in the <b>default</b> namespace:  <pre>spec:   targets:   - namespaceSelector:       matchLabels:         kubernetes.io/metadata.name: default       objectSelectors:       - name: xxx-xxx-xxx         kind: Deployment</pre> </li> </ul>
spec.weight	Weight of the priority policy	int32	<p>This field is mandatory. When there are multiple <b>Balancer</b> CRs in a cluster, an application may fall within the scope of more than one of them. In such cases, the <b>Balancer</b> CR with the highest weight will be applied.</p>

- A BalancerPolicyTemplate CR:**  
 apiVersion: autoscaling.volcano.sh/v1alpha1  
 kind: BalancerPolicyTemplate  
 metadata:  
   name: default-balancerpolicytemplate  
 spec:  
   policy:  
     policyName: Priority  
     priorities:  
       priorityGroups:  
         - priority: 10  
           requirements:  
             - key: node.cce.io/billing-mode  
               operator: In  
               values:  
                 - post-paid  
         - priority: 100  
           requirements:  
             - key: node.cce.io/billing-mode  
               operator: In  
               values:  
                 - pre-paid  
         - priority: 1  
           requirements:  
             - key: kubernetes.io/role  
               operator: In  
               values:  
                 - virtual-kubelet  
                 - bursting

**Table 6-12** Key parameters of a BalancerPolicyTemplate CR

Field	Description	Type	Remarks
metadata.name	Name	String	This field is mandatory.
spec.policy	Content of the priority policy	Struct	This field is mandatory.
spec.policy.policyname	Name of the priority policy	String	This field is mandatory. Only the priority policy named <b>Priority</b> is supported.

Field	Description	Type	Remarks
spec.policy.priorities.priorityGroups	Specific priority defined in the priority policy	Slice	<p>This field is mandatory. Example:</p> <ul style="list-style-type: none"> <li>Setting the priority of a yearly/monthly node to <b>100</b>: <pre>priorityGroups: - priority: 100 requirements: - key: node.cce.io/billing-mode operator: In values: - pre-paid</pre> </li> <li>Setting the priority of a pay-per-use node to <b>10</b>: <pre>priorityGroups: - priority: 10 requirements: - key: node.cce.io/billing-mode operator: In values: - post-paid</pre> </li> <li>Setting the priority of a virtual-kubelet or bursting node to <b>1</b>: <pre>priorityGroups: - priority: 1 requirements: - key: kubernetes.io/role operator: In values: - virtual-kubelet - bursting</pre> </li> </ul>

## Customizing an Application Scaling Priority Policy

The **BalancerPolicyTemplate** CRDs are used to define priority policies. If you need to customize an application scaling priority policy, you need to modify the **BalancerPolicyTemplate** CR.

### NOTE

If there are multiple **BalancerPolicyTemplate** CRs in a cluster, they will all affect the scaling result. Therefore, if the default scaling priority policy is not in use, run the following command to delete it:

```
kubectrl delete balancerpolicytemplate default-balancerpolicytemplate
```

Assume that **during scale-out, a workload is preferentially scheduled to a node running HCE 2.0 and then to a node running Euler. During scale-in, Volcano Scheduler first deletes the workload pods on the node running Euler and then deletes the pods on the node running HCE 2.0.**

### Step 1 Write a new **BalancerPolicyTemplate** CR.

```
vim new-balancerpolicytemplate.yaml
```

The content is as follows:

```
apiVersion: autoscaling.volcano.sh/v1alpha1
kind: BalancerPolicyTemplate
metadata:
```

```
name: new-balancerpolicytemplate
spec:
  policy:
    policyName: Priority
    priorities:
      priorityGroups:
        - priority: 10 # Set the priority of the node running EulerOS to 10.
          requirements:
            - key: os.name # Label of the Node OS
              operator: In
              values:
                - EulerOS_2.0_SP9x86_64 # The minor version number of the OS may be involved. You can add the
minor version number as needed.
        - priority: 100 # Set the priority of the node running HCE 2.0 to 100.
          requirements:
            - key: os.name # Label of the Node OS
              operator: In
              values:
                - Huawei_Cloud_EulerOS_2.0_x86_64
```

**Step 2** Create a new **BalancerPolicyTemplate** CR.

```
kubectl create -f new-balancerpolicytemplate.yaml
```

**Step 3** Modify **default-balancer**. You can also create a new **Balancer** CR as needed.

```
kubectl edit balancer default-balancer
```

The modified content is as follows:

```
apiVersion: autoscaling.volcano.sh/v1alpha1
kind: Balancer
metadata:
  name: default-balancer
spec:
  balancerPolicyTemplateName: new-balancerpolicytemplate
  targets:
    - namespaceSelector:
        matchExpressions:
          - key: kubernetes.io/metadata.name
            operator: Exists
      weight: 10
```

**Step 4** Check whether the value of **openvessel.io/workload-balancer-score** in each pod meets the expectation.

The value of **openvessel.io/workload-balancer-score** in each pod on the node running EulerOS is set to **10**. The value of **openvessel.io/workload-balancer-score** in each pod on the node running HCE 2.0 is set to **100**.

----End

## Appendix: Adjusting the Proportion of Nodes That Can Be Scheduled by Volcano Scheduler

Write a volcano-scheduler resource object.

```
kubectl edit deploy volcano-scheduler -nkube-system
```

The content is as follows:

```
apiVersion: apps/v1
kind: Deployment
metadata:
  labels:
    app: volcano-scheduler
    app.kubernetes.io/managed-by: Helm
  release: cceaddon-volcano
  name: volcano-scheduler
  namespace: kube-system
```

```
spec:
  progressDeadlineSeconds: 600
  replicas: 1
  revisionHistoryLimit: 10
  selector:
    matchLabels:
      app: volcano-scheduler
  strategy:
    rollingUpdate:
      maxSurge: 10%
      maxUnavailable: 1
    type: RollingUpdate
  template:
    metadata:
      creationTimestamp: null
    labels:
      app: volcano-scheduler
      release: cceaddon-volcano
  spec:
    affinity:
      podAntiAffinity:
        preferredDuringSchedulingIgnoredDuringExecution:
          - podAffinityTerm:
              labelSelector:
                matchExpressions:
                  - key: app
                    operator: In
                    values:
                      - volcano-scheduler
              topologyKey: topology.kubernetes.io/zone
            weight: 100
        requiredDuringSchedulingIgnoredDuringExecution:
          - labelSelector:
              matchExpressions:
                - key: app
                  operator: In
                  values:
                    - volcano-scheduler
              topologyKey: kubernetes.io/hostname
    containers:
      - command:
          - /bin/sh
          - -c
          - /volcano-scheduler --leader-elect=true --lock-object-namespace=kube-system
            --feature-
            gates=CSIMigrationFlexVolumeFuxi=true,CSIMigrationFlexVolumeFuxiComplete=true,MultiGPUScheduling=true
            --kube-api-qps=200 --alsologtostderr --listen-address=$(MY_POD_IP):8080
            --enable-healthz=true --healthz-address=$(MY_POD_IP):11251 --enable-metrics=true --percentage-
            nodes-to-find=100
            --scheduler-conf=/volcano.scheduler/default-scheduler.conf -v=3 1>>/var/log/volcano/volcano-
            scheduler.log
```

**--percentage-nodes-to-find=100** specifies that Volcano Scheduler can find all nodes in a cluster during scheduling selection.

## 6.6 Cloud Native Hybrid Deployment

### 6.6.1 Overview

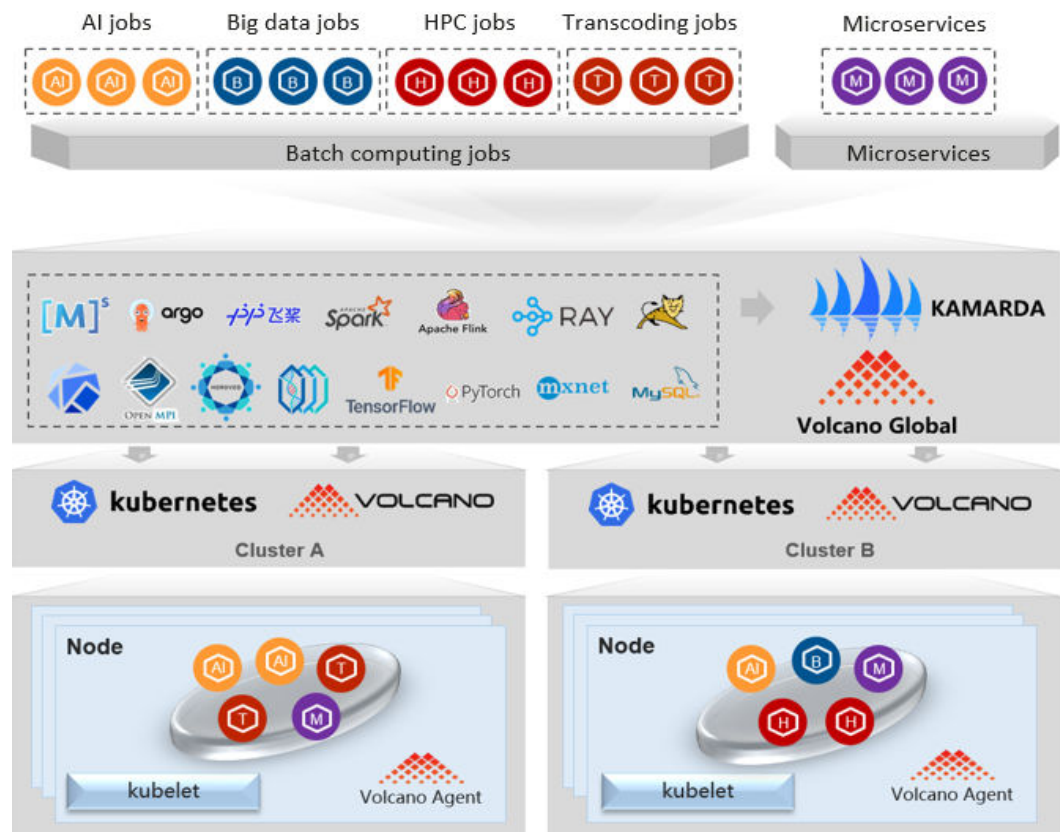
With the rapid development of cloud native technologies, many applications are going cloud native. From 2021 to 2022, the total number of cloud native applications in Kubernetes clusters has increased by more than 30% year-on-year. Kubernetes is becoming the platform for running almost anything — a virtual

"operating system" for cloud native applications in the cloud era. However, further research shows that the CPU usage of most user nodes in Kubernetes clusters is less than 15%. According to a survey conducted on a range of clients, the primary causes of low resource utilization may be summed up as follows, if interference elements like idle resources and package activities are taken out:

1. Nodes are deployed in different clusters. They cannot share compute resources with each other, resulting in an increase in resource fragments.
2. The node specifications are not ideal for applications that undergo frequent changes. At first, the node specifications 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 specifications. This leads to a decrease in the allocation rate of node resources and an increase in compute resource fragmentation.
3. 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.
4. 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.

These are typical instances of the difficulties encountered when creating cloud native applications. Various deployment solutions are needed for different service architectures during cloud native progress. Applications with varying architectures evolve at different paces, so development teams must balance service performance with service quality. How can these complex scenarios be made simpler so that customers can gradually increase resource usage and reduce costs?

CCE has created a cloud native hybrid deployment solution based on the Volcano and Kubernetes ecosystems, which helps users improve resource utilization, reduce costs, and increase efficiency. This solution is the result of years of exploration and practice in hybrid deployment.



As shown in this picture, hybrid deployment involves more than just merging small clusters into a larger one and deploying multiple services within that cluster. It requires the proper deployment of user applications and ensuring that the necessary resources are available to support their operation. This picture highlights the core designs in the cloud native hybrid deployment solution: unified scheduling of resources across all domains and hierarchical resource management.

## Unified Scheduling of Resources Across All Domains and Hierarchical Resource Management

### Unified scheduling of resources across all domains

Unified scheduling of resources across all domains ensures smooth scheduling of cross-cloud, cross-cluster resources in distributed cloud environments. This also enables unified scheduling of both online and offline services.

- Volcano uses static analysis to gather static features of applications, such as CPU, memory, storage, and GPU requirements, as well as affinity between applications, regions, and cloud platforms.
- Volcano then connects with the monitoring system to gather dynamic data from various cloud platforms, clusters, and running applications. This data is analyzed and used to learn about service fluctuations (daily, weekly, or monthly) and service types (CPU-sensitive, L3 cache-sensitive, or memory-sensitive).
- Finally, Volcano schedules applications to suitable environments using its flexible and customizable scheduling policies like prediction-based intelligent scheduling policy, service-based bin packing, rescheduling, and running status-based resource oversubscription.

Volcano manages resources on the distributed cloud platforms in a unified manner and schedules different types of applications to proper locations. This effectively solves the problem of resource fragmentation caused by multiple clusters and node specification mismatch caused by application updates. It frees users from complex resource planning and changes brought about by application iterations.

### Hierarchical resource management

Hierarchical resource management ensures that applications can access to the necessary resources in their designated environments after they have been scheduled.

Resources are isolated from multiple dimensions like CPU, L3 cache, memory, network, and storage based on Huawei Cloud EulerOS 2.0. Additionally, resources, mainly in the kernel mode, with some in the user mode, can be quickly preempted in milliseconds or evicted in seconds to ensure the quality of online services.

- Resource isolation measures (such as CPU core binding, NUMA affinity, tidal affinity, and network bandwidth control) ensure the resource-sensitive services to meet their SLOs.
- Resource priority control (such as hierarchical CPU suppression, memory tiering, network priority control, and disk I/O priority control) improves resource allocation and has little or no impact on the SLOs of high-priority services.

Hierarchical resource management provides a basis for deploying online services and hybrid deployment of online and offline services. This resolves problems that a large number of resources are reserved for applications and resources cannot be reused at different times.

## Online and Offline Jobs

Jobs can be classified into online jobs and offline jobs based on whether services are always online.

- **Online job:** Such jobs run for a long time, with regular traffic surges, tidal resource requests, and high requirements on SLA, such as advertising and e-commerce services.
- **Offline jobs:** Such jobs run for a short time, have high computing requirements, and can tolerate high latency, such as AI and big data services.

## Features

Function	Description	Documentation
Dynamic resource oversubscription	Based on the types of online and offline jobs, Volcano scheduling is used to utilize the resources that are requested but not used in the cluster (the difference between the number of requested resources and the number of used resources) for resource oversubscription and hybrid deployment to improve cluster resource utilization.	<a href="#">Dynamic Resource Oversubscription</a>



## 6.6.2 Dynamic Resource Oversubscription

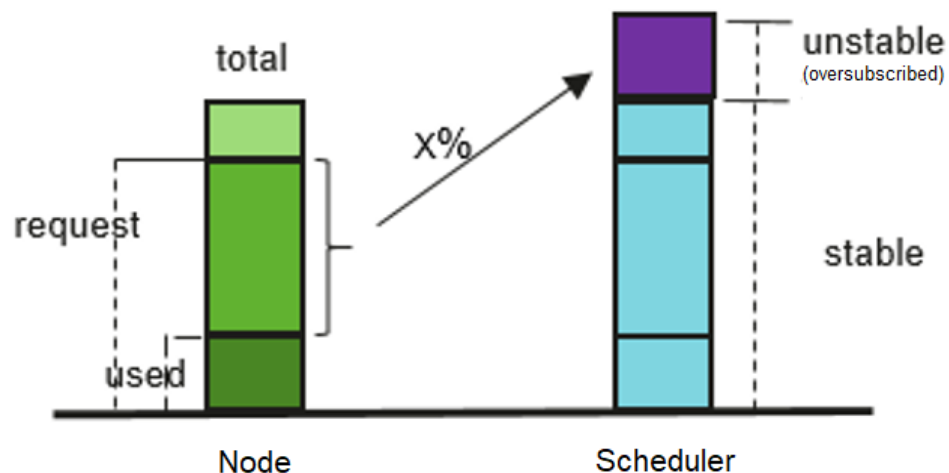
Many services see surges in traffic. To ensure performance and stability, resources are often requested at the maximum needed. However, the surges may ebb very shortly and resources, if not released, are wasted in non-peak hours. Especially for online jobs that request a large quantity of resources to ensure SLA, resource utilization can be as low as it gets.

Resource oversubscription is the process of making use of idle requested resources. Oversubscribed resources are suitable for deploying offline jobs, which focus on throughput but have low SLA requirements and can tolerate certain failures.

Hybrid deployment of online and offline jobs in a cluster can better utilize cluster resources.

**Figure 6-18** Resource oversubscription

$$\text{Oversubscription} = (\text{request} - \text{used}) \times \text{Ratio}$$



### Features

**NOTE**

After dynamic resource oversubscription and elastic scaling are enabled in a node pool, oversubscribed resources change rapidly because the resource usage of high-priority applications changes in real time. To prevent frequent node scale-ins and scale-outs, do not consider oversubscribed resources when evaluating node scale-ins.

Hybrid deployment is supported, and CPU and memory resources can be oversubscribed. The key features are as follows:

- Offline jobs preferentially run on oversubscribed nodes.  
If both oversubscribed and non-oversubscribed nodes exist, the former will score higher than the latter and offline jobs are preferentially scheduled to oversubscribed nodes.
- Online jobs can use only non-oversubscribed resources if scheduled to an oversubscribed node.

Offline jobs can use both oversubscribed and non-oversubscribed resources of an oversubscribed node.

- In the same scheduling period, online jobs take precedence over offline jobs. If both online and offline jobs exist, online jobs are scheduled first. When the node resource usage exceeds the upper limit and the node requests exceed 100%, offline jobs will be evicted.

- CPU/Memory resources can be isolation by kernel.

CPU isolation: Online jobs can quickly preempt CPU resources of offline jobs and suppress the CPU usage of the offline jobs.

Memory isolation: When system memory resources are used up and OOM Kill is triggered, the kernel evicts offline jobs first.

- kubelet offline jobs obey the following admission rules:

After the pod is scheduled to a node, kubelet starts the pod only when the node resources can meet the pod request (predicateAdmitHandler.Admit). kubelet starts the pod when both of the following conditions are met:

- The total request of pods to be started and online running jobs < allocatable nodes
- The total request of pods to be started and online/offline running job < allocatable nodes+oversubscribed nodes

- Resource oversubscription and hybrid deployment can be configured separately.

Enabling hybrid deployment of a node pool also enables oversubscription by default. Nodes are then labeled with both **volcano.sh/colocation="true"** and **volcano.sh/oversubscription="true"**. To use hybrid deployment for both online and offline jobs without oversubscription, simply disable oversubscription in hybrid deployment settings. This will remove the **volcano.sh/oversubscription="true"** label.

The following table lists the features that can be used after hybrid deployment or oversubscription is enabled.

Hybrid Deployment	Oversubscription	Oversubscription Resource	Scenario for Evicting Offline Pods
No	No	No	No
Yes	No	No	The actual resource usage of a node exceeds the upper limit.
No	Yes	Yes	The actual resource usage of a node exceeds the upper limit and the pod requests on the node exceed 100%.

<b>Hybrid Deployment</b>	<b>Oversubscription</b>	<b>Oversubscription Resource</b>	<b>Scenario for Evicting Offline Pods</b>
Yes	Yes	Yes	The actual resource usage of a node exceeds the upper limit.

## Compatible kubelet Oversubscription

### NOTICE

#### Specifications

- Cluster version
  - v1.19: v1.19.16-r4 or later
  - v1.21: v1.21.7-r0 or later
  - v1.23: v1.23.5-r0 or later
  - v1.25 or later
- Cluster type: CCE standard or CCE Turbo
- Node OS: EulerOS 2.9 (kernel-4.18.0-147.5.1.6.h729.6.eulerosv2r9.x86\_64) or Huawei Cloud EulerOS 2.0
- Node type: ECS
- Volcano version: 1.7.0 or later

#### Constraints

- Before enabling oversubscription, ensure that the overcommit add-on is not enabled on Volcano.
- Modifying the label of an oversubscribed node does not affect the running pods.
- Running pods cannot be converted between online and offline services. To convert services, rebuild pods.
- If the label **volcano.sh/oversubscription=true** is configured for a node in the cluster, the **oversubscription** configuration must be added to the Volcano add-on. Otherwise, the scheduling of oversold nodes will be abnormal. Ensure that you have correctly configured labels because the scheduler does not check the add-on and node configurations. For details, see [Table 6-13](#).
- To disable oversubscription, perform the following operations:
  - Remove the **volcano.sh/oversubscription** label from the oversubscribed node.
  - Set **over-subscription-resource** to **false**.
  - Modify the configmap of Volcano Scheduler named **volcano-scheduler-configmap** and remove the oversubscription add-on.
- If you have set **cpu-manager-policy** to statically bind CPU cores on a node, do not assign the QoS class of Guaranteed to offline pods. This is because offline pods may occupy the CPUs of online pods, leading to an online pod startup failure and offline pods failing to start even though they have been successfully scheduled. To prevent this, switch the pods to online pods if CPU core binding is required.
- If **cpu-manager-policy** is set to static CPU core binding on a node, do not bind CPU cores to all online pods. This is because doing so can cause online pods to occupy all available CPU or memory resources, leaving only a small number of oversubscribed resources.

---

If the label **volcano.sh/oversubscription=true** is configured for a node in the cluster, the **oversubscription** configuration must be added to the Volcano add-on.

Otherwise, the scheduling of oversold nodes will be abnormal. For details about the related configuration, see [Table 6-13](#).

Ensure that you have correctly configure labels because the scheduler does not check the add-on and node configurations.

**Table 6-13** Configuring oversubscription labels for scheduling

Oversubscription in Add-on	Oversubscription Label on Node	Scheduling
Yes	Yes	Triggered by oversubscription
Yes	No	Triggered
No	No	Triggered
No	Yes	Not triggered or failed. Avoid this configuration.

**Step 1** Use `kubectl` to access the cluster.

**Step 2** Check the Volcano configuration.

```
kubectl edit cm volcano-scheduler-configmap -n kube-system
```

Check the oversubscription configuration in **volcano-scheduler-configmap**.

Ensure that the add-on configuration does not contain the overcommit add-on. If **- name: overcommit** exists, delete this configuration.

```
...
data:
  volcano-scheduler.conf: |
    actions: "allocate, backfill, preempt" # Configure a preemption action.
    tiers:
    - plugins:
      - name: gang
        enablePreemptable: false
        enableJobStarving: false
      - name: priority
      - name: conformance
      - name: oversubscription
    - plugins:
      - name: drf
      - name: predicates
      - name: nodeorder
      - name: binpack
    - plugins:
      - name: cce-gpu-topology-predicate
      - name: cce-gpu-topology-priority
      - name: cce-gpu
...

```

**Step 3** Enable node oversubscription.

A label can be configured to use oversubscribed resources only after the oversubscription feature is enabled for a node. Related nodes can be created only in a node pool. To enable the oversubscription feature, perform the following steps:

1. Create a node pool.

2. Choose **Manage** in the **Operation** column of the created node pool.
3. On the **Manage Components** page, enable **Node oversubscription feature (over-subscription-resource)** and click **OK**.

**Step 4** Set the node oversubscription label.

The **volcano.sh/oversubscription** label needs to be configured for an oversubscribed node. If this label is set for a node and the value is **true**, the node is an oversubscribed node. Otherwise, the node is not an oversubscribed node.

```
kubectl label node 192.168.0.0 volcano.sh/oversubscription=true
```

An oversubscribed node also supports the oversubscription thresholds, as listed in [Table 6-14](#). For example:

```
kubectl annotate node 192.168.0.0 volcano.sh/evicting-cpu-high-watermark=70
```

Querying the node information

```
# kubectl describe node 192.168.0.0
Name:          192.168.0.0
Roles:        <none>
Labels:       ...
              volcano.sh/oversubscription=true
Annotations:  ...
              volcano.sh/evicting-cpu-high-watermark: 70
```

**Table 6-14** Node oversubscription annotations

Parameter	Description
volcano.sh/evicting-cpu-high-watermark	Upper limit for CPU usage. When the CPU usage of a node exceeds the specified value, offline job eviction is triggered and the node becomes unschedulable. The default value is <b>80</b> , indicating that offline job eviction is triggered when the CPU usage of a node exceeds 80%.
volcano.sh/evicting-cpu-low-watermark	Lower limit for CPU usage. When the CPU usage of a node is higher than the upper limit, offline jobs will be evicted. The node accepts the offline jobs again only when the CPU usage of the node is lower than the lower limit. The default value is <b>30</b> , indicating that offline jobs are accepted again when the CPU usage of a node is lower than 30%.
volcano.sh/evicting-memory-high-watermark	Upper limit for memory usage. When the memory usage of a node exceeds the specified value, offline job eviction is triggered and the node becomes unschedulable. The default value is <b>60</b> , indicating that offline job eviction is triggered when the memory usage of a node exceeds 60%.

Parameter	Description
volcano.sh/evicting-memory-low-watermark	Lower limit for memory usage. When the memory usage of a node is higher than the upper limit, offline jobs will be evicted. The node accepts the offline jobs again only when the memory usage of the node is lower than the lower limit.  The default value is <b>30</b> , indicating that offline jobs are accepted again when the memory usage of a node is less than 30%.
volcano.sh/oversubscription-types	Oversubscribed resource type. Options: <ul style="list-style-type: none"> <li>● <b>cpu</b>: oversubscribed CPU</li> <li>● <b>memory</b>: oversubscribed memory</li> <li>● <b>cpu,memory</b>: oversubscribed CPU and memory</li> </ul> The default value is <b>cpu,memory</b> .

**Step 5** Create resources at a high- and low-priorityClass, respectively.

```
cat <<EOF | kubectl apply -f -
apiVersion: scheduling.k8s.io/v1
description: Used for high priority pods
kind: PriorityClass
metadata:
  name: volcano-production
preemptionPolicy: PreemptLowerPriority
value: 999999
---
apiVersion: scheduling.k8s.io/v1
description: Used for low priority pods
kind: PriorityClass
metadata:
  name: volcano-free
preemptionPolicy: PreemptLowerPriority
value: -90000
EOF
```

**Step 6** Deploy online and offline jobs and configure priorityClasses for these jobs.

The **volcano.sh/qos-level** annotation needs to be added to distinguish offline jobs. The value is an integer ranging from -7 to 7. If the value is less than 0, the job is an offline job. If the value is greater than or equal to 0, the job is an online job. You do not need to set this annotation for online jobs. For both online and offline jobs, set **schedulerName** to **volcano** to enable Volcano.

 **NOTE**

The priorities between online jobs and between offline jobs are not differentiated, and the value validity is not verified. If the value of **volcano.sh/qos-level** of an offline job is not a negative integer ranging from -7 to 0, the job is processed as an online job.

For an offline job:

```
kind: Deployment
apiVersion: apps/v1
spec:
  replicas: 4
  template:
```

```

metadata:
  annotations:
    metrics.alpha.kubernetes.io/custom-endpoints: '[{"api":"","path":"","port":"","names":""}]'
    volcano.sh/qos-level: "-1" # Offline job annotation
  spec:
    schedulerName: volcano # Volcano is used.
    priorityClassName: volcano-free # volcano-free priorityClass
    ...

```

For an online job:

```

kind: Deployment
apiVersion: apps/v1
spec:
  replicas: 4
  template:
    metadata:
      annotations:
        metrics.alpha.kubernetes.io/custom-endpoints: '[{"api":"","path":"","port":"","names":""}]'
    spec:
      schedulerName: volcano # Volcano is used.
      priorityClassName: volcano-production # volcano-production priorityClass
      ...

```

**Step 7** Run the following command to check the number of oversubscribed resources and the resource usage:

`kubectl describe node <nodeIP>`

```

# kubectl describe node 192.168.0.0
Name:          192.168.0.0
Roles:         <none>
Labels:        ...
               volcano.sh/oversubscription=true
Annotations:   ...
               volcano.sh/oversubscription-cpu: 2335
               volcano.sh/oversubscription-memory: 341753856
Allocatable:
  cpu:          3920m
  memory:       6263988Ki
Allocated resources:
(Total limits may be over 100 percent, i.e., overcommitted.)
Resource       Requests      Limits
-----
cpu            4950m (126%) 4950m (126%)
memory        1712Mi (27%) 1712Mi (27%)

```

In the preceding command, CPU and memory are in the unit of m CPU cores and MiB, respectively.

----End

## Deployment Example

The following uses an example to describe how to deploy online and offline jobs in hybrid mode.

**Step 1** Configure a cluster with two nodes, one oversubscribed and the other non-oversubscribed.

```

# kubectl get node
NAME          STATUS  ROLES  AGE  VERSION
192.168.0.173 Ready   <none> 4h58m v1.19.16-r2-CCE22.5.1
192.168.0.3   Ready   <none> 148m  v1.19.16-r2-CCE22.5.1

```

- 192.168.0.173 is an oversubscribed node (with the **volcano.sh/oversubscription=true** label).



- 192.168.0.3 is a non-oversubscribed node (without the **volcano.sh/oversubscription=true** label).

```
# kubectl describe node 192.168.0.173
Name:          192.168.0.173
Roles:         <none>
Labels:        beta.kubernetes.io/arch=amd64
               ...
               volcano.sh/oversubscription=true
```

- Step 2** Submit offline job creation requests. If resources are sufficient, all offline jobs will be scheduled to the oversubscribed node.

The offline job template is as follows:

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: offline
  namespace: default
spec:
  replicas: 2
  selector:
    matchLabels:
      app: offline
  template:
    metadata:
      labels:
        app: offline
      annotations:
        volcano.sh/qos-level: "-1"    # Offline job label
    spec:
      schedulerName: volcano          # Volcano is used.
      priorityClassName: volcano-free # volcano-free priorityClass
      containers:
        - name: container-1
          image: nginx:latest
          imagePullPolicy: IfNotPresent
          resources:
            requests:
              cpu: 500m
              memory: 512Mi
            limits:
              cpu: "1"
              memory: 512Mi
          imagePullSecrets:
            - name: default-secret
```

Offline jobs are scheduled to the oversubscribed node.

```
# kubectl get pod -o wide
NAME                READY STATUS RESTARTS AGE IP          NODE
offline-69cdd49bf4-pmjp8 1/1 Running 0      5s 192.168.10.178 192.168.0.173
offline-69cdd49bf4-z8kxh 1/1 Running 0      5s 192.168.10.131 192.168.0.173
```

- Step 3** Submit online job creation requests. If resources are sufficient, the online jobs will be scheduled to the non-oversubscribed node.

The online job template is as follows:

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: online
  namespace: default
spec:
  replicas: 2
  selector:
    matchLabels:
      app: online
  template:
    metadata:
```

```

labels:
  app: online
spec:
  schedulerName: volcano          # Volcano is used.
  priorityClassName: volcano-production # volcano-production priorityClass
  containers:
  - name: container-1
    image: resource_consumer:latest
    imagePullPolicy: IfNotPresent
    resources:
      requests:
        cpu: 1400m
        memory: 512Mi
      limits:
        cpu: "2"
        memory: 512Mi
    imagePullSecrets:
    - name: default-secret

```

Online jobs are scheduled to the non-oversubscribed node.

```
# kubectl get pod -o wide
NAME          READY STATUS RESTARTS AGE IP          NODE
online-ffb46f656-4mwr6 1/1 Running 0      5s 192.168.10.146 192.168.0.3
online-ffb46f656-dqdv2 1/1 Running 0      5s 192.168.10.67 192.168.0.3
```

**Step 4** Improve the resource usage of the oversubscribed node and observe whether offline job eviction is triggered.

Deploy online jobs to the oversubscribed node (192.168.0.173).

```

apiVersion: apps/v1
kind: Deployment
metadata:
  name: online
  namespace: default
spec:
  replicas: 2
  selector:
    matchLabels:
      app: online
  template:
    metadata:
      labels:
        app: online
    spec:
      affinity:
        nodeAffinity:
          requiredDuringSchedulingIgnoredDuringExecution:
            nodeSelectorTerms:
            - matchExpressions:
              - key: kubernetes.io/hostname
                operator: In
                values:
                - 192.168.0.173
      schedulerName: volcano          # Volcano is used.
      priorityClassName: volcano-production # volcano-production priorityClass
      containers:
      - name: container-1
        image: resource_consumer:latest
        imagePullPolicy: IfNotPresent
        resources:
          requests:
            cpu: 700m
            memory: 512Mi
          limits:
            cpu: 700m
            memory: 512Mi
        imagePullSecrets:
        - name: default-secret

```

Submit the online or offline jobs to the oversubscribed node (192.168.0.173) at the same time.

```
# kubectl get pod -o wide
NAME                READY STATUS RESTARTS AGE IP          NODE
offline-69cdd49bf4-pmjp8 1/1 Running 0      13m 192.168.10.178 192.168.0.173
offline-69cdd49bf4-z8kxh 1/1 Running 0      13m 192.168.10.131 192.168.0.173
online-6f44bb68bd-b8z9p 1/1 Running 0      3m4s 192.168.10.18 192.168.0.173
online-6f44bb68bd-g6xk8 1/1 Running 0      3m12s 192.168.10.69 192.168.0.173
```

Check the oversubscribed node with IP address 192.168.0.173. It is found that resources are oversubscribed, where there are 2343m CPU cores and 3073653200 MiB of memory. Additionally, the CPU allocation rate exceeded 100%.

```
# kubectl describe node 192.168.0.173
Name:                192.168.0.173
Roles:                <none>
Labels:              ...
                    volcano.sh/oversubscription=true
Annotations:         ...
                    volcano.sh/oversubscription-cpu: 2343
                    volcano.sh/oversubscription-memory: 3073653200
                    ...
Allocated resources:
(Total limits may be over 100 percent, i.e., overcommitted.)
Resource            Requests    Limits
-----
cpu                  4750m (121%) 7350m (187%)
memory              3760Mi (61%) 4660Mi (76%)
...

```

Increase the CPU usage of online jobs on the node. Offline job eviction is triggered.

```
# kubectl get pod -o wide
NAME                READY STATUS RESTARTS AGE IP          NODE
offline-69cdd49bf4-bwdm7 1/1 Running 0      11m 192.168.10.208 192.168.0.3
offline-69cdd49bf4-pmjp8 0/1 Evicted 0      26m <none>         192.168.0.173
offline-69cdd49bf4-qpdss 1/1 Running 0      11m 192.168.10.174 192.168.0.3
offline-69cdd49bf4-z8kxh 0/1 Evicted 0      26m <none>         192.168.0.173
online-6f44bb68bd-b8z9p 1/1 Running 0      24m 192.168.10.18 192.168.0.173
online-6f44bb68bd-g6xk8 1/1 Running 0      24m 192.168.10.69 192.168.0.173
```

----End

## Handling Suggestions

- After kubelet of the oversubscribed node is restarted, the resource view of Volcano Scheduler is not synchronized with that of kubelet. As a result, OutOfCPU occurs in some newly scheduled jobs, which is normal. After a period of time, Volcano Scheduler can properly schedule online and offline jobs.
- After online and offline jobs are submitted, you are not advised to dynamically change the job type (adding or deleting annotation volcano.sh/qos-level: "-1") because the current kernel does not support the change of an offline job to an online job.
- CCE collects the resource usage (CPU/memory) of all pods running on a node based on the status information in the cgroups system. The resource usage may be different from the monitored resource usage, for example, the resource statistics displayed by running the **top** command.
- You can add oversubscribed resources (such as CPU and memory) at any time. You can reduce the oversubscribed resource types only when the resource allocation rate does not exceed 100%.

- If an offline job is deployed on a node ahead of an online job and the online job cannot be scheduled due to insufficient resources, configure a higher priorityClass for the online job than that for the offline job.
- If there are only online jobs on a node and the eviction threshold is reached, the offline jobs that are scheduled to the current node will be evicted soon. This is normal.

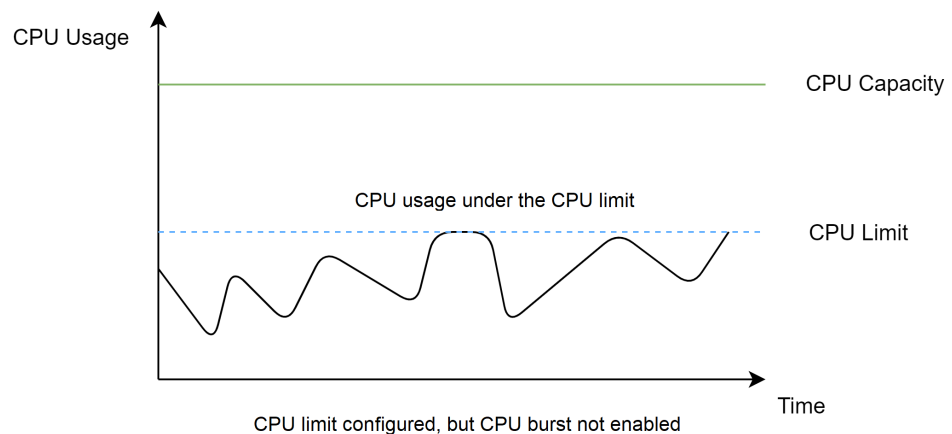
### 6.6.3 CPU Burst

If a CPU limit is set for the containers in a pod, the CPU usage of the containers cannot exceed the limit. Frequent CPU traffic limiting affects service performance and increases the long-tail response latency, especially for latency-sensitive services.

CPU burst is an elastic traffic limiting mechanism that allows temporarily exceeding the CPU limit to reduce the long-tail response time of services. When the CPU quota for a service in each CPU scheduling period is remaining, the system accumulates the CPU quota. If the CPU limit needs to be exceeded in subsequent scheduling periods, the accumulated CPU quota can be used.

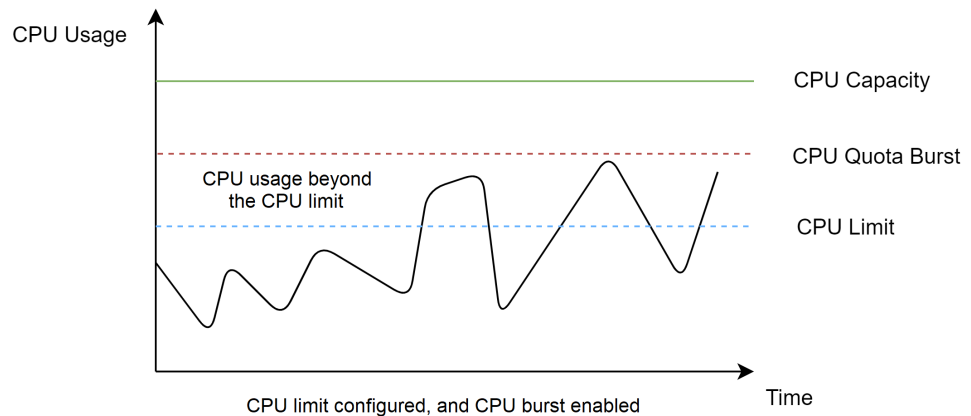
- If CPU burst is not enabled, the CPU quota for a container cannot exceed the limit, and the accumulated burst resources cannot be used.

**Figure 6-19** CPU burst not enabled



- After CPU burst is enabled, the CPU quota for a container can exceed the limit to use the accumulated burst resources.

Figure 6-20 CPU burst enabled



## Notes and Constraints

- Cluster version: CCE Turbo cluster v1.23.5-r0 or later
- OS version: Huawei Cloud EulerOS 2.0
- Volcano: v1.9.0 or later; hybrid deployment: enabled

## Procedure

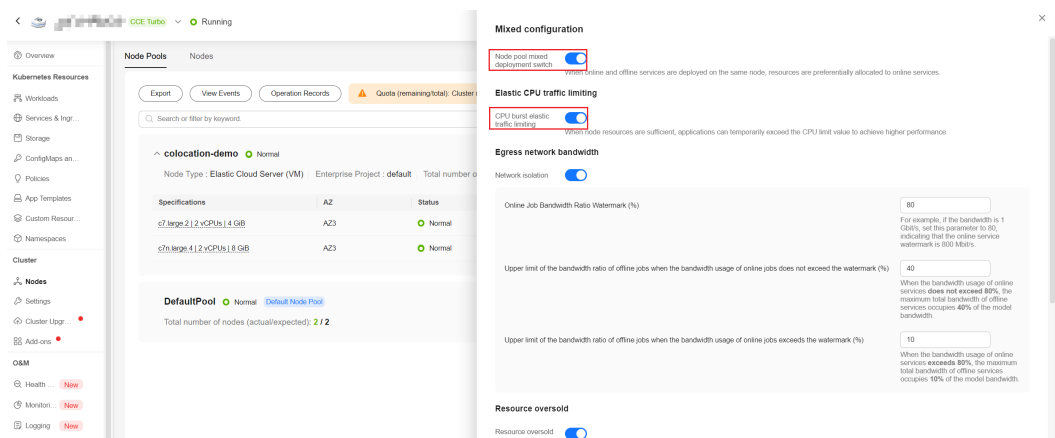
**Step 1** Log in to the CCE console and click the cluster name to access the cluster console.

**Step 2** In the navigation pane, choose **Nodes**. On the **Node Pools** tab page, locate the target node pool and choose **More** > **Mixed configuration**.

Ensure that node pool hybrid deployment and traffic limit for CPU burst are enabled.

### NOTE

After CPU burst is disabled, this function is still enabled on the existing pods where CPU burst has been enabled. Disabling CPU burst takes effect only on new pods.



**Step 3** Deploy a workload in a node pool where hybrid deployment has been enabled. Take Nginx as an example. Set **requests** to **2** and **limits** to **4**, and create a Service that can be accessed in the cluster for the workload.

```

apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx
  namespace: default
spec:
  replicas: 1
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
      annotations:
        volcano.sh/enable-quota-burst: "true"
        volcano.sh/quota-burst-time: "200000"
    spec:
      containers:
        - name: container-1
          image: nginx:latest
          resources:
            limits:
              cpu: "4"
            requests:
              cpu: "2"
          imagePullSecrets:
            - name: default-secret
---
apiVersion: v1
kind: Service
metadata:
  name: nginx
  namespace: default
  labels:
    app: nginx
spec:
  selector:
    app: nginx
  ports:
    - name: cce-service-0
      targetPort: 80
      nodePort: 0
      port: 80
      protocol: TCP
  type: ClusterIP

```

Annotation	Mandatory	Description
volcano.sh/enable-quota-burst	Yes	CPU burst is enabled for the workload.

Annotation	Mandatory	Description
volcano.sh/quota-burst-time	No	To ensure CPU scheduling stability and reduce contention when multiple containers encounter CPU bursts at the same time, the default <b>CPU Burst</b> value is the same as the <b>CPU Quota</b> value, indicating that a container can use a maximum of twice the <b>CPU Limit</b> value. By default, <b>CPU Burst</b> is set for all service containers in a pod.  For example, if the <b>CPU Limit</b> of a container is 4, the default <b>CPU Burst</b> value is 400000 (1 core = 100000), indicating that a maximum of four additional CPU cores can be used after the value of CPU limit is reached.

**Step 4** Verify CPU burst.

You can use the wrk tool to increase load of the workload and observe the service latency, traffic limiting, and CPU limit exceeding when CPU Burst is enabled and disabled, respectively.

1. Run the following command to increase load of the pod. *<service\_ip>* indicates the service IP address associated with the pod.

```
# Download and install the wrk tool on the node.
# The Gzip compression module is enabled in the Apache configuration to simulate the computing logic for the server to process requests.
# Run the following command to increase the load. Note that you need to change the IP address of the target application.
wrk -H "Accept-Encoding: deflate, gzip" -t 4 -c 28 -d 120 --latency --timeout 2s http://<service_ip>
```

2. Obtain the pod ID.
3. You can run the following commands on the node to view the traffic limiting status and CPU limit exceeding status. In the command, *<pod\_id>* indicates the pod ID.

```
cat /sys/fs/cgroup/cpu/kubepods/burstable/pod<pod_id>/cpu.stat
```

Information similar to the following is displayed:

```
nr_periods 0 # Number of scheduling periods
nr_throttled 0 # Traffic limiting times
throttled_time 0 # Traffic limiting duration (ns)
nr_bursts 0 # CPU Limit exceeding times
burst_time 0 # Total Limit exceeding duration
```

**Table 6-15** Result summary in this example

CPU Burst	P99 latency	nr_throttled Traffic Limiting Times	throttled_time Traffic Limiting Duration	nr_bursts Limit Exceeding Times	bursts_time Total Limit Exceeding Duration
Not enabled	2.96 ms	986	14.3s	0	0

CPU Burst	P99 latency	nr_throttled Traffic Limiting Times	throttled_time Traffic Limiting Duration	nr_bursts Limit Exceeding Times	bursts_time Total Limit Exceeding Duration
Enabled	456 μs	0	0	469	3.7s

----End

## 6.6.4 Guaranteed Egress Network Bandwidth

An egress network bandwidth is guaranteed by configuring network priorities for the following scenarios:

- The egress network bandwidth used by online and offline services is balanced to ensure sufficient network bandwidth for online services. When the threshold is reached for online services, the bandwidth usage of offline services will be reduced.
- When online services occupy a small number of network resources, offline services can use more bandwidth. When online services occupy a large number of network resources, the resource usage of offline services will be reduced to ensure that more network bandwidth prioritizes online services.

### Notes and Constraints

To use a guaranteed egress network bandwidth, ensure the following requirements are met:

- Only nodes running Huawei Cloud EulerOS 2.0 are supported.
- Only CCE Turbo clusters of v1.23 or later are supported.
- Volcano of v1.9.0 or later must be installed in the cluster, and hybrid deployment must be enabled by setting **colocation\_enable** in the advanced settings to **true**.
- Before enabling, modifying, or disabling a guaranteed egress network bandwidth, ensure Volcano is working properly.
- For pods that have been running on the node before Volcano is installed, manually restart the pods after enabling guaranteed network bandwidth so that the feature can take effect.
- Uninstalling Volcano or disabling hybrid deployment does not affect the guaranteed egress network bandwidth settings already existing on the node. To disable the guaranteed egress network bandwidth, disable network isolation.
- If bandwidth limit is enabled, the protocol stack cache may be stacked. For protocols without backpressure mechanisms, such as UDP, packet loss and ENOBUFFS may occur.
- Bandwidth limit increases the risk that offline services cannot obtain bandwidth. Services may even be abnormal due to insufficient bandwidth or pod health check may fail.



- Egress network bandwidth guarantee is not prioritized in the following scenarios:
  - When **network bandwidth limit** is used for hybrid online or offline pods, the priority of network bandwidth limit is higher than that of the current function.
  - When a pod uses the node network (**hostNetwork**), the egress network bandwidth guarantee function does not take effect.

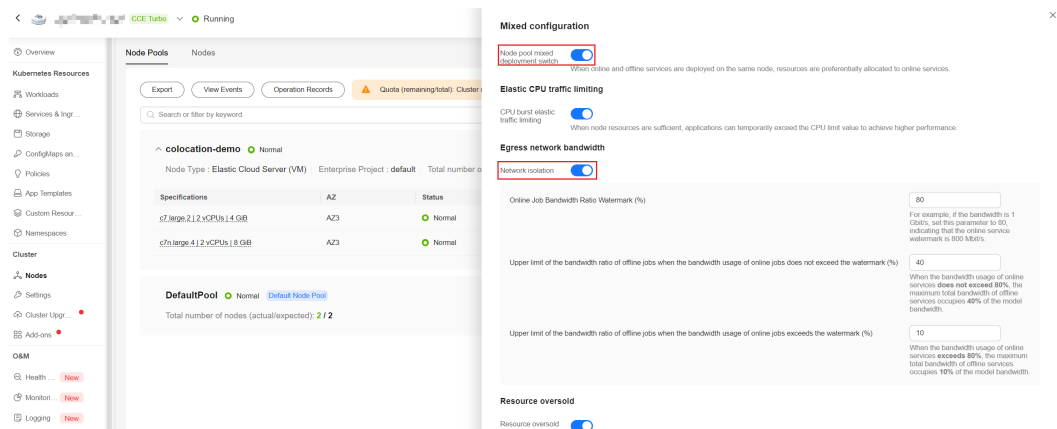
## Procedure

The following describes how to enable or disable egress network bandwidth guarantee.

**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 **Nodes**. On the **Node Pools** tab page, locate the target node pool and choose **More > Mixed configuration**.

Ensure that node pool hybrid deployment and network isolation are enabled.



**Step 3** (Optional) Modify parameters for egress network bandwidth guarantee.

### NOTE

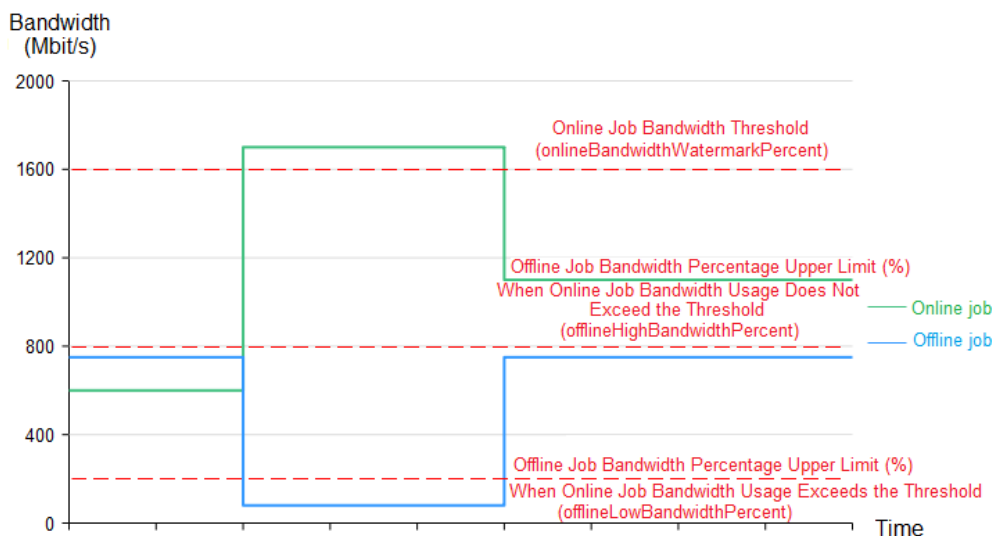
The modified parameters take effect for all nodes running Huawei Cloud EulerOS 2.0 in the cluster.

**Table 6-16** Network isolation parameters

Parameter	Parameter	Description	Default Value	Configuration Range
Online Job Bandwidth Threshold (%)	onlineBandwidthWatermarkPercent	Ratio of the total bandwidth threshold of online services to the assured bandwidth of the node type  <i>Total bandwidth threshold of online services = Assured bandwidth of the node type x onlineBandwidthWatermarkPercent/100</i>	80	Value range: 1 to 1000  <b>NOTE</b> The actual network bandwidth may be larger than the assured bandwidth but less than the maximum bandwidth.
Offline Job Bandwidth Percentage Upper Limit (%) When Online Job Bandwidth Usage Does Not Exceed the Threshold	offlineHighBandwidthPercent	Ratio of the maximum total bandwidth usage of offline services to the assured bandwidth of the node type when the bandwidth usage of online services does not exceed the threshold.  If the total bandwidth usage of online services on the same node does not exceed the value of <i>Assured bandwidth of the node type x onlineBandwidthWatermarkPercent/100</i> , the total bandwidth usage of offline services on the same node cannot exceed the value of <i>Assured bandwidth of the node type x offlineHighBandwidthPercent/100</i> .	40	Therefore, the value can be greater than 100.

Parameter	Parameter	Description	Default Value	Configuration Range
Offline Job Bandwidth Percentage Upper Limit (%) When Online Job Bandwidth Usage Exceeds the Threshold	offlineLowBandwidthPercent	Ratio of the maximum total bandwidth usage of offline services to the assured bandwidth of the node type when the bandwidth usage of online services exceeds the threshold.  If the total bandwidth usage of online services on the same node exceeds the value of <i>Assured bandwidth of the node type x onlineBandwidthWatermarkPercent/100</i> , the total bandwidth usage of offline services on the same node cannot exceed the value of <i>Assured bandwidth of the node type x offlineLowBandwidthPercent/100</i> .	10	

Figure 6-21 Example of egress network bandwidth guarantee



In the preceding figure, when the bandwidth of the online job is lower than the bandwidth baseline, the bandwidth threshold of the offline job is relatively high, indicating that the offline job can use certain bandwidth. When the bandwidth of the online job exceeds the bandwidth baseline, the bandwidth threshold of the

offline job will be lowered accordingly to reduce the bandwidth used by the offline job so that a higher bandwidth can be reserved for the online job.

**----End**

# 7 Network

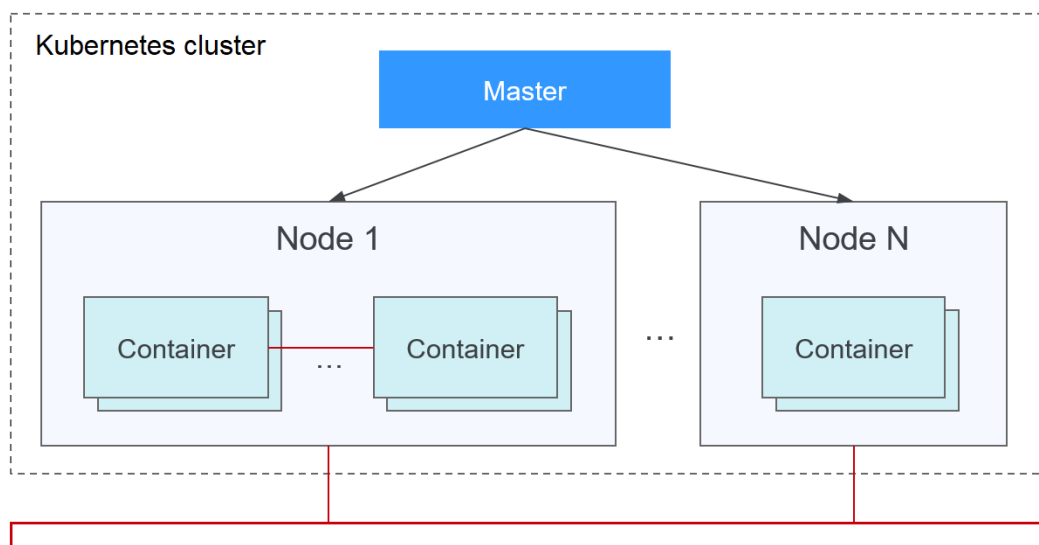
## 7.1 Overview

You can learn about a cluster network from the following two aspects:

- What is a cluster network like? A cluster consists of multiple nodes, and pods (or containers) are running on the nodes. Nodes and containers need to communicate with each other. For details about the cluster network types and their functions, see [Cluster Network Structure](#).
- How is pod access implemented in a cluster? Accessing a pod or container is a process of accessing services of a user. Kubernetes provides [Service](#) and [Ingress](#) to address pod access issues. This section summarizes common network access scenarios. You can select the proper scenario based on site requirements. For details about the network access scenarios, see [Access Scenarios](#).

### Cluster Network Structure

All nodes in the cluster are located in a VPC and use the VPC network. The container network is managed by dedicated network add-ons.



- **Node Network**

A node network assigns IP addresses to hosts (nodes in the figure above) in a cluster. Select a VPC subnet as the node network of the CCE cluster. The number of available IP addresses in a subnet determines the maximum number of nodes (including master nodes and worker nodes) that can be created in a cluster. This quantity is also affected by the container network. For details, see the container network model.

- **Container Network**

A container network assigns IP addresses to pods in a cluster. CCE inherits the IP-Per-Pod-Per-Network network model of Kubernetes. That is, each pod has an independent IP address on a network plane and all containers in a pod share the same network namespace. All pods in a cluster exist in a directly connected flat network. They can access each other through their IP addresses without using NAT. Kubernetes only provides a network mechanism for pods, but does not directly configure pod networks. The configuration of pod networks is implemented by specific container network add-ons. The container network add-ons are responsible for configuring networks for pods and managing container IP addresses.

Currently, CCE supports the following container network models:

- **Container tunnel network:** The container tunnel network is constructed on but independent of the node network through tunnel encapsulation. This network model uses VXLAN to encapsulate Ethernet packets into UDP packets and transmits them in tunnels. Open vSwitch serves as the backend virtual switch.
- **VPC network:** The VPC network model seamlessly combines VPC routing with the underlying network, making it ideal for high-performance scenarios. However, the maximum number of nodes allowed in a cluster is determined by the VPC route quota. Each node is assigned a CIDR block of a fixed size. The VPC network model outperforms the container tunnel network model in terms of performance because it does not have tunnel encapsulation overhead. In addition, as VPC routing includes routes to node IP addresses and the container CIDR block, container pods in a cluster can be directly accessed from outside the cluster.

The performance, networking scale, and application scenarios of a container network vary according to the container network model. For details about the functions and features of different container network models, see [Overview](#).

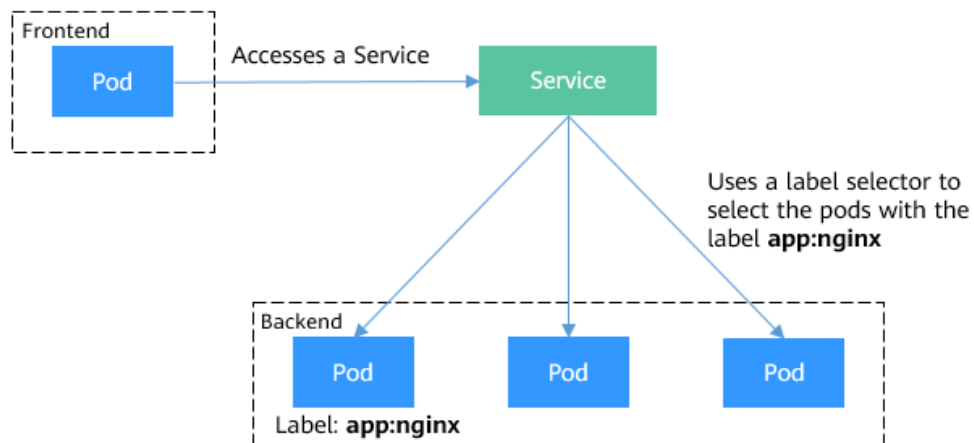
- **Service Network**

Service is also a Kubernetes object. Each Service has a static IP address. When creating a cluster on CCE, you can specify the Service CIDR block. The Service CIDR block cannot overlap with the node or container CIDR block. The Service CIDR block can be used only within a cluster.

## Service

A Service is used for pod access. With a static IP address, a Service forwards access traffic to pods and performs load balancing for these pods.

**Figure 7-1** Accessing pods through a Service



You can configure the following types of Services:

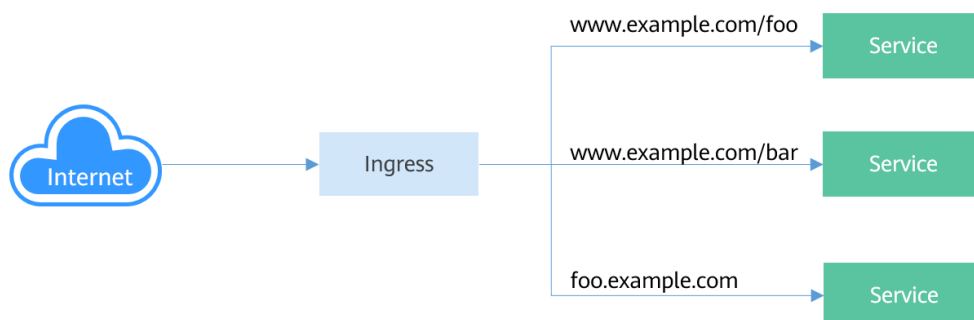
- ClusterIP: used to make the Service only reachable from within a cluster.
- NodePort: used for access from outside a cluster. A NodePort Service is accessed through the port on the node.
- LoadBalancer: used for access from outside a cluster. It is an extension of NodePort, to which a load balancer routes, and external systems only need to access the load balancer.

For details about the Service, see [Overview](#).

## Ingress

Services forward requests using layer-4 TCP and UDP protocols. Ingresses forward requests using layer-7 HTTP and HTTPS protocols. Domain names and paths can be used to achieve finer granularities.

**Figure 7-2** Ingress and Service



For details about the ingress, see [Overview](#).

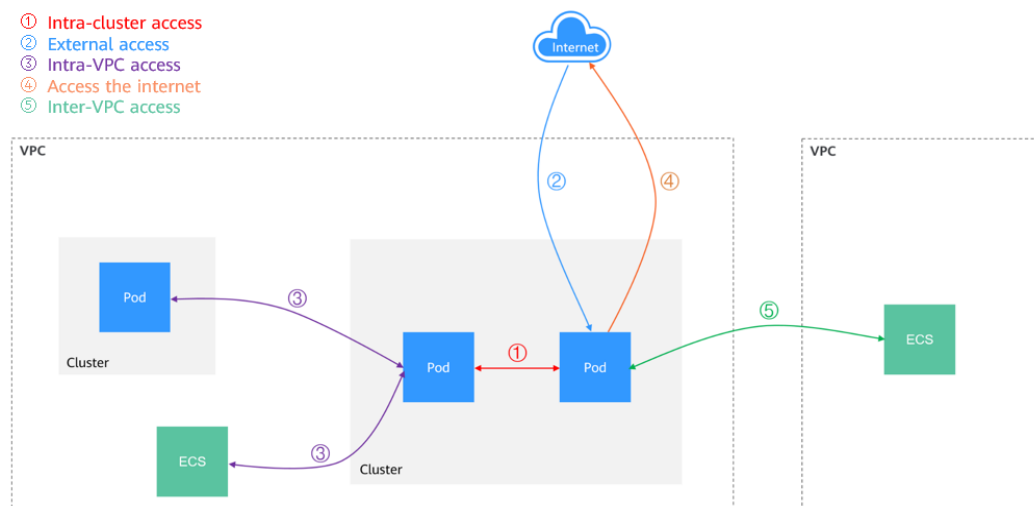
## Access Scenarios

Workload access scenarios can be categorized as follows:

- Intra-cluster access: A ClusterIP Service is used for workloads in the same cluster to access each other.

- Access from outside a cluster: A Service (NodePort or LoadBalancer type) or an ingress is recommended for a workload outside a cluster to access workloads in the cluster.
  - Access through the public network: An EIP should be bound to the node or load balancer.
  - Access through the private network: The workload can be accessed through the internal IP address of the node or load balancer. If workloads are located in different VPCs, a peering connection is required to enable communication between different VPCs.
- The workload can access the external network as follows:
  - Accessing an intranet: The workload accesses the intranet address, but the implementation method varies depending on container network models. Ensure that the peer security group allows the access requests from the container CIDR block.
  - Accessing a public network: Assign an EIP to the node where the workload runs, or configure SNAT rules through the NAT gateway. For details, see [Accessing the Internet from a Container](#).

**Figure 7-3** Network access diagram



## 7.2 Container Network

### 7.2.1 Overview

The container network assigns IP addresses to pods in a cluster and provides networking services. In CCE, you can select the following network models for your cluster:

- [Cloud Native Network 2.0](#)
- [VPC network](#)
- [Tunnel network](#)



## Network Model Comparison

**Table 7-1** describes the differences of network models supported by CCE.

---

 **CAUTION**

After a cluster is created, the network model cannot be changed.

---

**Table 7-1** Network model comparison

Dimension	Tunnel Network	VPC Network	Cloud Native Network 2.0
Application scenarios	<ul style="list-style-type: none"> <li>• 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.</li> <li>• 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</li> </ul>	<ul style="list-style-type: none"> <li>• 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.</li> <li>• Small- and medium-scale networks: Due to the limitation on VPC routing tables, it is recommended that the number of nodes in a cluster be less than or equal to 1000.</li> </ul>	<ul style="list-style-type: none"> <li>• 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.</li> <li>• Large-scale networking: Cloud Native 2.0 networks support a maximum of 2,000 ECS nodes and 100,000 pods.</li> </ul>

Dimension	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 technology	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.
Interconnecting 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	<ul style="list-style-type: none"> <li>Separate container CIDR blocks needed</li> <li>Container CIDR blocks divided by node and dynamically added after being allocated</li> </ul>	<ul style="list-style-type: none"> <li>Separate container CIDR blocks needed</li> <li>Container CIDR blocks divided by node and statically allocated (the allocated CIDR blocks cannot be changed after a node is created)</li> </ul>	Container CIDR blocks divided from a VPC subnet (You do not need to configure separate container CIDR blocks.)

Dimension	Tunnel Network	VPC Network	Cloud Native Network 2.0
Network performance	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
Networking scale	A maximum of 2000 nodes are supported.	Suitable for small- and medium-scale networks due to the limitation on VPC routing 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 routing tables. Evaluate the cluster scale that is limited by the VPC routing tables before creating the cluster.	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.

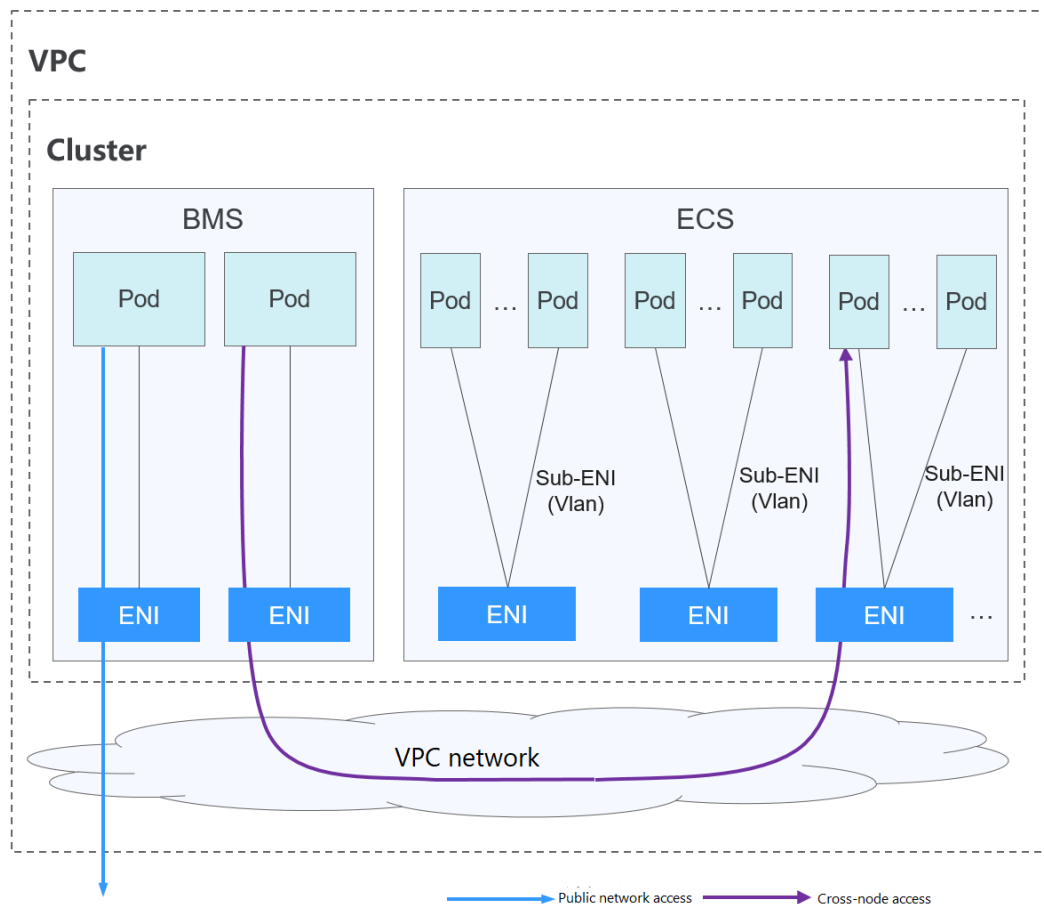
## 7.2.2 Cloud Native Network 2.0 Settings

### 7.2.2.1 Cloud Native 2.0 Network Model

#### Model Definition

Cloud Native 2.0 network model is a proprietary, next-generation container network model that combines the elastic network interfaces (ENIs) and supplementary network interfaces (sub-ENIs) of the Virtual Private Cloud (VPC). This allows ENIs or sub-ENIs to be directly bound to pods, giving each pod its own unique IP address within the VPC. Furthermore, it supports additional features like ELB passthrough container, pod binding to a security group, and pod binding to an EIP. Because container tunnel encapsulation and NAT are not required, the Cloud Native 2.0 network model enables higher network performance than the container tunnel network model and VPC network model.

**Figure 7-4** Cloud Native 2.0 network model



In a cluster using the Cloud Native 2.0 network model, pods rely on ENIs or sub-ENIs to connect to external networks.

- Pods running on BMS nodes use ENIs.
- Pods running on ECS nodes use sub-ENIs that are bound to the ECS' ENIs through VLAN sub-interfaces.
- To run a pod, it is necessary to bind ENIs to it. The number of pods that can run on a node depends on the number of ENIs that can be bound to the node and the number of ENI ports available on the node.
- Traffic for communications between pods on a node, communications between pods on different nodes, and access to networks outside a cluster is forwarded through the ENI or sub-ENI of the VPC.

## Notes and Constraints

This network model is available only to CCE Turbo clusters.

## Advantages and Disadvantages

### Advantages

- VPCs serve as the foundation for constructing container networks. Every pod has its own network interface and IP address, which simplifies network problem-solving and enhances performance.
- In the same VPC, ENIs are directly bound to pods in a cluster, so that resources outside the cluster can directly communicate with containers within the cluster.

 **NOTE**

Similarly, if the VPC is accessible to other VPCs or data centers, resources in other VPCs or data centers can directly communicate with containers in the cluster, provided there are no conflicts between the network CIDR blocks.

- The load balancing, security group, and EIP capabilities provided by VPC can be directly used by pods.

### Disadvantages

Container networks are built on VPCs, with each pod receiving an IP address from the VPC CIDR block. As a result, it is crucial to plan the container CIDR block carefully before creating a cluster.

## Application Scenarios

- 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.

## Recommendation for CIDR Block Planning

As explained in [Cluster Network Structure](#), network addresses in a cluster are divided into the cluster network, container network, and service network. When planning network addresses, consider the following factors:

- All subnets (including extended subnets) in the VPC where the cluster resides cannot conflict with the Service CIDR blocks.
- Ensure that **each CIDR block has sufficient IP addresses**.
  - The IP addresses in the cluster CIDR block must match the cluster scale. Otherwise, nodes cannot be created due to insufficient IP addresses.
  - The IP addresses in the container CIDR block must match the service scale. Otherwise, pods cannot be created due to insufficient IP addresses.

In the Cloud Native 2.0 network model, the container CIDR block and node CIDR block share the network IP addresses in a VPC. It is recommended that the container subnet and node subnet not use the same subnet. Otherwise, containers or nodes may fail to be created due to insufficient IP addresses.

In addition, a subnet can be added to the container CIDR block after a cluster is created to increase the number of available IP addresses. In this case, ensure that the added subnet does not conflict with other subnets in the container CIDR block.

## Example of Cloud Native Network 2.0 Access

In this example, a CCE Turbo cluster is created, and the cluster contains three ECS nodes.

You can check the basic information about a node on the ECS console. You can see that a primary network interface and an extended network interface are bound to the node. Both of them are ENIs. The IP address of the extended network interface belongs to the container CIDR block and is used to bind sub-ENIs to pods on the node.

The following is an example of creating a workload in a cluster using the Cloud Native 2.0 network model:

**Step 1** Use `kubectl` to access the cluster. For details, see [Connecting to a Cluster Using kubectl](#).

**Step 2** Create a Deployment in the cluster.

Create the **deployment.yaml** file. The following shows an example:

```
kind: Deployment
apiVersion: apps/v1
metadata:
  name: example
  namespace: default
spec:
  replicas: 6
  selector:
    matchLabels:
      app: example
  template:
    metadata:
      labels:
        app: example
    spec:
      containers:
        - name: container-0
          image: 'nginx:perl'
      resources:
        limits:
          cpu: 250m
          memory: 512Mi
        requests:
          cpu: 250m
          memory: 512Mi
      imagePullSecrets:
        - name: default-secret
```

Create the workload.

```
kubectl apply -f deployment.yaml
```

**Step 3** Check the running pods.

```
kubectl get pod -owide
```

Command output:

NAME	READY	STATUS	RESTARTS	AGE	IP	NODE	NOMINATED NODE
READINESS GATES							
example-5bdc5699b7-54v7g	1/1	Running	0	7s	10.1.18.2	10.1.0.167	<none> <none>
example-5bdc5699b7-6dzx5	1/1	Running	0	7s	10.1.18.216	10.1.0.186	<none> <none>
example-5bdc5699b7-gq7xs	1/1	Running	0	7s	10.1.16.63	10.1.0.144	<none> <none>
example-5bdc5699b7-h9rvb	1/1	Running	0	7s	10.1.16.125	10.1.0.167	<none> <none>
example-5bdc5699b7-s9fts	1/1	Running	0	7s	10.1.16.89	10.1.0.144	<none> <none>

```
example-5bdc5699b7-swq6q 1/1 Running 0 7s 10.1.17.111 10.1.0.167 <none>  
<none>
```

The IP addresses of all pods are sub-ENIs, which are bound to the ENI (extended network interface) of the node.

For example, the IP address of the extended network interface of **node 10.1.0.167** is **10.1.17.172**. On the network interfaces console, you can see that there are three sub-ENIs bound to the extended network interface whose IP address is **10.1.17.172**. These sub-ENIs are the IP addresses of the pods running on the node.

**Step 4** Log in to an ECS in the same VPC and access the IP address of a pod from outside the cluster.

In this example, the accessed pod IP address is *10.1.18.2*.

```
curl 10.1.18.2
```

If the following information is displayed, the workload can be properly accessed:

```
<!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>  
<p>If you see this page, the nginx web server is successfully installed and  
working. Further configuration is required.</p>  
  
<p>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>.</p>  
  
<p><em>Thank you for using nginx.</em></p>  
</body>  
</html>
```

----End

## 7.2.2.2 Configuring Pod Subnets of a Cluster

### Scenario

If the pod subnet configured during CCE Turbo cluster creation cannot meet service expansion requirements, you can add a pod subnet for the cluster.

### Notes and Constraints

- This function is available only for CCE Turbo clusters of v1.19 or later.

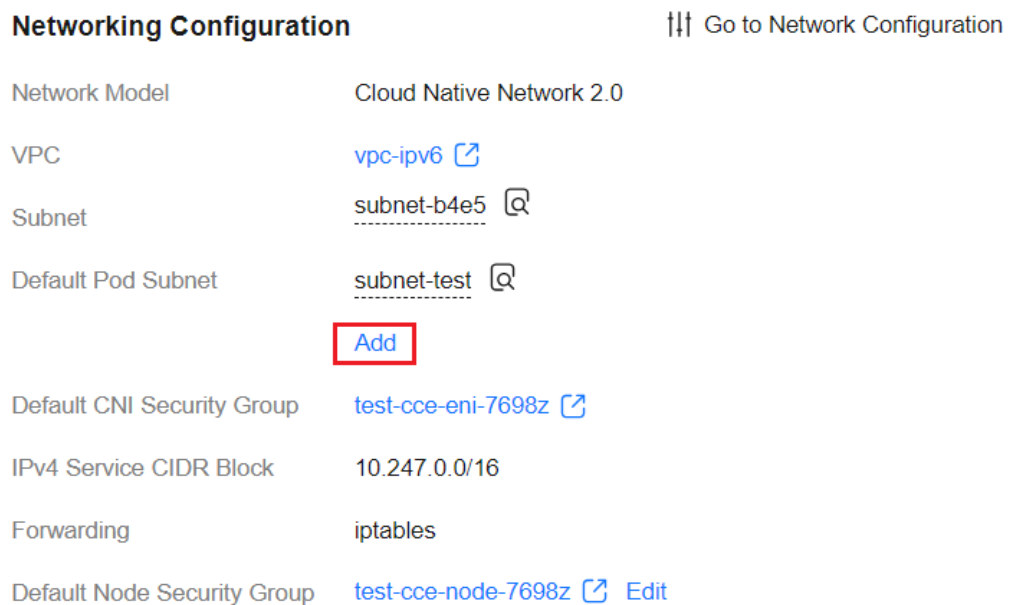
### Procedure

**Step 1** Log in to the CCE console and access the CCE Turbo cluster console.



**Step 2** On the **Overview** page, locate the **Networking Configuration** area and click **Add**.

**Figure 7-5** Adding a pod subnet



**Step 3** Select a pod subnet in the same VPC. You can add multiple pod subnets at a time. If no other pod subnet is available, go to the VPC console and create one.

**Figure 7-6** Selecting a pod subnet



**Step 4** Click **OK**.

----End

### 7.2.2.3 Binding a Security Group to a Workload Using a Security Group Policy

In Cloud Native Network 2.0, pods use VPC ENIs or sub-ENIs for networking. You can directly bind security groups and EIPs to pods. To bind CCE pods with security groups, CCE provides a custom resource object named **SecurityGroup**. Using this resource object, you can customize security isolation for workloads.

 NOTE

The priority of the security group bound to pods using the security group policy is higher than that of the security group in the [NetworkAttachmentDefinition](#).

## Notes and Constraints

- This function is supported for CCE Turbo clusters of v1.19 and later. Upgrade your CCE Turbo clusters if their versions are earlier than v1.19.
- A workload can be bound to a maximum of five security groups.

## Using the Console

- 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 **Workloads**. On the displayed page, click the desired workload name.
- Step 3** Switch to the **SecurityGroups** tab and click **Create**.
- Step 4** Set the parameters as described in [Table 7-2](#).

**Table 7-2** Configuration parameters

Parameter	Description	Example
Security Group Policy Name	Enter a security policy name. Enter 1 to 63 characters. The value must start with a lowercase letter and cannot end with a hyphen (-). Only lowercase letters, digits, and hyphens (-) are allowed.	security-group
Associate Security Group	The selected security group will be bound to the ENI or supplementary ENI of the selected workload. A maximum of five security groups can be selected from the drop-down list. You must select one or multiple security groups to create a SecurityGroup.  If no security group has not been created, click <b>Create Security Group</b> . After the security group is created, click the refresh button.  <b>NOTICE</b> <ul style="list-style-type: none"> <li>• A maximum of five security groups can be selected.</li> <li>• Hover the cursor on the security group name, and you can obtain details about the security group.</li> </ul>	64566556-bd6f-48fb-b2c6-df8f44617953  5451f1b0-bd6f-48fb-b2c6-df8f44617953

**Step 5** After setting the parameters, click **OK**.

After the security group is created, the system automatically returns to the security group list page where you can see the new security group.

----End

## Using kubectl

**Step 1** Use kubectl to access the cluster. For details, see [Connecting to a Cluster Using kubectl](#).

**Step 2** Create a description file named **securitygroup-demo.yaml**.

### vi securitygroup-demo.yaml

For example, create the following SecurityGroup to bind all nginx workloads with two security groups 64566556-bd6f-48fb-b2c6-df8f44617953 and 5451f1b0-bd6f-48fb-b2c6-df8f44617953 that have been created in advance. An example is as follows:

```
apiVersion: crd.yangtse.cni/v1
kind: SecurityGroup
metadata:
  name: demo
  namespace: default
spec:
  podSelector:
    matchLabels:
      app: nginx
  securityGroups:
    - id: 64566556-bd6f-48fb-b2c6-df8f44617953
    - id: 5451f1b0-bd6f-48fb-b2c6-df8f44617953
```

[Table 7-3](#) describes the parameters in the YAML file.

**Table 7-3** Description

Field	Description	Mandatory
apiVersion	API version. The value is <b>crd.yangtse.cni/v1</b> .	Yes
kind	Type of the object to be created.	Yes
metadata	Metadata definition of the resource object.	Yes
name	Name of the SecurityGroup.	Yes
namespace	Name of the namespace.	Yes
spec	Detailed description of the SecurityGroup.	Yes
podSelector	Used to define the workload to be associated with security groups in the SecurityGroup.	Yes
securityGroups	Security group ID.	Yes

**Step 3** Run the following command to create the SecurityGroup:

**kubectl create -f securitygroup-demo.yaml**

If the following information is displayed, the SecurityGroup is being created.

```
securitygroup.crd.yangtse.cni/demo created
```

**Step 4** Run the following command to check the SecurityGroup:

**kubectl get sg**

If the name of the created SecurityGroup is **demo** in the command output, the SecurityGroup is created successfully.

NAME	POD-SELECTOR	AGE
all-no	map[matchLabels:map[app:nginx]]	4h1m
s001test	map[matchLabels:map[app:nginx]]	19m
demo	map[matchLabels:map[app:nginx]]	2m9s

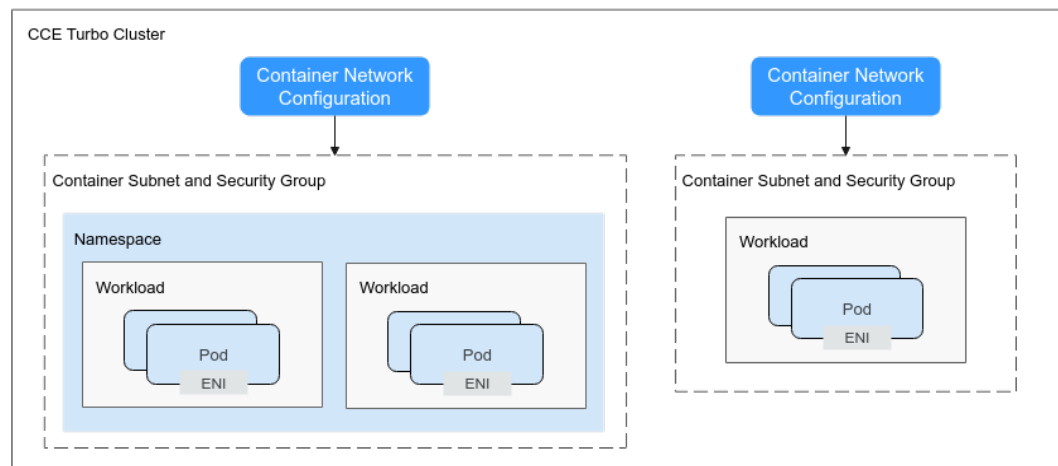
----End

### 7.2.2.4 Binding a Subnet and Security Group to a Namespace or Workload Using a Container Network Configuration

#### Scenario

In a CCE Turbo cluster, you can configure subnets and security groups for containers by namespace or workload using NetworkAttachmentDefinition [CRDs](#). To configure a particular container subnet and security group for a specified namespace or workload, create a container network configuration (NetworkAttachmentDefinition) and associate it with the target namespace or workload. In this way, service subnets can be planned or services can be securely isolated.

**Figure 7-7** Custom container network settings



The following table lists the resources that a container network configuration can be associated with.

**Table 7-4** Associated resources

Category	Resources a Container Network Configuration Can Associate with	
	Namespace	Workload
Subnet and security group configurations	All workloads created in the namespace associated with a container network configuration will use the same subnet and security group configurations.	The workloads using the same container network configuration will use the same subnet and security group configurations.
Supported cluster versions	Available only in CCE Turbo clusters of 1.23.8-r0, 1.25.3-r0, or later.	Available only in CCE Turbo clusters of 1.23.11-r0, 1.25.6-r0, 1.27.3-r0, 1.28.1-r0, or later.
Notes and Constraints	A namespace can be associated with only one container network configuration.	Only the custom container network configurations that are not associated with any namespace can be specified.

 **NOTE**

- The priorities (in descending order) of the container network configurations used by a pod are as follows: Container network configuration directly associated with the pod > Container network configuration associated with the pod namespace > Default container network configuration of the cluster (**default-network**)
- If **default-network** is available in a cluster, it takes effect on all pods where no custom container network configuration has been configured. The default container subnet in the network settings on the **Overview** page is the container subnet in **default-network**. **default-network** cannot be deleted.
- If there is only one container network configuration in a cluster, it is the default container network configuration. If there are multiple container network configurations in a cluster, the one with annotation **yangtse.io/default-network: true** is the default container network configuration. If there is no default container network configuration in a cluster, the pod that is not associated with any container network configuration fails to be start after being created because no NIC can be allocated for the pod.

## Notes and Constraints

- Only the default container network configuration supports dynamic pre-binding of container NICs. When the quota of node NICs is used up, the pod that uses the custom container network configuration attempts to unbind the pre-bound NIC of the default container network configuration, leading to slower pod startup. Therefore, if you need to frequently use the custom container network configuration, disable dynamic pre-binding of global container NICs in the target cluster. If you require high-speed pod elasticity using the default container network configuration, properly plan dynamic pre-binding of container NICs in the target node pool based on scheduling.
- If a workload with a fixed IP address needs to be associated with a new container network configuration, the fixed IP address will be invalid when

Pods are rebuilt. In this case, delete the workload, release the fixed IP address, and create a workload again.

- Before deleting a custom container network configuration, delete the pods (with the **cni.yangtse.io/network-status** annotation) created using the configuration in the target namespace. For details, see [Deleting a Container Network Configuration](#).

## Creating a Container Network Configuration of the Namespace Type

After such a container network configuration is created and associated with a namespace, all workloads created in the namespace will use the same subnet and security group configurations.

## Operations on the Console for the Namespace Type

**Step 1** Log in to the CCE console.

**Step 2** Click the cluster name to access the cluster console. Choose **Settings** in the navigation pane. In the right pane, click the **Network** tab.

**Step 3** View the **Container Network Security Policy Configuration (Namespace Level)** and click **Add**. In the window that is displayed, configure parameters such as the pod subnet and security group.

- **Name:** Enter a name that contains a maximum of 63 characters. Do not use **default-network**, **default**, **mgnt0**, or **mgnt1**.
- **Associated Resource Type:** resource type associated with the custom container network configuration. For details, see [Table 7-4](#). To create a container network configuration of the namespace type, select **Namespace**.
- **Namespace:** Select the namespace to be associated. The namespaces associated with different container network configurations must be unique. If no namespace is available, click **Create Namespace** to create one.
- **Pod Subnet:** Select a subnet. If no subnet is available, click **Create Subnet** to create one. After the subnet is created, click the refresh button.
- **Associate Security Group:** The default value is the container ENI security group. You can also click **Create Security Group** to create one. After the security group is created, click the refresh button. A maximum of five security groups can be selected.

**Figure 7-8** Creating a container network configuration of the namespace type

**Create Network Configurations** ×

**i** 1. Newly created network configurations only take effect for newly created Pods, and existing Pods need to be rebuilt to take effect  
2. In the new configuration, the container subnet does not support dynamic NIC preheating. As a result, the Pods creation speed is slow.

Name:

Associated Resource Type: **Namespace** | Workloads

Namespace:  × ▼ [Create Namespace](#)

Pod Subnet:  × ▼ [Create Subnet](#) ↗

**⚠** The added container subnet cannot be deleted.

Associate Security Group:  ? × ▼ [Create Security Group](#) ↗

**Step 4** Click **OK**. After the creation, you will be redirected to the custom container network configuration list, where the new container network configuration is included.

**Figure 7-9** Container network configuration list

**Custom Container Network Settings**

If you want to configure a specified container CIDR block and security group for a specific namespace or workload, create a custom container network configuration and associate the configuration with the namespace or workload. [Configuration Guide](#)

workload.detail.configurationName	Container Subnet or CIDR Block	Associate Security Group	Associated Resource Type	Associated Resource Name	Operation
<input type="checkbox"/> default-network	subnet-c123 192.168.1.0/24 (IPv4)		Namespace	All namespaces	<a href="#">Update</a> <a href="#">View YAML</a> <a href="#">Delete</a>
<input type="checkbox"/> test	subnet-bc7b 192.168.0.0/24 (IPv4)		Workloads	<a href="#">Go to Workloads</a>	<a href="#">Update</a> <a href="#">Edit YAML</a> <a href="#">Delete</a>

[Add](#)

----End

## Operations Using kubectl for the Namespace Type

This section describes how to use kubectl to create a container network configuration of the namespace type.

**Step 1** Use kubectl to access the cluster. For details, see [Connecting to a Cluster Using kubectl](#).

**Step 2** Modify the **networkattachment-test.yaml** file.

```
vi networkattachment-test.yaml
```

Example file content:

```
apiVersion: k8s.cni.cncf.io/v1
kind: NetworkAttachmentDefinition
metadata:
  annotations:
    yangtse.io/project-id: 05e38**
  name: example
  namespace: kube-system
spec:
  config: |
  {
```

```

"type": "eni-neutron",
"args": {
  "securityGroups": "41891**",
  "subnets": [
    {
      "subnetID": "27d95**"
    }
  ]
},
"selector": {
  "namespaceSelector": {
    "matchLabels": {
      "kubernetes.io/metadata.name": "default"
    }
  }
}
}
}

```

**Table 7-5** Key parameters

Parameter	Mandatory	Type	Description
apiVersion	Yes	String	API version. The value is fixed at <b>k8s.cni.cncf.io/v1</b> .
kind	Yes	String	Type of the object to be created. The value is fixed at <b>NetworkAttachmentDefinition</b> .
yangtse.io/project-id	Yes	String	Project ID in the current region. For details, see <a href="#">Obtaining a Project ID</a> .
name	Yes	String	Configuration item name.
namespace	Yes	String	Namespace of the configuration resource. The value is fixed to <b>kube-system</b> .
config	Yes	<a href="#">Table 7-6</a> object	Configuration content, which is a string in JSON format.

**Table 7-6** config parameters

Parameter	Mandatory	Type	Description
type	Yes	String	The value is fixed at <b>eni-neutron</b> .
args	No	<a href="#">Table 7-7</a> object	Configuration parameters.
selector	No	<a href="#">Table 7-8</a> object	Namespace in which the configuration takes effect.



**Table 7-7** args parameters

Parameter	Mandatory	Type	Description
securityGroups	No	String	<p>Security group ID. If no security group is planned, ensure that the security group is the same as that in <b>default-network</b>.</p> <p>How to obtain: Log in to the VPC console. In the navigation pane, choose <b>Access Control &gt; Security Groups</b>. Click the target security group name and copy the ID on the <b>Summary</b> tab page.</p>
subnets	Yes	Array of subnetID Objects	<p>List of container subnet IDs. At least one subnet ID must be entered. The format is as follows: [{"subnetID":"27d95***"}, {"subnetID":"827bb***"}, {"subnetID":"bdd6b***"}]</p> <p>Subnet ID not used by the cluster in the same VPC.</p> <p>How to obtain: Log in to the VPC console. In the navigation pane, choose <b>Virtual Private Cloud &gt; Subnets</b>. Click the target subnet name and copy the <b>Subnet ID</b> on the <b>Summary</b> tab page.</p>

**Table 7-8** selector parameters

Parameter	Mandatory	Type	Description
namespaceSelector	No	matchLabels object	<p>A Kubernetes standard selector. Enter the namespace label in the following format: "matchLabels":{   "kubernetes.io/metadata.name":"default" }</p> <p>The namespaces of different configurations cannot overlap.</p>

**Step 3** Create the container network configuration.

```
kubectl create -f networkattachment-test.yaml
```

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

```
networkattachmentdefinition.k8s.cni.cncf.io/example created
```

----End

## Creating a Container Network Configuration of the Workload Type

After such a container network configuration is created, all workloads associated with it will use the same subnet and security group configurations.

### Operations on the Console for the Workload Type

- Step 1** Log in to the CCE console.
- Step 2** Click the cluster name to access the cluster console. Choose **Settings** in the navigation pane. In the right pane, click the **Network** tab.
- Step 3** View the **Container Network Security Policy Configuration (Namespace Level)** and click **Add**. In the window that is displayed, configure parameters such as the pod subnet and security group.
  - **Name:** Enter a name that contains a maximum of 63 characters. Do not use **default-network**, **default**, **mgnt0**, or **mgnt1**.
  - **Associated Resource Type:** resource type associated with the custom container network configuration. For details, see [Table 7-4](#). To create a container network configuration of the workload type, select **Workload**.
  - **Pod Subnet:** Select a subnet. If no subnet is available, click **Create Subnet** to create one. After the subnet is created, click the refresh button.
  - **Associate Security Group:** The default value is the container ENI security group. You can also click **Create Security Group** to create one. After the security group is created, click the refresh button. A maximum of five security groups can be selected.

**Figure 7-10** Creating a container network configuration of the workload type

**Create Network Configurations**
×

1. Newly created network configurations only take effect for newly created Pods, and existing Pods need to be rebuilt to take effect

2. In the new configuration, the container subnet does not support dynamic NIC preheating. As a result, the Pods creation speed is slow.

Name

Associated Resource Type Namespace Workloads

After creating a container network configuration for a workload, select the created container network configuration name on the Create Workload page.

Pod Subnet  C [Create Subnet](#)

⚠ The added container subnet cannot be deleted.

Associate Security Group  C [Create Security Group](#)

- Step 4** Click **OK**. After the creation, you will be redirected to the custom container network configuration list, where the new container network configuration is included.

**Figure 7-11** Container network configuration list

**Custom Container Network Settings**  
If you want to configure a specified container CIDR block and security group for a specific namespace or workload, create a custom container network configuration and associate the configuration with the namespace or workload. [Configuration Guide](#)

Delete

<input type="checkbox"/>	workload.detail.configurationName	Container Subnet or CIDR Block	Associate Security Group	Associated Resource Type	Associated Resource Name	Operation
<input type="checkbox"/>	default-network	subnet-c123 192.168.1.0/24 (IPv4)		Namespace	All namespaces	<a href="#">Update</a> <a href="#">View YAML</a> <a href="#">Delete</a>
<input type="checkbox"/>	test	subnet-bc7b 192.168.0.0/24 (IPv4)		Workloads	<a href="#">Go to Workloads</a>	<a href="#">Update</a> <a href="#">Edit YAML</a> <a href="#">Delete</a>

**Step 5** When creating a workload, you can select a custom container network configuration.

1. Choose **Workloads** in the navigation pane. In the right pane, click the **Deployments** tab.
2. Click **Create Workload** in the upper right corner of the page. In the **Advanced Settings** area, choose **Network Configuration** and determine whether to enable a specified container network configuration.
3. Select an existing container network configuration. If no configuration is available, click **Add** to create one.

**Figure 7-12** Selecting a container network configuration

**Advanced Settings**

Upgrade Pod inbound bandwidth limiting  [Introduction to Ingress Bandwidth](#)

Scheduling Pod egress bandwidth limit  [Introduction to Egress Bandwidth](#)

Tolerance

Labels and Annotations Indicates whether to enable the network configuration of a specified container

DNS Enable the specified container network configuration. The workload is created using the container subnet and security group in the specified container network configuration. [Understand the function details and usage restrictions.](#)

API Settings Specifies the container network configuration name.     
 You can select only the custom container network configuration whose associated resource type is workload.

**Network Configuration**

4. After the configuration, click **Create Workload**.  
Return to the **Settings** page. In the container network configuration list, the name of the resource associated with the created container network configuration is displayed.

**Figure 7-13** Resource associated with a container network configuration

**Custom Container Network Settings**  
If you want to configure a specified container CIDR block and security group for a specific namespace or workload, create a custom container network configuration and associate the configuration with the namespace or workload. [Configuration Guide](#)

Delete

<input type="checkbox"/>	workload.detail.configurationName	Container Subnet or CIDR Block	Associate Security Group	Associated Resource Type	Associated Resource Name	Operation
<input type="checkbox"/>	default-network	subnet-c123 192.168.1.0/24 (IPv4)	<a href="#">w00568049-volcano-test-ccc-eni-gqaw</a>	Namespace	All namespaces	<a href="#">Update</a> <a href="#">View YAML</a> <a href="#">Delete</a>
<input type="checkbox"/>	test	subnet-bc7b 192.168.0.0/24 (IPv4)	<a href="#">w00568049-volcano-test-ccc-eni-gqaw</a>	Workloads	<a href="#">nginx1-78d8b4db9c-f69t</a> <a href="#">Go to Workloads</a>	<a href="#">Update</a> <a href="#">Edit YAML</a> <a href="#">Delete</a>

----End

## Operations Using kubectl for the Workload Type

This section describes how to use kubectl to create a container network configuration of the workload type.

**Step 1** Use `kubectl` to access the cluster. For details, see [Connecting to a Cluster Using kubectl](#).

**Step 2** Modify the `networkattachment-test.yaml` file.

```
vi networkattachment-test.yaml
```

Example file content:

```
apiVersion: k8s.cni.cncf.io/v1
kind: NetworkAttachmentDefinition
metadata:
  annotations:
    yangtse.io/project-id: 80d5a**
  name: example
  namespace: kube-system
spec:
  config: |
    {
      "type": "eni-neutron",
      "args": {
        "securityGroups": "f4983**",
        "subnets": [
          {
            "subnetID": "5594b**"
          }
        ]
      }
    }
  }
```

**Table 7-9** Key parameters

Parameter	Mandatory	Type	Description
apiVersion	Yes	String	API version. The value is fixed at <b>k8s.cni.cncf.io/v1</b> .
kind	Yes	String	Type of the object to be created. The value is fixed at <b>NetworkAttachmentDefinition</b> .
yangtse.io/project-id	Yes	String	Project ID in the current region. For details, see <a href="#">Obtaining a Project ID</a> .
name	Yes	String	Configuration item name.
namespace	Yes	String	Namespace of the configuration resource. The value is fixed to <b>kube-system</b> .
config	Yes	<a href="#">Table 7-10</a> object	Configuration content, which is a string in JSON format.

**Table 7-10** config parameters

Parameter	Mandatory	Type	Description
type	Yes	String	The value is fixed at <b>eni-neutron</b> .

Parameter	Mandator y	Type	Description
args	No	<a href="#">Table 7-11</a> object	Configuration parameters.

**Table 7-11** args parameters

Parameter	Mandator y	Type	Description
securityGrou ps	No	String	Security group ID. If no security group is planned, ensure that the security group is the same as that in <b>default-network</b> .  How to obtain: Log in to the VPC console. In the navigation pane, choose <b>Access Control &gt; Security Groups</b> . Click the target security group name and copy the ID on the <b>Summary</b> tab page.
subnets	Yes	Array of subnetID Objects	List of container subnet IDs. At least one subnet ID must be entered. The format is as follows: [{"subnetID":"27d95***"}, {"subnetID":"827bb***"}, {"subnetID":"bdd6b***"}]  Subnet ID not used by the cluster in the same VPC.  How to obtain: Log in to the VPC console. In the navigation pane, choose <b>Virtual Private Cloud &gt; Subnets</b> . Click the target subnet name and copy the <b>Subnet ID</b> on the <b>Summary</b> tab page.

**Step 3** Create a NetworkAttachmentDefinition.

```
kubectl create -f networkattachment-test.yaml
```

If information similar to the following is displayed, the NetworkAttachmentDefinition has been created.

```
networkattachmentdefinition.k8s.cni.cncf.io/example created
```

**Step 4** Create a Deployment workload and associate it with the newly created container network configuration.

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx
spec:
```

```

replicas: 3
selector:
  matchLabels:
    app: nginx
template:
  metadata:
    labels:
      app: nginx
      yangtse.io/network: "example" # Name of the custom container network configuration, which can
      be used to obtain all pods associated with the container network configuration by label
    annotations:
      yangtse.io/network: "example" # Name of the custom container network configuration
  spec:
    containers:
      - name: container-0
        image: nginx:alpine
        resources:
          limits:
            cpu: 100m
            memory: 200Mi
          requests:
            cpu: 100m
            memory: 200Mi
    imagePullSecrets:
      - name: default-secret

```

- **yangtse.io/network**: name of the specified custom container network configuration. Only a container network configuration that is not associated with any namespace can be specified. Add this parameter to the label so that you can use the label to obtain all pods associated with this container network configuration.

----End

## Verify That a Container Network Configuration Is Associated with a Namespace or Workload

Verify that a subnet and security group specified in the container network configuration have been associated with a workload. To check if a namespace is associated with a container network configuration, check whether a workload in the namespace is associated with a subnet and security group.

**Step 1** Verify that a subnet is associated.

1. Check the IP address of a pod for a workload.

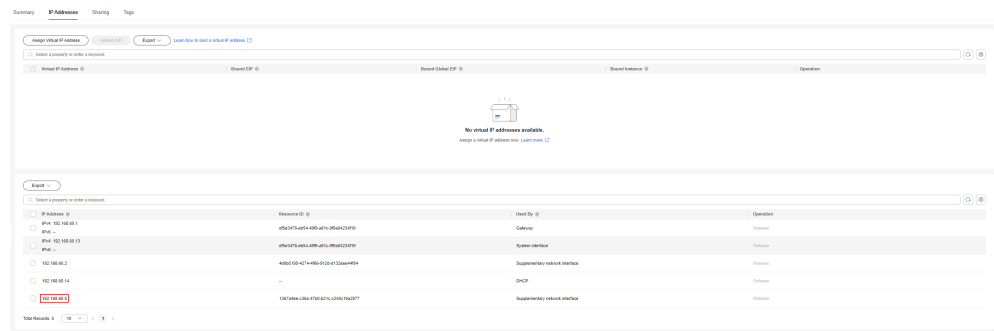
```
kubectl get pod -o wide
```

Information similar to the following is displayed:

NAME	READY	STATUS	RESTARTS	AGE	IP	NODE
NOMINATED NODE READINESS GATES						
nginx-85dbdb8c5d-ng5bb	1/1	Running	0	5d18h	192.168.60.5	ca50a5ae-e1ef-41c6-b3fc-6ebcd10a1e07
	<none>		<none>			

2. Log in to [Network Console](#). In the navigation pane, choose **Virtual Private Cloud > Subnets**. On the displayed page, click the subnet name.
3. Click **IP Addresses**. If the IP address of the pod is displayed, the subnet is associated.

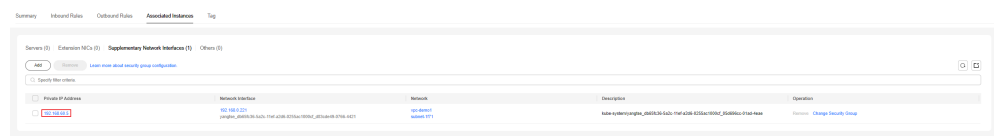
**Figure 7-14** Viewing the IP address associated with a pod



**Step 2** Verify that a security group is associated.

1. Log in to **Network Console**. In the navigation pane, choose **Access Control > Security Groups**. On the displayed page, click the security group name.
2. On the **Associated Instances** tab, click **Supplementary Network Interfaces**.
3. View the private IP address of the pod in the **Private IP Address** column. If the private IP address of the pod is displayed, the security group is associated.

**Figure 7-15** Viewing the IP address associated with a security group



----End

## Deleting a Container Network Configuration

You can delete the new container network configuration or view its YAML file.

**NOTE**

Before deleting a container network configuration, delete all pods using the configuration. Otherwise, the deletion will fail.

1. Run the following command to filter the pods that uses the configuration in the cluster (**example** is used as an example):  

```
kubectl get pod -A -o=jsonpath="{.items[?(@.metadata.annotations.cni\yangtse\io\network-status==['{\"name\": \"example\"}'])]['metadata.namespace', 'metadata.name']}"
```

The command output contains the pod name and namespace associated with the configuration.
2. Delete the owner of the pod. The owner may be a Deployment, StatefulSet, DaemonSet, or Job.

### 7.2.2.5 Configuring Shared Bandwidth for a Pod with IPv6 Dual-Stack ENIs

#### Application Scenarios

By default, pods with IPv6 dual-stack ENIs can access only the IPv6 private network. To access the public network, configure shared bandwidth for such pods.

## Notes and Constraints

- Only CCE Turbo clusters are supported and the following constraints must be met:
  - IPv6 dual stack has been enabled for the cluster.
  - The cluster version is v1.23.8-r0, v1.25.3-r0, or later.
- The number of IPv6 ENIs that can be added to a shared bandwidth is limited by the tenant quota. The default value is 20.
- HostNetwork Pods are not supported.
- All types of workloads are supported. When configuring IPv6 shared bandwidth for workloads with the replicas attribute, such as Deployment and StatefulSet, ensure that the number of replicas and the maximum number of pods during the upgrade are less than the remaining quota of IPv6 ENIs that can be added to the shared bandwidth.
- IPv6 dual-stack pod configured with shared bandwidth: When a pod is created, the CNI returns a success message only after the IPv6 dual-stack ENI is inserted into the shared bandwidth. When a pod is deleted, the IPv6 dual-stack ENI is removed from the shared bandwidth after the pod is completely deleted or the pod is in the deleting status for 30 seconds.
- If the IPv6 dual-stack ENI corresponding to the pod fails to be added to the shared bandwidth, an alarm event **FailedIPv6InsertBandwidth** is generated on the pod, for example, when the quota is exceeded or flow control is triggered. Rectify the fault based on the alarm event.
- On the **Shared Bandwidths** page of the EIP console, go to the target shared bandwidth details page, and click the **IPv6 Addresses** tab. The IPv6 dual-stack ENI whose **Associated Instance** is CCE is displayed. Do not remove the ENI directly on the page or using VPC APIs, otherwise, your services may be affected.

## Using the CCE Console

When creating a workload, you can set the IPv6 shared bandwidth on the **Advanced Settings > Network Configuration** area.

## Using kubectl

You can add annotations to a Deployment to specify the shared bandwidth to be added to the IPv6 dual-stack ENI of the pod. The following is an example:

```
...
spec:
  selector:
    matchLabels:
      app: demo
      version: v1
  template:
    metadata:
      annotations:
        yangtse.io/ipv6-bandwidth-id: "xxx"
```

- **yangtse.io/ipv6-bandwidth-id**: specifies the ID of the shared bandwidth. The IPv6 dual-stack ENIs corresponding to the pod will be added to the shared bandwidth. You can query the ID on the **Shared Bandwidths** page on the EIP console.



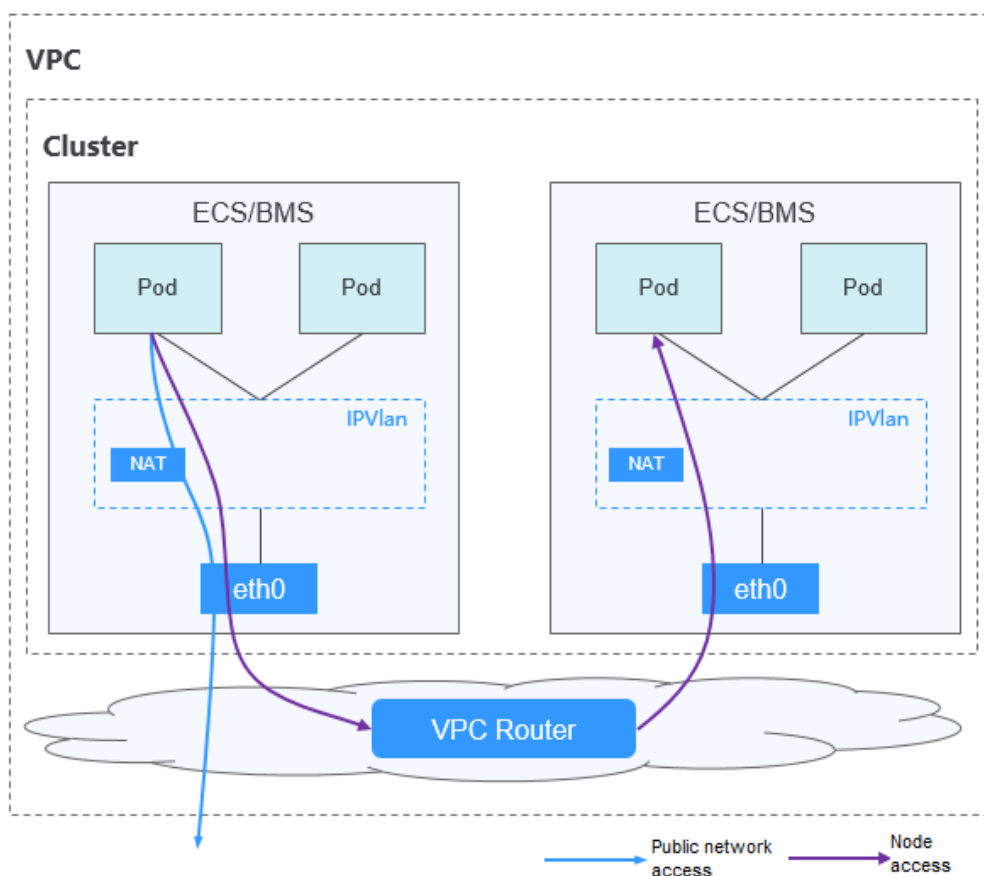
## 7.2.3 VPC Network Settings

### 7.2.3.1 VPC Network Model

#### Model Definition

The VPC network model seamlessly combines VPC routing with the underlying network, making it ideal for high-performance scenarios. However, the maximum number of nodes allowed in a cluster is determined by the VPC route quota. In the VPC network model, container CIDR blocks are separate from node CIDR blocks. To allocate IP addresses to containers running on a node in a cluster, each node in the cluster is allocated with a container CIDR block for a fixed number of IP addresses. The VPC network model outperforms the container tunnel network model in terms of performance because it does not have tunnel encapsulation overhead. When using the VPC network model in a cluster, the VPC routing table automatically configures the routes between container CIDR blocks and VPC CIDR blocks. This means that pods within the cluster can be accessed directly from cloud servers in the same VPC, even if they are outside the cluster.

Figure 7-16 VPC network model



In a cluster using the VPC network model, network communication paths are as follows:

- Inter-pod communication on the same node: Packets are directly forwarded through IPvlan.
- Inter-pod communication on different nodes: The traffic accesses the default gateway by following the route specified in the VPC routing table and then is forwarded to the target pod on another node using VPC routing.
- Pod communication with the Internet: When a container in a cluster needs to access the Internet, CCE uses NAT to translate the container's IP address into the node IP address so that the pod communicates externally using the node IP address.

#### NOTE

In a cluster using a VPC network, 10.0.0.0/8, 172.16.0.0/12, and 192.168.0.0/16 are regarded as private CIDR blocks of the cluster by default. If the VPC to which the cluster resides uses a secondary CIDR block, operations such as creating or resetting a node will also add the secondary CIDR block to the private CIDR blocks.

If a pod tries to access a private CIDR block, the source node will not perform NAT on the pod IP address. Instead, the upper-layer VPC 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.

If your VPC network cluster is of v1.23.14-r0, v1.25.9-r0, v1.27.6-r0, v1.28.4-r0, or later, CCE allows you to use **nonMasqueradeCIDRs** to configure private CIDR blocks for your cluster to tailor your cluster to different application needs. For details, see [Retaining the Original IP Address of a Pod](#).

## Advantages and Disadvantages

### Advantages

- High performance and simplified network fault locating are achieved by eliminating the need for tunnel encapsulation.
- A VPC routing table automatically configures routes between container CIDR blocks and VPC CIDR blocks. This enables resources in the VPC to directly communicate with containers in the cluster.

#### NOTE

Similarly, if the VPC is accessible to other VPCs or data centers and the VPC routing table includes routes to the container CIDR blocks, resources in other VPCs or data centers can directly communicate with containers in the cluster, provided there are no conflicts between the network CIDR blocks.

### Disadvantages

- The number of nodes is limited by the VPC route quota.
- Each node is assigned a CIDR block with a fixed size, which results in IP address wastage in the container CIDR block.
- Pods cannot directly use features like EIPs and security groups.

## Application Scenarios

- 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.

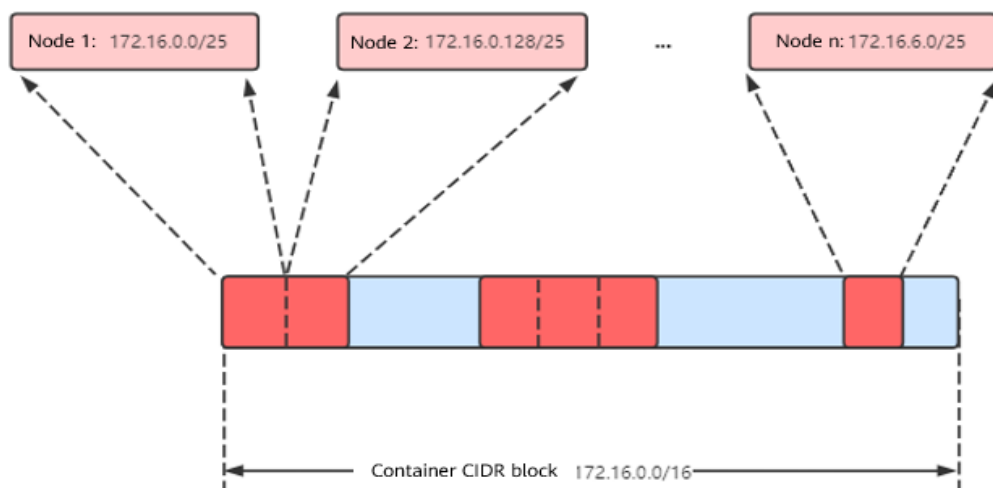
- Small- and medium-scale networks: Due to the limitation on VPC routing tables, it is recommended that the number of nodes in a cluster be less than or equal to 1000.

## Container IP Address Management

The VPC network model assigns container IP addresses based on the following guidelines:

- Container CIDR blocks are separate from node CIDR blocks.
- IP addresses are allocated by node. One CIDR block with a fixed size (configurable) is allocated to each node in a cluster from the container CIDR block.
- A container CIDR block assigns CIDR blocks to new nodes in a cyclical sequence.
- IP addresses from the CIDR blocks assigned to a node are allocated to pods scheduled to that node in a cyclical manner.

**Figure 7-17** IP address management of the VPC network



Maximum number of nodes that can be created in the cluster using the VPC network = Number of IP addresses in the container CIDR block/Number of IP addresses in the CIDR block allocated to the node by the container CIDR block

For example, if the container CIDR block is 172.16.0.0/16, the number of IP addresses is 65536. The mask of the container CIDR block allocated to a node is 25. That is, the number of container IP addresses on each node is 128. Therefore, a maximum of 512 (65536/128) nodes can be created. The number of nodes that can be added to a cluster is also determined by the available IP addresses in the node subnet and the scale of the cluster. For details, see [Recommendation for CIDR Block Planning](#).

## Recommendation for CIDR Block Planning

As explained in [Cluster Network Structure](#), network addresses in a cluster are divided into the cluster network, container network, and service network. When planning network addresses, consider the following factors:

- **The three CIDR blocks cannot overlap.** Otherwise, a conflict occurs.
- Ensure that **each CIDR block has sufficient IP addresses.**
  - The IP addresses in the cluster CIDR block must match the cluster scale. Otherwise, nodes cannot be created due to insufficient IP addresses.
  - The IP addresses in the container CIDR block must match the service scale. Otherwise, pods cannot be created due to insufficient IP addresses. The number of pods that can be created on each node also depends on other parameter settings.

Assume that a cluster contains 200 nodes and the network model is VPC network.

In this case, the number of available IP addresses in the selected subnet must be greater than 200. Otherwise, nodes cannot be created due to insufficient IP addresses.

The container CIDR block is 172.16.0.0/16, and the number of available IP addresses is 65536. As described in [Container IP Address Management](#), the VPC network is allocated a CIDR block with a fixed size (using the mask to determine the maximum number of container IP addresses allocated to each node). For example, if the upper limit is 128, the cluster supports a maximum of 512 (65536/128) nodes.

## Example of VPC Network Access

In this example, a cluster using the VPC network model is created, and the cluster contains one node.

On the VPC console, locate the VPC to which the cluster belongs and check the VPC routing table.

You can find that CCE has created a custom route in the routing table. This route has a destination address corresponding to the container CIDR block assigned to the node, and the next hop is directed towards the target node. In the example, the container CIDR block for the cluster is 172.16.0.0/16, with 128 container IP addresses assigned to each node. Therefore, the node's container CIDR block is 172.16.0.0/25, providing a total of 128 container IP addresses.

When a container IP address is accessed, the VPC route will forward the traffic to the next-hop node that corresponds to the destination address. The following is an example:

**Step 1** Use `kubectl` to access the cluster. For details, see [Connecting to a Cluster Using kubectl](#).

**Step 2** Create a Deployment in the cluster.

Create the `deployment.yaml` file. The following shows an example:

```
kind: Deployment
apiVersion: apps/v1
metadata:
  name: example
  namespace: default
spec:
  replicas: 4
  selector:
    matchLabels:
      app: example
  template:
```

```
metadata:
  labels:
    app: example
spec:
  containers:
  - name: container-0
    image: 'nginx:perl'
  imagePullSecrets:
  - name: default-secret
```

Create the workload.

```
kubectl apply -f deployment.yaml
```

### Step 3 Check the running pods.

```
kubectl get pod -owide
```

Command output:

NAME	READY	STATUS	RESTARTS	AGE	IP	NODE	NOMINATED	NODE
example-86b9779494-l8qrw	1/1	Running	0	14s	172.16.0.6	192.168.0.99	<none>	<none>
example-86b9779494-svs8t	1/1	Running	0	14s	172.16.0.7	192.168.0.99	<none>	<none>
example-86b9779494-x8kl5	1/1	Running	0	14s	172.16.0.5	192.168.0.99	<none>	<none>
example-86b9779494-zt627	1/1	Running	0	14s	172.16.0.8	192.168.0.99	<none>	<none>

### Step 4 Use a cloud server in the same VPC to directly access a pod's IP address from outside the cluster. You can also access a pod using its IP address within the pod or from a node in the cluster. In the following example, access a pod's IP address within the pod. *example-86b9779494-l8qrw* is the pod name, and *172.16.0.7* is the pod IP address.

```
kubectl exec -it example-86b9779494-l8qrw -- curl 172.16.0.7
```

If the following information is displayed, the workload can be properly accessed:

```
<!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>
<p>If you see this page, the nginx web server is successfully installed and
working. Further configuration is required.</p>

<p>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>.</p>

<p><em>Thank you for using nginx.</em></p>
</body>
</html>
```

----End

### 7.2.3.2 Adding a Container CIDR Block for a Cluster

#### Scenario

If the container CIDR block configured during CCE cluster creation cannot meet service expansion requirements, you can add a container CIDR block for the cluster.

#### Notes and Constraints

- This function is available only for clusters of v1.19 or later using a VPC network.
- An added container CIDR block cannot be deleted.

#### Adding a Container CIDR Block for a CCE Standard Cluster

**Step 1** Log in to the CCE console and click the cluster name to access the cluster console.

**Step 2** On the **Overview** page, locate the **Networking Configuration** area and click **Add**.

**Figure 7-18** Adding container CIDR block

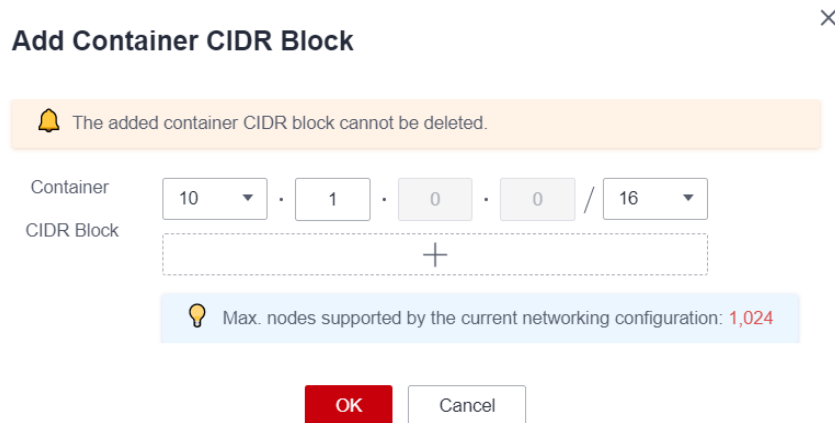
Networking Configuration		Go to Network Configuration
Network Model	VPC network	
VPC	<a href="#">vpc-autotest</a>	
Subnet	<a href="#">subnet-autotest1</a>	
Container CIDR Block	172.16.0.0/16	
	<a href="#">Add</a>	
IPv4 Service CIDR Block	10.247.0.0/16	
Forwarding	iptables	
Default Node Security Group	<a href="#">[redacted]-cce-node-bclbz</a>	<a href="#">Edit</a>

**Step 3** Configure the container CIDR block to be added. You can click **+** to add multiple container CIDR blocks at a time.

#### NOTE

New container CIDR blocks cannot conflict with service CIDR blocks, VPC CIDR blocks, and existing container CIDR blocks.

**Figure 7-19** Configuring the container CIDR block



**Step 4** Click **OK**.

----End

## 7.2.4 Tunnel Network Settings

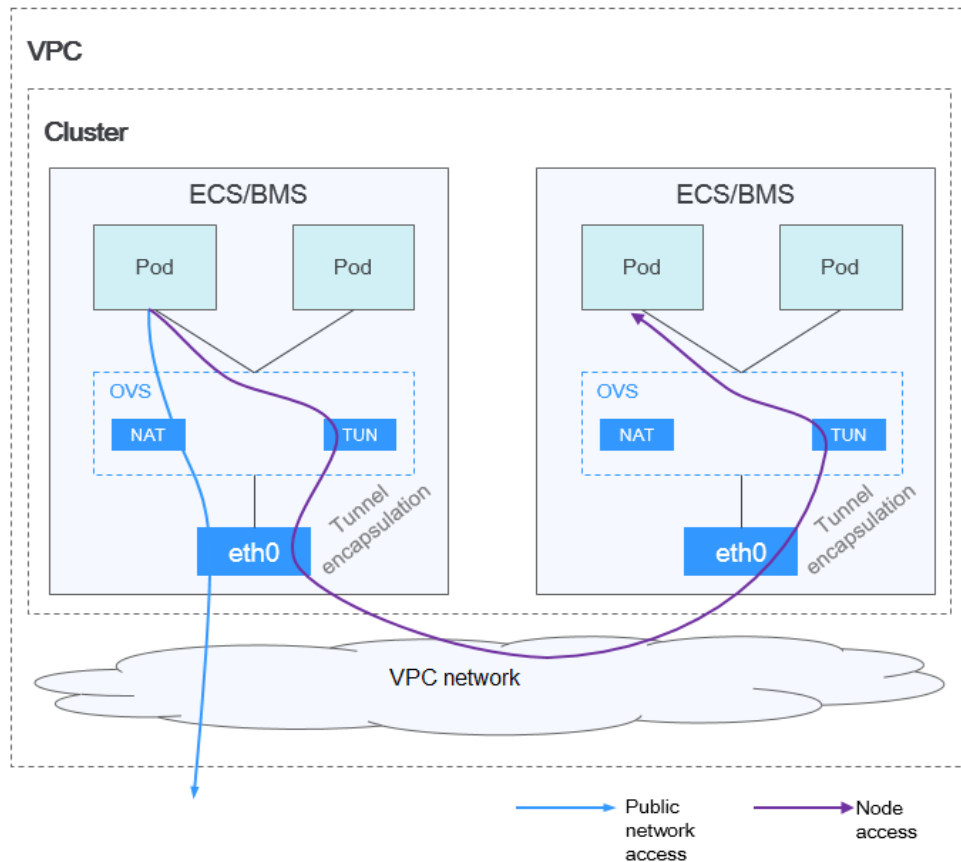
### 7.2.4.1 Tunnel Network Model

#### Model Definition

A container tunnel network creates a separate network plane for containers by using tunnel encapsulation on the host network plane. The container tunnel network of a CCE cluster uses VXLAN for tunnel encapsulation and Open vSwitch as the virtual switch backend. VXLAN is a protocol that encapsulates Ethernet packets into UDP packets to transmit them through tunnels. Open vSwitch is an open-source virtual switch software that provides functions such as network isolation and data forwarding.

While there may be some performance costs, packet encapsulation and tunnel transmission allow for greater interoperability and compatibility with advanced features, such as network policy-based isolation, in most common scenarios.

**Figure 7-20** Container tunnel network



In a cluster using the container tunnel model, the communication paths between pods on the same node and between pods on different nodes are different.

- Inter-pod communication on the same node: Packets are directly forwarded via the OVS bridge on the node.
- Inter-pod communication on different nodes: Packets are encapsulated in the OVS bridge and then forwarded to the target pod on the peer node through the host NIC.

## Advantages and Disadvantages

### Advantages

- The container network is decoupled from the node network and is not limited by the VPC quotas and response speed (such as the number of VPC routes, number of ENIs, and creation speed).
- Network isolation is supported. For details, see [Configuring Network Policies to Restrict Pod Access](#).
- Bandwidth limits are supported.
- Large-scale networking with a maximum of 2000 nodes is supported.

### Disadvantages

- High encapsulation overhead results in poor performance and makes it difficult to locate network faults.



- Pods cannot directly use features like EIPs and security groups.
- Container IP addresses cannot be directly accessed by external networks.

## Application Scenarios

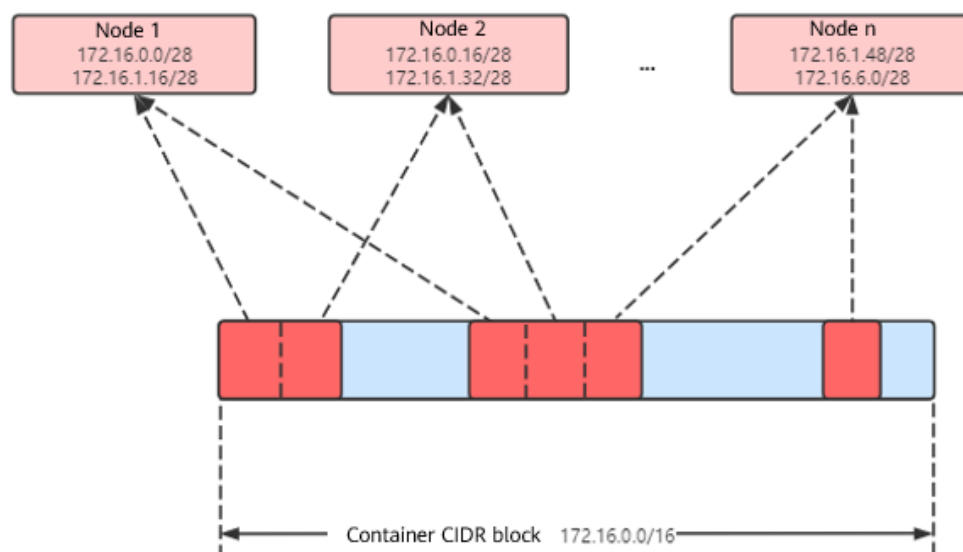
- Low requirements on performance: As the container tunnel network requires additional VXLAN tunnel encapsulation, it has about 5% to 15% of performance loss when compared with the other two container network models. Therefore, the container tunnel network applies to the scenarios that do not have high 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 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.

## Container IP Address Management

The container tunnel network allocates container IP addresses according to the following rules:

- Container CIDR blocks are separate from node CIDR blocks.
- IP addresses are allocated by node. One or more CIDR blocks with a fixed size (16 by default) are allocated to each node in a cluster from the container CIDR block.
- When the IP addresses on a node are used up, you can apply for a new CIDR block.
- A container CIDR block assigns CIDR blocks to new nodes or existing nodes in a cyclical sequence.
- IP addresses from one or more CIDR blocks assigned to a node are allocated to pods scheduled to that node in a cyclical manner.

Figure 7-21 IP address allocation of the container tunnel network



Maximum number of nodes that can be created in the cluster using the container tunnel network = Number of IP addresses in the container CIDR block/Size of the IP CIDR block allocated to the node by the container CIDR block at a time (16 by default)

For example, if the container CIDR block is 172.16.0.0/16, the number of IP addresses is 65536. The mask of the container CIDR block allocated to a node is 28. That is, a total of 16 container IP addresses are allocated each time. Therefore, a maximum of 4096 (65536/16) nodes can be created. This is an extreme case. If 4096 nodes are created, a maximum of 16 pods can be created for each node because only a CIDR block with 16 IP addresses is allocated to each node. The number of nodes that can be added to a cluster is also determined by the available IP addresses in the node subnet and the scale of the cluster.

## Recommendation for CIDR Block Planning

As explained in [Cluster Network Structure](#), network addresses in a cluster are divided into the cluster network, container network, and service network. When planning network addresses, consider the following factors:

- **The three CIDR blocks cannot overlap.** Otherwise, a conflict occurs.
- Ensure that **each CIDR block has sufficient IP addresses.**
  - The IP addresses in the cluster CIDR block must match the cluster scale. Otherwise, nodes cannot be created due to insufficient IP addresses.
  - The IP addresses in the container CIDR block must match the service scale. Otherwise, pods cannot be created due to insufficient IP addresses. The number of pods that can be created on each node also depends on other parameter settings.

## Example of Container Tunnel Network Access

The following is an example of creating a workload in a cluster using the container tunnel network model:

**Step 1** Use `kubectl` to access the cluster. For details, see [Connecting to a Cluster Using kubectl](#).

**Step 2** Create a Deployment in the cluster.

Create the `deployment.yaml` file. The following shows an example:

```
kind: Deployment
apiVersion: apps/v1
metadata:
  name: example
  namespace: default
spec:
  replicas: 4
  selector:
    matchLabels:
      app: example
  template:
    metadata:
      labels:
        app: example
    spec:
      containers:
        - name: container-0
          image: 'nginx:perl'
```

```
resources:
  limits:
    cpu: 250m
    memory: 512Mi
  requests:
    cpu: 250m
    memory: 512Mi
  imagePullSecrets:
  - name: default-secret
```

Create the workload.

```
kubectl apply -f deployment.yaml
```

### Step 3 Check the running pods.

```
kubectl get pod -owide
```

Command output:

NAME	READY	STATUS	RESTARTS	AGE	IP	NODE	NOMINATED NODE
READINESS GATES							
example-5bdc5699b7-5rvq4	1/1	Running	0	3m28s	10.0.0.20	192.168.0.42	<none>
example-5bdc5699b7-984j9	1/1	Running	0	3m28s	10.0.0.21	192.168.0.42	<none>
example-5bdc5699b7-lfxkm	1/1	Running	0	3m28s	10.0.0.22	192.168.0.42	<none>
example-5bdc5699b7-wjcmg	1/1	Running	0	3m28s	10.0.0.52	192.168.0.64	<none>

### Step 4 Use a cloud server in the same VPC to directly access a pod's IP address from outside the cluster. The access failed.

You can access a pod using its IP address within the pod or from a node in the cluster. In the following example, access a pod's IP address within the pod. *example-5bdc5699b7-5rvq4* is the pod name, and *10.0.0.21* is the pod IP address.

```
kubectl exec -it example-5bdc5699b7-5rvq4 -- curl 10.0.0.21
```

If the following information is displayed, the workload can be properly accessed:

```
<!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>
<p>If you see this page, the nginx web server is successfully installed and
working. Further configuration is required.</p>

<p>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>.</p>

<p><em>Thank you for using nginx.</em></p>
</body>
</html>
```

----End

## 7.2.5 Pod Network Settings

### 7.2.5.1 Configuring hostNetwork for Pods

#### Scenario

Kubernetes allows pods to directly use the host/node network. When a pod is configured with **hostNetwork: true**, applications running in the pod can directly view the network interface of the host where the pod is located.

#### Configuration

Add **hostNetwork: true** to the pod definition.

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx
spec:
  replicas: 1
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      hostNetwork: true
      containers:
        - image: nginx:alpine
          name: nginx
          imagePullSecrets:
            - name: default-secret
```

The configuration succeeds if the pod IP is the same as the node IP.

```
$ kubectl get pod -owide
NAME                                READY STATUS RESTARTS AGE IP      NODE      NOMINATED NODE
READINESS GATES
nginx-6fdf99c8b-6wwft 1/1    Running 0       3m41s 10.1.0.55 10.1.0.55 <none>      <none>
```

#### Precautions

When a pod uses a host network, it uses a host port and its IP address is the same as the host's IP address. Before using a host network, verify that the pod port does not conflict with any service port on the host. Use a host network only if a workload pod must access a specific host port.

When using the host network, you access a pod on a node through a node port. Therefore, **allow access from the security group port of the node**. Otherwise, the access fails.

In addition, using the host network requires you to reserve host ports for the pods. When using a Deployment to deploy pods of the hostNetwork type, ensure that **the number of pods does not exceed the number of nodes**. Otherwise, multiple pods will be scheduled onto the node, and they will fail to start due to port conflicts. For example, in the preceding example nginx YAML, if two pods (setting **replicas** to 2) are deployed in a cluster with only one node, one pod cannot be

created. The pod logs will show that the Nginx cannot be started because the port is occupied.

**CAUTION**

Do not schedule multiple pods that use the host network on the same node. Otherwise, when a ClusterIP Service is created to access a pod, the cluster IP address cannot be accessed.

```
$ kubectl get deploy
NAME READY UP-TO-DATE AVAILABLE AGE
nginx 1/2 2 1 67m
$ kubectl get pod
NAME READY STATUS RESTARTS AGE
nginx-6fdf99c8b-6wwft 1/1 Running 0 67m
nginx-6fdf99c8b-rglm7 0/1 CrashLoopBackOff 13 44m
$ kubectl logs nginx-6fdf99c8b-rglm7
/docker-entrypoint.sh: /docker-entrypoint.d/ is not empty, will attempt to perform configuration
/docker-entrypoint.sh: Looking for shell scripts in /docker-entrypoint.d/
/docker-entrypoint.sh: Launching /docker-entrypoint.d/10-listen-on-ipv6-by-default.sh
10-listen-on-ipv6-by-default.sh: info: Getting the checksum of /etc/nginx/conf.d/default.conf
10-listen-on-ipv6-by-default.sh: info: Enabled listen on IPv6 in /etc/nginx/conf.d/default.conf
/docker-entrypoint.sh: Launching /docker-entrypoint.d/20-envsubst-on-templates.sh
/docker-entrypoint.sh: Launching /docker-entrypoint.d/30-tune-worker-processes.sh
/docker-entrypoint.sh: Configuration complete; ready for start up
2022/05/11 07:18:11 [emerg] 1#1: bind() to 0.0.0.0:80 failed (98: Address in use)
nginx: [emerg] bind() to 0.0.0.0:80 failed (98: Address in use)
2022/05/11 07:18:11 [emerg] 1#1: bind() to [::]:80 failed (98: Address in use)
nginx: [emerg] bind() to [::]:80 failed (98: Address in use)
2022/05/11 07:18:11 [emerg] 1#1: bind() to 0.0.0.0:80 failed (98: Address in use)
nginx: [emerg] bind() to 0.0.0.0:80 failed (98: Address in use)
2022/05/11 07:18:11 [emerg] 1#1: bind() to [::]:80 failed (98: Address in use)
nginx: [emerg] bind() to [::]:80 failed (98: Address in use)
2022/05/11 07:18:11 [emerg] 1#1: bind() to 0.0.0.0:80 failed (98: Address in use)
nginx: [emerg] bind() to 0.0.0.0:80 failed (98: Address in use)
2022/05/11 07:18:11 [emerg] 1#1: bind() to [::]:80 failed (98: Address in use)
nginx: [emerg] bind() to [::]:80 failed (98: Address in use)
2022/05/11 07:18:11 [emerg] 1#1: bind() to 0.0.0.0:80 failed (98: Address in use)
nginx: [emerg] bind() to 0.0.0.0:80 failed (98: Address in use)
2022/05/11 07:18:11 [emerg] 1#1: bind() to [::]:80 failed (98: Address in use)
nginx: [emerg] bind() to [::]:80 failed (98: Address in use)
2022/05/11 07:18:11 [emerg] 1#1: still could not bind()
nginx: [emerg] still could not bind()
```

## 7.2.5.2 Configuring QoS for a Pod

### Scenario

Bandwidth preemption occurs between different containers deployed on the same node, which may cause service jitter. You can configure QoS rate limiting for inter-pod access to prevent this problem.

### Notes and Constraints

The following shows constraints on setting the rate limiting for inter-pod access:

Constraint	Tunnel Network	VPC Network	Cloud Native 2.0 Network
Supported versions	All versions	Clusters of v1.19.10 and later	Clusters of v1.19.10 and later
Supported runtime types	Only regular containers (runC as the container runtime) are supported. Secure containers (Kata Containers as the container runtime) are not supported.		
Supported pod types	Only non-HostNetwork pods		
Supported scenarios	Inter-pod access, pods accessing nodes, and pods accessing services		
Constraints	None	None	<ul style="list-style-type: none"> <li>Pods access external cloud service CIDR blocks 100.64.0.0/10 and 214.0.0.0/8.</li> <li>Traffic rate limiting of health check</li> </ul>
Value range of rate limit	Only the rate limit in the unit of Mbit/s or Gbit/s is supported, for example, 100 Mbit/s and 1 Gbit/s. The minimum value is 1 Mbit/s and the maximum value is 4.29 Gbit/s.		

## Using kubectl

You can add annotations to a workload to specify its egress and ingress bandwidth.

```

apiVersion: apps/v1
kind: Deployment
metadata:
  name: test
  namespace: default
  labels:
    app: test
spec:
  replicas: 2
  selector:
    matchLabels:
      app: test
  template:
    metadata:
      labels:
        app: test
      annotations:
        kubernetes.io/ingress-bandwidth: 100M
        kubernetes.io/egress-bandwidth: 100M
    spec:
      containers:
        - name: container-1
  
```

```
image: nginx:alpine
imagePullPolicy: IfNotPresent
imagePullSecrets:
- name: default-secret
```

- **kubernetes.io/ingress-bandwidth**: ingress bandwidth of the pod
- **kubernetes.io/egress-bandwidth**: egress bandwidth of the pod

If these two parameters are not specified, the bandwidth is not limited.

#### NOTE

After modifying the ingress or egress bandwidth limit of a pod, restart the container for the modification to take effect. After annotations are modified in a pod not managed by workloads, the container will not be restarted, so the bandwidth limits do not take effect. You can create a pod again or manually restart the container.

### 7.2.5.3 Configuring Network Policies to Restrict Pod Access

Network policies are designed by Kubernetes to restrict pod access. It is equivalent to a firewall at the application layer to enhance network security. The capabilities supported by network policies depend on the capabilities of the network add-ons of the cluster.

By default, if a namespace does not have any policy, pods in the namespace accept traffic from any source and send traffic to any destination.

Network policies are classified into the following types:

- **namespaceSelector**: selects particular namespaces for which all pods should be allowed as ingress sources or egress destinations.
- **podSelector**: selects particular pods in the same namespace as the network policy which should be allowed as ingress sources or egress destinations.
- **IPBlock**: selects particular IP blocks to allow as ingress sources or egress destinations.

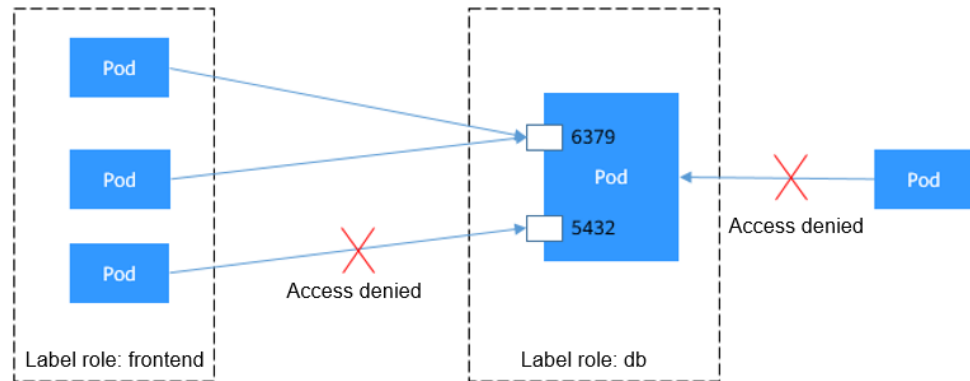
#### Notes and Constraints

- Only clusters that use the tunnel network model support network policies. Network policies are classified into the following types:
  - Ingress: All versions support this type.
  - Egress: This rule type cannot be set currently.
- Network isolation is not supported for IPv6 addresses.

#### Using Ingress Rules Through YAML

- **Scenario 1: Use a network policy to limit access to a pod to only pods with specific labels.**

**Figure 7-22** podSelector



The pod labeled with **role=db** only permits access to its port 6379 from pods labeled with **role=frontend**. To do so, perform the following operations:

- a. Create the **access-demo1.yaml** file.

```
vim access-demo1.yaml
```

File content:

```
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: access-demo1
  namespace: default
spec:
  podSelector:           # The rule takes effect for pods with the role=db label.
    matchLabels:
      role: db
  ingress:               # This is an ingress rule.
  - from:
    - podSelector:       # Only allow the access of the pods labeled with role=frontend.
      matchLabels:
        role: frontend
    ports:               # Only TCP can be used to access port 6379.
      - protocol: TCP
        port: 6379
```

- b. Run the following command to create the network policy based on the **access-demo1.yaml** file:

```
kubectl apply -f access-demo1.yaml
```

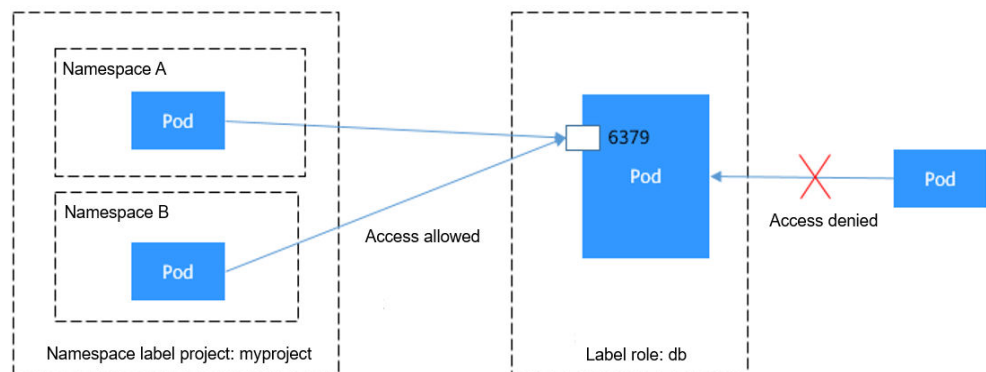
Expected output:

```
networkpolicy.networking.k8s.io/access-demo1 created
```

- **Scenario 2: Use a network policy to limit access to a pod to only pods in a specific namespace.**



**Figure 7-23 namespaceSelector**



The pod labeled with **role=db** only permits access to its port 6379 from pods in the namespace labeled with **project=myproject**. To do so, perform the following operations:

- a. Create the **access-demo2.yaml** file.

```
vim access-demo2.yaml
```

File content:

```
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: access-demo2
spec:
  podSelector:          # The rule takes effect for pods with the role=db label.
    matchLabels:
      role: db
  ingress:              # This is an ingress rule.
  - from:
    - namespaceSelector: # Only allow the access of the pods in the namespace labeled
      with project=myproject.
      matchLabels:
        project: myproject
    ports:              # Only TCP can be used to access port 6379.
    - protocol: TCP
      port: 6379
```

- b. Run the following command to create the network policy based on the **access-demo2.yaml** file:

```
kubectl apply -f access-demo2.yaml
```

Expected output:

```
networkpolicy.networking.k8s.io/access-demo2 created
```

## Creating a Network Policy on the Console

**Step 1** Log in to the CCE console and click the cluster name to access the cluster console.

**Step 2** Choose **Policies** in the navigation pane, click the **Network Policies** tab, and click **Create Network Policy** in the upper right corner.

- **Policy Name:** Specify a network policy name.
- **Namespace:** Select a namespace in which the network policy is applied.
- **Selector:** Enter a label, select the pod to be associated, and click **Add**. You can also click **Reference Workload Label** to use the label of an existing workload.
- **Inbound Rule:** Click **+** to add an inbound rule. For details about parameter settings, see [Table 7-12](#).

**Table 7-12** Adding an inbound rule

Parameter	Description
Protocol & Port	Select the protocol type and port. Currently, TCP and UDP are supported.
Source Namespace	Select a namespace whose objects can be accessed. If this parameter is not specified, the object belongs to the same namespace as the current policy.
Source Pod Label	Allow accessing the pods with this label. If this parameter is not specified, all pods in the namespace can be accessed.

**Step 3** Click **OK**.

----End

## 7.3 Service

### 7.3.1 Overview

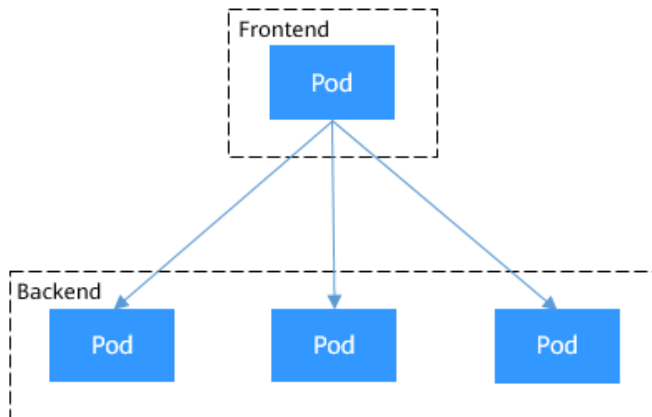
#### Direct Access to a Pod

After a pod is created, the following problems may occur if you directly access the pod:

- The pod can be deleted and recreated at any time by a controller such as a Deployment, and the result of accessing the pod becomes unpredictable.
- The IP address of the pod is allocated only after the pod is started. Before the pod is started, the IP address of the pod is unknown.
- An application is usually composed of multiple pods that run the same image. Accessing pods one by one is not efficient.

For example, an application uses Deployments to create the frontend and backend. The frontend calls the backend for computing, as shown in [Figure 7-24](#). Three pods are running in the backend, which are independent and replaceable. When a backend pod is re-created, the new pod is assigned with a new IP address, of which the frontend pod is unaware.

Figure 7-24 Inter-pod access

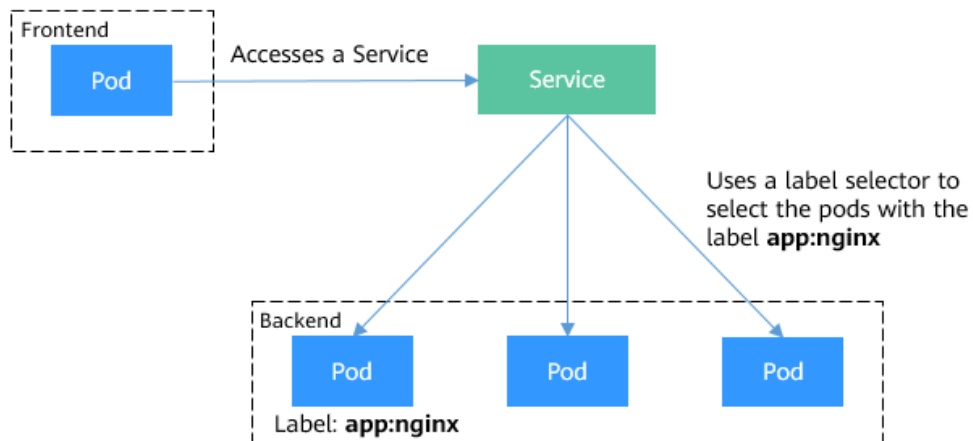


## Using Services for Pod Access

Kubernetes Services are used to solve the preceding pod access problems. A Service has a fixed IP address. (When a CCE cluster is created, a Service CIDR block is set, which is used to allocate IP addresses to Services.) A Service forwards requests accessing the Service to pods based on labels, and at the same time, perform load balancing for these pods.

In the preceding example, a Service is added for the frontend pod to access the backend pods. In this way, the frontend pod does not need to be aware of the changes on backend pods, as shown in [Figure 7-25](#).

Figure 7-25 Accessing pods through a Service



## Service Types

Kubernetes allows you to specify a Service of a required type. The values and actions of different types of Services are as follows:

- **ClusterIP**

ClusterIP Services allow workloads in the same cluster to use their cluster-internal domain names to access each other.

- **NodePort**  
A Service is exposed on each node's IP address at a static port (NodePort). A ClusterIP Service, to which the NodePort Service will route, is automatically created. By requesting <NodeIP>:<NodePort>, you can access a NodePort Service from outside the cluster.
- **LoadBalancer**  
LoadBalancer Services can access workloads from the public network through a load balancer, which is more reliable than EIP-based access. LoadBalancer Services are recommended for accessing workloads from outside the cluster.

## externalTrafficPolicy (Service Affinity)

For a NodePort and LoadBalancer Service, requests are first sent to the node port, then the Service, and finally the pod backing the Service. The backing pod may be not located in the node receiving the requests. By default, the backend workload can be accessed from any node IP address and service port. If the pod is not on the node that receives the request, the request will be redirected to the node where the pod is located, which may cause performance loss.

The **externalTrafficPolicy** parameter in a Service is used to determine whether the external traffic can be routed to the local nodes or cluster-wide endpoints. The following is an example:

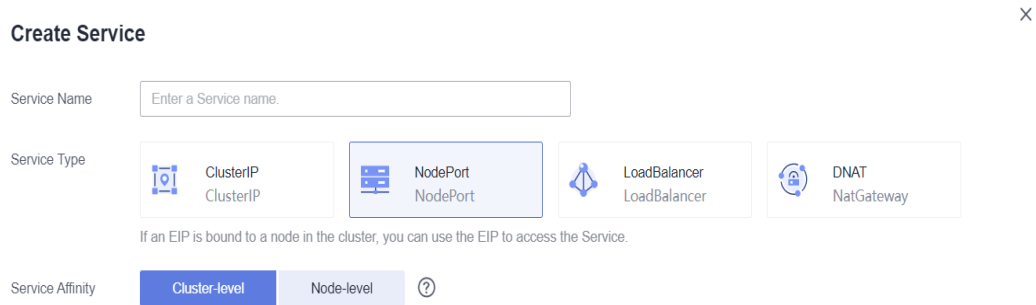
```
apiVersion: v1
kind: Service
metadata:
  name: nginx-nodeport
spec:
  externalTrafficPolicy: Local
  ports:
  - name: service
    nodePort: 30000
    port: 80
    protocol: TCP
    targetPort: 80
  selector:
    app: nginx
    type: NodePort
```

If the value of **externalTrafficPolicy** is **Local**, requests sent from *Node IP address:Service port* will be forwarded only to the pod on the local node. If the node does not have a pod, the requests are suspended.

If the value of **externalTrafficPolicy** is **Cluster**, requests are forwarded within the cluster and the backend workload can be accessed from any node IP address and service port.

If **externalTrafficPolicy** is not set, the default value **Cluster** will be used.

When creating a NodePort on the CCE console, you can configure this parameter using the **Service Affinity** option.



The following table compares the two options of **externalTrafficPolicy**.

**Table 7-13** Comparison of the two types of service affinity

Dimension	externalTrafficPolicy (Service Affinity)	
	Cluster-level (Cluster)	Node-level (Local)
Application scenario	This mode applies to scenarios where high performance is not required and the source IP address of the client does not need to be retained. This mode brings more balanced load to each node in the cluster.	This mode applies to scenarios where high performance is required and the source IP address of the client needs to be retained. However, traffic is forwarded only to the node where the container resides, and source IP address translation is not performed.
Access mode	The IP addresses and access ports of all nodes in a cluster can access the workload associated with the Service.	Only the IP address and access port of the node where the workload is located can access the workload associated with the Service.
Obtaining client source IP address	The source IP address of the client cannot be obtained.	The source IP address of the client can be obtained.
Access performance	Service access will cause performance loss due to route redirection, and the next hop for a data packet may be another node.	Service access will not cause performance loss due to route redirection.
Load balancing	Traffic propagation has good overall load balancing.	There is a potential risk of unbalanced traffic propagation.

Dimension	externalTrafficPolicy (Service Affinity)	
	Cluster-level (Cluster)	Node-level (Local)
Other special case	None	In different container network models and service forwarding modes, accessing Services from within the cluster may fail. For details, see <a href="#">Why a Service Fail to Be Accessed from Within the Cluster</a> .

## Why a Service Fail to Be Accessed from Within the Cluster

If the service affinity of a Service is set to the node level, that is, the value of **externalTrafficPolicy** is **Local**, the Service may fail to be accessed from within the cluster (specifically, nodes or containers). Information similar to the following is displayed:

```
upstream connect error or disconnect/reset before headers. reset reason: connection failure
Or
curl: (7) Failed to connect to 192.168.10.36 port 900: Connection refused
```

It is common that a load balancer in a cluster cannot be accessed. The reason is as follows: When Kubernetes creates a Service, kube-proxy adds the access address of the load balancer as an external IP address (External-IP, as shown in the following command output) to iptables or IPVS. If a client inside the cluster initiates a request to access the load balancer, the address is considered as the external IP address of the Service, and the request is directly forwarded by kube-proxy without passing through the load balancer outside the cluster.

```
# kubectl get svc nginx
NAME      TYPE          CLUSTER-IP   EXTERNAL-IP   PORT(S)          AGE
nginx    LoadBalancer 10.247.76.156 123.**.**.**,192.168.0.133 80:32146/TCP    37s
```

When the value of **externalTrafficPolicy** is **Local**, the access failures in different container network models and service forwarding modes are as follows:

### NOTE

- For a multi-pod workload, ensure that all pods are accessible. Otherwise, there is a possibility that the access to the workload fails.
- The table lists only the scenarios where the access may fail. Other scenarios that are not listed in the table indicate that the access is normal.

Service Type Released on the Server	Access Type	Request Initiation Location on the Client	Tunnel Network Cluster (IPVS)	VPC Network Cluster (IPVS)	Tunnel Network Cluster (iptables)	VPC Network Cluster (iptables)
NodePort Service	Public/Private network	Same node as the service pod	<p>Access the IP address and NodePort on the node where the server is located: The access is successful.</p> <p>Access the IP address and NodePort on a node other than the node where the server is located: The access failed.</p>	<p>Access the IP address and NodePort on the node where the server is located: The access is successful.</p> <p>Access the IP address and NodePort on a node other than the node where the server is located: The access failed.</p>	<p>Access the IP address and NodePort on the node where the server is located: The access is successful.</p> <p>Access the IP address and NodePort on a node other than the node where the server is located: The access failed.</p>	<p>Access the IP address and NodePort on the node where the server is located: The access is successful.</p> <p>Access the IP address and NodePort on a node other than the node where the server is located: The access failed.</p>

Service Type Released on the Server	Access Type	Request Initiation Location on the Client	Tunnel Network Cluster (IPVS)	VPC Network Cluster (IPVS)	Tunnel Network Cluster (iptables)	VPC Network Cluster (iptables)
		Different nodes from the service pod	<p>Access the IP address and NodePort on the node where the server is located: The access is successful.</p> <p>Access the IP address and NodePort on a node other than the node where the server is located: The access failed.</p>	<p>Access the IP address and NodePort on the node where the server is located: The access is successful.</p> <p>Access the IP address and NodePort on a node other than the node where the server is located: The access failed.</p>	The access is successful.	The access is successful.



Service Type Released on the Server	Access Type	Request Initiation Location on the Client	Tunnel Network Cluster (IPVS)	VPC Network Cluster (IPVS)	Tunnel Network Cluster (iptables)	VPC Network Cluster (iptables)
		Other containers on the same node as the service pod	<p>Access the IP address and NodePort on the node where the server is located: The access is successful.</p> <p>Access the IP address and NodePort on a node other than the node where the server is located: The access failed.</p>	The access failed.	<p>Access the IP address and NodePort on the node where the server is located: The access is successful.</p> <p>Access the IP address and NodePort on a node other than the node where the server is located: The access failed.</p>	The access failed.

Service Type Released on the Server	Access Type	Request Initiation Location on the Client	Tunnel Network Cluster (IPVS)	VPC Network Cluster (IPVS)	Tunnel Network Cluster (iptables)	VPC Network Cluster (iptables)
		Other containers on different nodes from the service pod	<p>Access the IP address and NodePort on the node where the server is located: The access is successful.</p> <p>Access the IP address and NodePort on a node other than the node where the server is located: The access failed.</p>	<p>Access the IP address and NodePort on the node where the server is located: The access is successful.</p> <p>Access the IP address and NodePort on a node other than the node where the server is located: The access failed.</p>	<p>Access the IP address and NodePort on the node where the server is located: The access is successful.</p> <p>Access the IP address and NodePort on a node other than the node where the server is located: The access failed.</p>	<p>Access the IP address and NodePort on the node where the server is located: The access is successful.</p> <p>Access the IP address and NodePort on a node other than the node where the server is located: The access failed.</p>
LoadBalancer Service using a dedicated load balancer	Private network	Same node as the service pod	The access failed.	The access failed.	The access failed.	The access failed.

Service Type Released on the Server	Access Type	Request Initiation Location on the Client	Tunnel Network Cluster (IPVS)	VPC Network Cluster (IPVS)	Tunnel Network Cluster (iptables)	VPC Network Cluster (iptables)
		Other containers on the same node as the service pod	The access failed.	The access failed.	The access failed.	The access failed.
DNAT gateway Service	Public network	Same node as the service pod	The access failed.	The access failed.	The access failed.	The access failed.
		Different nodes from the service pod	The access failed.	The access failed.	The access failed.	The access failed.
		Other containers on the same node as the service pod	The access failed.	The access failed.	The access failed.	The access failed.
		Other containers on different nodes from the service pod	The access failed.	The access failed.	The access failed.	The access failed.

Service Type Released on the Server	Access Type	Request Initiation Location on the Client	Tunnel Network Cluster (IPVS)	VPC Network Cluster (IPVS)	Tunnel Network Cluster (iptables)	VPC Network Cluster (iptables)
nginx-ingress add-on connected with a dedicated load balancer (Local)	Private network	Same node as cceaddon-nginx-ingress-controller pod	The access failed.	The access failed.	The access failed.	The access failed.
		Other containers on the same node as the cceaddon-nginx-ingress-controller pod	The access failed.	The access failed.	The access failed.	The access failed.

The following methods can be used to solve this problem:

- **(Recommended)** In the cluster, use the ClusterIP Service or service domain name for access.
- Set **externalTrafficPolicy** of the Service to **Cluster**, which means cluster-level service affinity. Note that this affects source address persistence.

```
apiVersion: v1
kind: Service
metadata:
  annotations:
    kubernetes.io/elb.class: union
    kubernetes.io/elb.autocreate: '{"type":"public","bandwidth_name":"cce-bandwidth","bandwidth_chargemode":"bandwidth","bandwidth_size":5,"bandwidth_sharetype":"PER","eip_type":"5_bgp","name":"james"}'
  labels:
    app: nginx
    name: nginx
spec:
  externalTrafficPolicy: Cluster
  ports:
  - name: service0
    port: 80
    protocol: TCP
    targetPort: 80
  selector:
    app: nginx
  type: LoadBalancer
```

## 7.3.2 ClusterIP

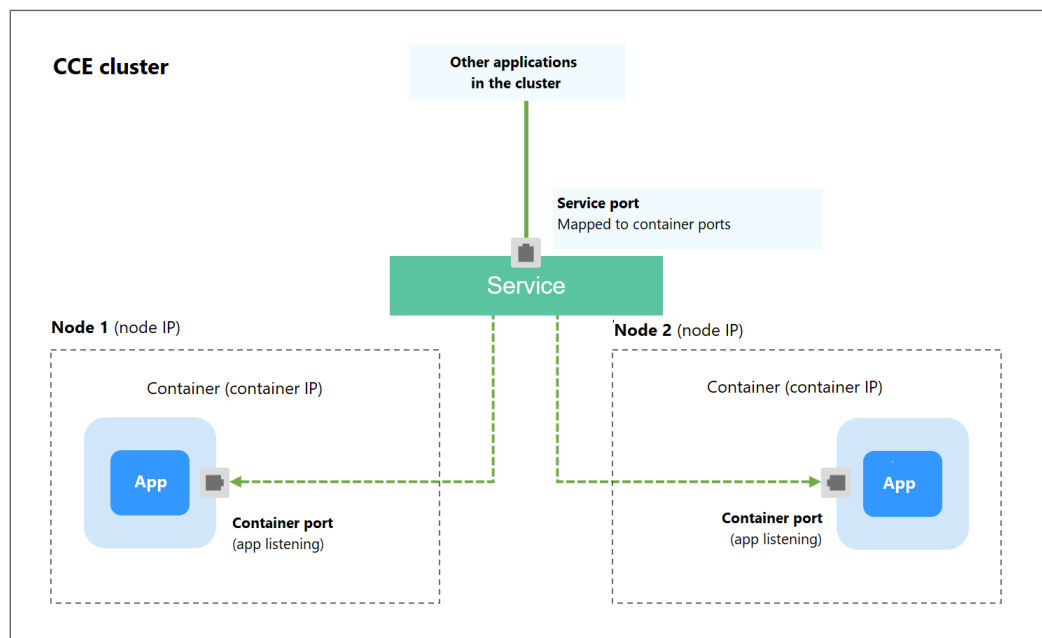
### Scenario

ClusterIP Services allow workloads in the same cluster to use their cluster-internal domain names to access each other.

The cluster-internal domain name format is *<Service name>.<Namespace of the workload>.svc.cluster.local:<Port>*, for example, **nginx.default.svc.cluster.local:80**.

[Figure 7-26](#) shows the mapping relationships between access channels, container ports, and access ports.

**Figure 7-26** Intra-cluster access (ClusterIP)



### Creating a ClusterIP Service

- 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 **Services & Ingresses**. In the upper right corner, click **Create Service**.
- Step 3** Configure intra-cluster access parameters.
  - **Service Name:** Specify a Service name, which can be the same as the workload name.
  - **Service Type:** Select **ClusterIP**.
  - **Namespace:** namespace that the workload belongs to.
  - **Selector:** Add a label and click **Confirm**. The Service will use this label to select pods. You can also click **Reference Workload Label** to use the label of an existing workload. In the dialog box that is displayed, select a workload and click **OK**.

- **Ports**
  - **Protocol:** protocol used by the Service.
  - **Service Port:** port used by the Service. The port number ranges from 1 to 65535.
  - **Container Port:** listener port of the workload. For example, Nginx uses port 80 by default.

**Step 4** Click **OK**.

----End

## Setting the Access Type Using kubectl

You can configure Service access using kubectl. This section uses an Nginx workload as an example to describe how to implement intra-cluster access using kubectl.

**Step 1** Use kubectl to access the cluster. For details, see [Connecting to a Cluster Using kubectl](#).

**Step 2** Create and edit the **nginx-deployment.yaml** and **nginx-clusterip-svc.yaml** files.

The file names are user-defined. **nginx-deployment.yaml** and **nginx-clusterip-svc.yaml** are merely example file names.

### vi nginx-deployment.yaml

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx
spec:
  replicas: 1
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
        - image: nginx:latest
          name: nginx
          imagePullSecrets:
            - name: default-secret
```

### vi nginx-clusterip-svc.yaml

```
apiVersion: v1
kind: Service
metadata:
  labels:
    app: nginx
  name: nginx-clusterip
spec:
  ports:
    - name: service0
      port: 8080 # 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 a Service to access the target container. This port is closely related
                    # to the applications running in a 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. In this example, select the pod with the app:nginx label.
    app: nginx
```

```
type: ClusterIP          # Type of a Service. ClusterIP indicates that a Service is only reachable from within the cluster.
```

### Step 3 Create a workload.

#### **kubectl create -f nginx-deployment.yaml**

If information similar to the following is displayed, the workload has been created.

```
deployment "nginx" created
```

#### **kubectl get po**

If information similar to the following is displayed, the workload is running.

NAME	READY	STATUS	RESTARTS	AGE
nginx-2601814895-znhbr	1/1	Running	0	15s

### Step 4 Create a Service.

#### **kubectl create -f nginx-clusterip-svc.yaml**

If information similar to the following is displayed, the Service is being created:

```
service "nginx-clusterip" created
```

#### **kubectl get svc**

If information similar to the following is displayed, the Service has been created, and a cluster-internal IP address has been assigned to the Service.

```
# kubectl get svc
NAME          TYPE        CLUSTER-IP   EXTERNAL-IP  PORT(S)    AGE
kubernetes   ClusterIP   10.247.0.1   <none>       443/TCP    4d6h
nginx-clusterip ClusterIP   10.247.74.52 <none>       8080/TCP   14m
```

### Step 5 Access the Service.

A Service can be accessed from containers or nodes in a cluster.

Create a pod, access the pod, and run the **curl** command to access *IP address:Port* or the domain name of the Service, as shown in the following figure.

The domain name suffix can be omitted. In the same namespace, you can directly use **nginx-clusterip:8080** for access. In other namespaces, you can use **nginx-clusterip.default:8080** for access.

```
# kubectl run -i --tty --image nginx:alpine test --rm /bin/sh
If you do not see a command prompt, try pressing Enter.
/ # curl 10.247.74.52:8080
<!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>
<p>If you see this page, the nginx web server is successfully installed and working. Further configuration is required.</p>
<p>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>.</p>

<p><em>Thank you for using nginx.</em></p>
</body>
</html>
/ # curl nginx-clusterip.default.svc.cluster.local:8080
...
<h1>Welcome to nginx!</h1>
...
/ # curl nginx-clusterip.default:8080
...
<h1>Welcome to nginx!</h1>
...
/ # curl nginx-clusterip:8080
...
<h1>Welcome to nginx!</h1>
...

```

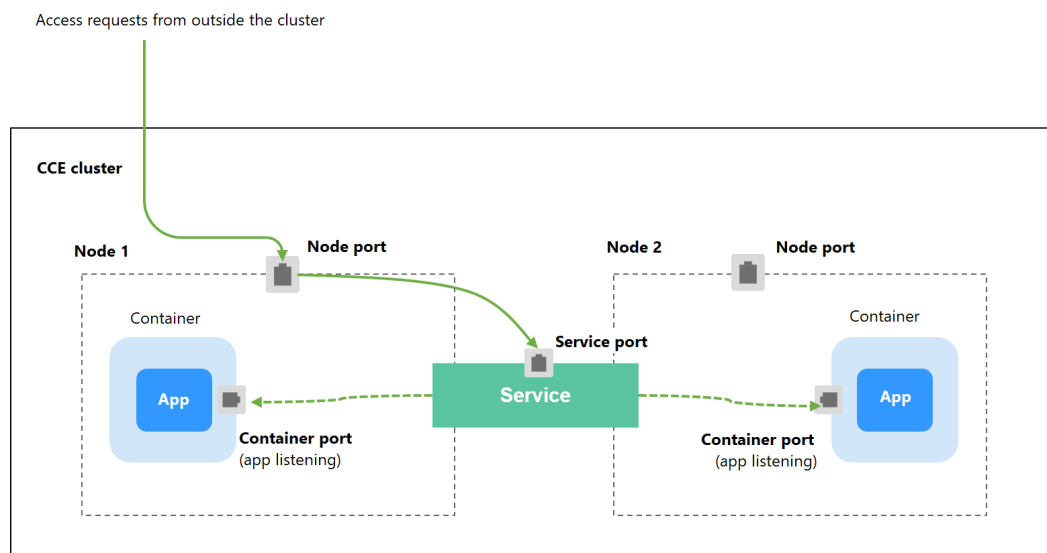
----End

### 7.3.3 NodePort

#### Scenario

A Service is exposed on each node's IP address at a static port (NodePort). When you create a NodePort Service, Kubernetes automatically allocates an internal IP address (ClusterIP) of the cluster. When clients outside the cluster access <NodeIP>:<NodePort>, the traffic will be forwarded to the target pod through the ClusterIP of the NodePort Service.

**Figure 7-27** NodePort access



#### Notes and Constraints

- By default, a NodePort Service is accessed within a VPC. To use an EIP to access a NodePort Service through public networks, bind an EIP to the node in the cluster in advance.
- After a Service is created, if the affinity setting is switched from the cluster level to the node level, the connection tracing table will not be cleared. Do



not modify the Service affinity setting after the Service is created. To modify it, create a Service again.

- In VPC network mode, when container A is published through a NodePort service and the service affinity is set to the node level (that is, **externalTrafficPolicy** is set to **local**), container B deployed on the same node cannot access container A through the node IP address and NodePort service.
- When a NodePort service is created in a cluster of v1.21.7 or later, the port on the node is not displayed using **netstat** by default. If the cluster forwarding mode is **iptables**, run the **iptables -t nat -L** command to view the port. If the cluster forwarding mode is **IPVS**, run the **ipvsadm -Ln** command to view the port.

## Creating a NodePort Service

**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 **Services & Ingresses**. In the upper right corner, click **Create Service**.

**Step 3** Configure intra-cluster access parameters.

- **Service Name:** Specify a Service name, which can be the same as the workload name.
- **Service Type:** Select **NodePort**.
- **Namespace:** namespace that the workload belongs to.
- **Service Affinity:** For details, see [externalTrafficPolicy \(Service Affinity\)](#).
  - **Cluster level:** The IP addresses and access ports of all nodes in a cluster can 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 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.
- **Selector:** Add a label and click **Confirm**. The Service will use this label to select pods. You can also click **Reference Workload Label** to use the label of an existing workload. In the dialog box that is displayed, select a workload and click **OK**.
- **Ports**
  - **Protocol:** protocol used by the Service.
  - **Service Port:** port used by the Service. The port number ranges from 1 to 65535.
  - **Container Port:** listener port of the workload. For example, Nginx uses port 80 by default.
  - **Node Port:** You are advised to select **Auto**. You can also specify a port. The default port ranges from 30000 to 32767.

**Step 4** Click **OK**.

----End

## Using kubectl

You can configure Service access using kubectl. This section uses an Nginx workload as an example to describe how to configure a NodePort Service using kubectl.

**Step 1** Use kubectl to access the cluster. For details, see [Connecting to a Cluster Using kubectl](#).

**Step 2** Create and edit the **nginx-deployment.yaml** and **nginx-nodeport-svc.yaml** files.

The file names are user-defined. **nginx-deployment.yaml** and **nginx-nodeport-svc.yaml** are merely example file names.

### vi nginx-deployment.yaml

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx
spec:
  replicas: 1
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - image: nginx:latest
        name: nginx
        imagePullSecrets:
        - name: default-secret
```

### vi nginx-nodeport-svc.yaml

```
apiVersion: v1
kind: Service
metadata:
  labels:
    app: nginx
  name: nginx-nodeport
spec:
  ports:
  - name: service
    nodePort: 30000 # Node port. The value ranges from 30000 to 32767.
    port: 8080 # 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 a Service to access the target container. This port is closely related to the applications running in a 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. In this example, select the pod with the app:nginx label.
    app: nginx
  type: NodePort # Service type. NodePort indicates that the Service is accessed through a node port.
```

**Step 3** Create a workload.

### kubectl create -f nginx-deployment.yaml

If information similar to the following is displayed, the workload has been created.

```
deployment "nginx" created
```

### kubectl get po

If information similar to the following is displayed, the workload is running.

```
NAME                READY   STATUS    RESTARTS   AGE
nginx-2601814895-qhxqv 1/1     Running   0           9s
```

#### Step 4 Create a Service.

##### **kubectl create -f nginx-nodeport-svc.yaml**

If information similar to the following is displayed, the Service is being created:

```
service "nginx-nodeport" created
```

##### **kubectl get svc**

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

```
# kubectl get svc
NAME                TYPE        CLUSTER-IP   EXTERNAL-IP   PORT(S)        AGE
kubernetes          ClusterIP   10.247.0.1   <none>        443/TCP        4d8h
nginx-nodeport      NodePort    10.247.30.40 <none>        8080:30000/TCP 18s
```

#### Step 5 Access the Service.

By default, a NodePort Service can be accessed by using *Any node IP address:Node port*.

The Service can be accessed from a node in another cluster in the same VPC or in another pod in the cluster. If a public IP address is bound to the node, you can also use the public IP address to access the Service. Create a container in the cluster and access the container by using *Node IP address:Node port*.

```
# kubectl get node -owide
NAME                STATUS    ROLES    AGE   INTERNAL-IP   EXTERNAL-IP   OS-IMAGE      KERNEL-VERSION   CONTAINER-RUNTIME
10.100.0.136       Ready    <none>   152m  10.100.0.136  <none>        CentOS Linux 7 (Core)
3.10.0-1160.25.1.el7.x86_64 docker://18.9.0
10.100.0.5         Ready    <none>   152m  10.100.0.5    <none>        CentOS Linux 7 (Core)
3.10.0-1160.25.1.el7.x86_64 docker://18.9.0
# kubectl run -i --tty --image nginx:alpine test --rm /bin/sh
If you do not see a command prompt, try pressing Enter.
/ # curl 10.100.0.136:30000
<!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>
<p>If you see this page, the nginx web server is successfully installed and working. Further configuration is required.</p>

<p>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>.</p>

<p><em>Thank you for using nginx.</em></p>
</body>
</html>
/ #
```

----End

## 7.3.4 LoadBalancer

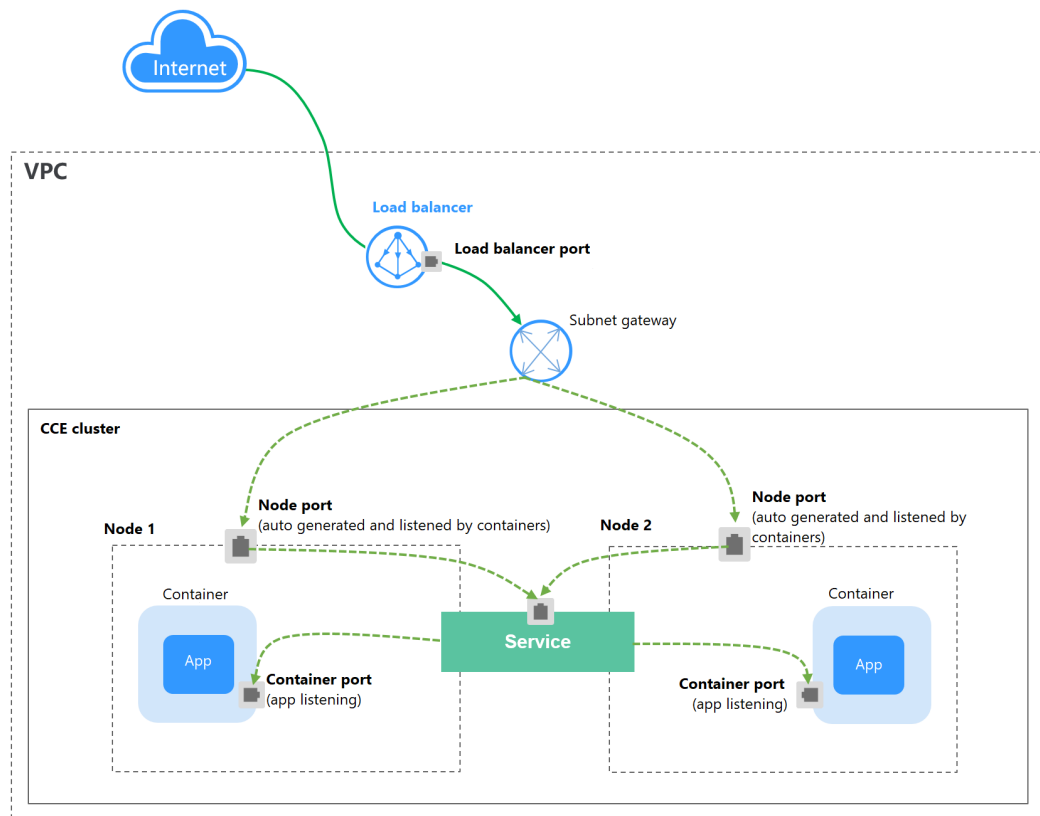
### 7.3.4.1 Creating a LoadBalancer Service

#### Scenario

LoadBalancer Services can access workloads from the public network through a load balancer, which is more reliable than EIP-based access. The LoadBalancer access address is in the format of *IP address of public network load balancer.Access port*, for example, **10.117.117.117:80**.

In this access mode, requests are transmitted through an ELB load balancer to a node and then forwarded to the destination pod through the Service.

Figure 7-28 LoadBalancer



#### Notes and Constraints

- LoadBalancer Services allow workloads to be accessed from public networks through ELB. This access mode has the following restrictions:
  - Automatically created load balancers should not be used by other resources. Otherwise, these load balancers cannot be completely deleted.
  - Do not change the listener name for the load balancer in clusters of v1.15 and earlier. Otherwise, the load balancer cannot be accessed.
- After a Service is created, if the affinity setting is switched from the cluster level to the node level, the connection tracing table will not be cleared. Do

not modify the Service affinity setting after the Service is created. To modify it, create a Service again.

- If service affinity is set to the node level (that is, [externalTrafficPolicy](#) is set to **Local**), the cluster may fail to access the Service by using the ELB address. For details, see [Why a Service Fail to Be Accessed from Within the Cluster](#).
- When the cluster service forwarding (proxy) mode is IPVS, the node IP cannot be configured as the external IP of the Service. Otherwise, the node is unavailable.
- In a cluster using the IPVS proxy mode, if the ingress and Service use the same ELB load balancer, the ingress cannot be accessed from the nodes and containers in the cluster because kube-proxy mounts the LoadBalancer Service address to the ipvs-0 bridge. This bridge intercepts the traffic of the load balancer connected to the ingress. Use different load balancers for the ingress and Service.

## Using the Console

**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 **Services & Ingresses**. In the upper right corner, click **Create Service**.

**Step 3** Configure parameters.

- **Service Name:** Specify a Service name, which can be the same as the workload name.
- **Service Type:** Select **LoadBalancer**.
- **Namespace:** namespace that the workload belongs to.
- **Service Affinity:** For details, see [externalTrafficPolicy \(Service Affinity\)](#).
  - **Cluster level:** The IP addresses and access ports of all nodes in a cluster can 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 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.
- **Selector:** Add a label and click **Confirm**. The Service will use this label to select pods. You can also click **Reference Workload Label** to use the label of an existing workload. In the dialog box that is displayed, select a workload and click **OK**.
- **Load Balancer:** Select a load balancer type and creation mode.  
A load balancer can be dedicated or shared. A dedicated load balancer supports **Network (TCP/UDP)**, **Application (HTTP/HTTPS)**, or **Network (TCP/UDP) & Application (HTTP/HTTPS)**.

You can select **Use existing** or **Auto create** to obtain a load balancer. For details about the configuration of different creation modes, see [Table 7-14](#).

**Table 7-14** Load balancer configurations

How to Create	Configuration
Use existing	Only the load balancers in the same VPC as the cluster can be selected. If no load balancer is available, click <b>Create Load Balancer</b> to create one on the ELB console.
Auto create	<ul style="list-style-type: none"> <li>- <b>Instance Name:</b> Enter a load balancer name.</li> <li>- <b>EIP:</b> If you select <b>Auto create</b>, you can configure the size of the public network bandwidth.</li> <li>- <b>Resource Tag:</b> You can add resource tags to classify resources. You can create <b>predefined tags</b> on the TMS console. The predefined tags are available to all resources that support tags. You can use these tags to improve the tag creation and resource migration efficiency.</li> </ul>

**Set ELB:** You can click **Edit** and configure the load balancing algorithm and sticky session.

- **Algorithm:** Three algorithms are available: weighted round robin, weighted least connections algorithm, or source IP hash.

 **NOTE**

- **Weighted round robin:** Requests are forwarded to different servers based on their weights, which indicate server processing performance. Backend servers with higher weights receive proportionately more requests, whereas equal-weighted servers receive the same number of requests. This algorithm is often used for short connections, such as HTTP services.
  - **Weighted least connections:** In addition to the weight assigned to each server, the number of connections processed by each backend server is considered. Requests are forwarded to the server with the lowest connections-to-weight ratio. Building on **least connections**, the **weighted least connections** algorithm assigns a weight to each server based on their processing capability. This algorithm is often used for persistent connections, such as database connections.
  - **Source IP hash:** The source IP address of each request is calculated using the hash algorithm to obtain a unique hash key, and all backend servers are numbered. The generated key allocates the client to a particular server. This enables requests from different clients to be distributed in load balancing mode and ensures that requests from the same client are forwarded to the same server. This algorithm applies to TCP connections without cookies.
- **Sticky Session:** This function is disabled by default. You can use source IP addresses. Source IP address-based sticky session means that access requests from the same IP address are forwarded to the same backend server.

 **NOTE**

When the **distribution policy** uses the source IP hash, sticky session cannot be set.

- **Health Check:** Configure health check for the load balancer.
  - **Global health check:** applies only to ports using the same protocol. You are advised to select **Custom health check**.
  - **Custom health check:** applies to **ports** using different protocols. For details about the YAML configuration for custom health check, see [Configuring Health Check on Multiple Ports of a LoadBalancer Service](#).

**Table 7-15** Health check parameters

Parameter	Description
Protocol	<p>When the protocol of <b>Port</b> is set to <b>TCP</b>, the TCP and HTTP protocols are supported. When the protocol of <b>Port</b> is set to <b>UDP</b>, the UDP protocol is supported.</p> <ul style="list-style-type: none"> <li>– <b>Check Path</b> (supported only by HTTP for health check): specifies the health check URL. The check path must start with a slash (/) and contain 1 to 80 characters.</li> </ul>
Port	<p>By default, the service port (NodePort or container port of the Service) is used for health check. You can also specify another port for health check. After the port is specified, a service port named <b>cce-healthz</b> will be added for the Service.</p> <ul style="list-style-type: none"> <li>– <b>Node Port:</b> If a shared load balancer is used or no ENI instance is associated, the node port is used as the health check port. If this parameter is not specified, a random port is used. The value ranges from 30000 to 32767.</li> <li>– <b>Container Port:</b> When a dedicated load balancer is associated with an ENI instance, the container port is used for health check. The value ranges from 1 to 65535.</li> </ul>
Check Period (s)	Specifies the maximum interval between health checks. The value ranges from 1 to 50.
Timeout (s)	Specifies the maximum timeout duration for each health check. The value ranges from 1 to 50.
Max. Retries	Specifies the maximum number of health check retries. The value ranges from 1 to 10.

- **Ports**
  - **Protocol:** protocol used by the Service.
  - **Container Port:** listener port of the workload. For example, Nginx uses port 80 by default.
  - **Service Port:** port used by the Service. The port number ranges from 1 to 65535.
  - **Frontend Protocol:** the frontend protocol of the load balancer listener for establishing a traffic distribution connection with the client. When a dedicated load balancer is selected, HTTP/HTTPS can be configured only when **Application (HTTP/HTTPS)** is selected.

- **Health Check:** If **Health Check** is set to **Custom health check**, you can configure health check for ports using different protocols. For details, see **Table 7-15**.

 **NOTE**

When a LoadBalancer Service is created, a random node port number (NodePort) is automatically generated.

- **Listener**

- **Server Certificate:** If **HTTPS/TLS** is enabled on the listener port, you must select a server certificate.
- **SNI:** If **HTTPS/TLS** is enabled on the listener port, you must determine whether to add an SNI certificate. Before adding an SNI certificate, ensure the certificate contains a domain name.

If an SNI certificate cannot be found based on the domain name requested by the client, the server certificate will be returned by default.

- **Advanced Options**

Configuration	Description	Restrictions
Idle Timeout (s)	Timeout for an idle client connection. If there are no requests reaching the load balancer during the timeout duration, the load balancer will disconnect the connection from the client and establish a new connection when there is a new request.	This configuration is not supported if the port of a shared load balancer uses UDP.
Request Timeout (s)	Timeout for waiting for a request from a client. There are two cases: <ul style="list-style-type: none"> <li>▪ If the client fails to send a request header to the load balancer during the timeout duration, the request will be interrupted.</li> <li>▪ If the interval between two consecutive request bodies reaching the load balancer is greater than the timeout duration, the connection will be disconnected.</li> </ul>	This parameter is available only after <b>HTTP/HTTPS</b> is enabled on ports.



Configuration	Description	Restrictions
Response Timeout (s)	Timeout for waiting for a response from a backend server. After a request is forwarded to the backend server, if the backend server does not respond during the timeout duration, the load balancer will stop waiting and return HTTP 504 Gateway Timeout.	This parameter is available only after <a href="#">HTTP/HTTPS</a> is enabled on ports.
HTTP2	Whether to use HTTP/2 for a client to communicate with a load balancer. Request forwarding using HTTP/2 improves the access performance between your application and the load balancer. However, the load balancer still uses HTTP/1.x to forward requests to the backend server.	This parameter is available only after <a href="#">HTTPS</a> is enabled on ports.

- **Annotation:** The LoadBalancer Service has some advanced CCE functions, which are implemented by annotations. For details, see [Configuring LoadBalancer Services Using Annotations](#).

**Step 4** Click **OK**.

----End

## Using kubectl to Create a Service (Using an Existing Load Balancer)

You can configure Service access using kubectl when creating a workload. This section uses an Nginx workload as an example to describe how to add a LoadBalancer Service using kubectl.

**Step 1** Use kubectl to access the cluster. For details, see [Connecting to a Cluster Using kubectl](#).

**Step 2** Create and edit the `nginx-deployment.yaml` and `nginx-elb-svc.yaml` files.

The file names are user-defined. `nginx-deployment.yaml` and `nginx-elb-svc.yaml` are merely example file names.

**vi nginx-deployment.yaml**

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx
spec:
  replicas: 1
  selector:
    matchLabels:
      app: nginx
```

```
template:
  metadata:
    labels:
      app: nginx
  spec:
    containers:
      - image: nginx
        name: nginx
    imagePullSecrets:
      - name: default-secret
```

## vi nginx-elb-svc.yaml

### NOTE

To enable sticky session, ensure anti-affinity is configured for the workload pods so that the pods are deployed onto different nodes. For details, see [Configuring Workload Affinity or Anti-affinity Scheduling \(podAffinity or podAntiAffinity\)](#).

```
apiVersion: v1
kind: Service
metadata:
  name: nginx
  annotations:
    kubernetes.io/elb.id: <your_elb_id> # Load balancer ID. Replace it with the actual
value.
    kubernetes.io/elb.class: # Load balancer type
    kubernetes.io/elb.lb-algorithm: ROUND_ROBIN # Load balancer algorithm
    kubernetes.io/elb.session-affinity-mode: SOURCE_IP # The sticky session type is source IP address.
    kubernetes.io/elb.session-affinity-option: '{"persistence_timeout": "30"}' # Stickiness duration (min)
    kubernetes.io/elb.health-check-flag: 'on' # Enable ELB health check.
    kubernetes.io/elb.health-check-option: '{
  "protocol": "TCP",
  "delay": "5",
  "timeout": "10",
  "max_retries": "3"
}'
spec:
  selector:
    app: nginx
  ports:
    - name: service0
      port: 80 # Port for accessing the Service, which is also the listener port on the load balancer.
      protocol: TCP
      targetPort: 80 # Port used by a Service to access the target container. This port is closely related to the
applications running in a container.
      nodePort: 31128 # Port number of the node. If this parameter is not specified, a random port number
ranging from 30000 to 32767 is generated.
      type: LoadBalancer
```

The preceding example uses annotations to implement some advanced functions of load balancing, such as sticky session and health check. For details, see [Table 7-16](#).

For more annotations and examples related to advanced functions, see [Configuring LoadBalancer Services Using Annotations](#).

**Table 7-16** annotations parameters

Parameter	Mandatory	Type	Description
kubernetes.io/elb.id	Yes	String	<p>ID of a load balancer.</p> <p>Mandatory when an existing load balancer is to be associated.</p> <p><b>How to obtain:</b></p> <p>On the management console, click <b>Service List</b>, and choose <b>Networking &gt; Elastic Load Balance</b>. Click the name of the target load balancer. On the <b>Summary</b> tab page, find and copy the ID.</p> <p><b>NOTE</b></p> <p>The system preferentially connects to the load balancer based on the <b>kubernetes.io/elb.id</b> field. If this field is not specified, the <b>spec.loadBalancerIP</b> field is used (optional and available only in 1.23 and earlier versions).</p> <p>Do not use the <b>spec.loadBalancerIP</b> field to connect to the load balancer. This field will be discarded by Kubernetes. For details, see <a href="#">Deprecation</a>.</p>
kubernetes.io/elb.class	Yes	String	<p>Select a proper load balancer type.</p> <p>The value can be:</p> <ul style="list-style-type: none"> <li>• <b>performance</b>: dedicated load balancer, which can be used only in clusters of v1.17 and later.</li> </ul> <p><b>NOTE</b></p> <p>If a LoadBalancer Service accesses an existing dedicated load balancer, the dedicated load balancer must support TCP/UDP networking.</p>
kubernetes.io/elb.lb-algorithm	No	String	<p>Specifies the load balancing algorithm of the backend server group. The default value is <b>ROUND_ROBIN</b>.</p> <p>Options:</p> <ul style="list-style-type: none"> <li>• <b>ROUND_ROBIN</b>: weighted round robin algorithm</li> <li>• <b>LEAST_CONNECTIONS</b>: weighted least connections algorithm</li> <li>• <b>SOURCE_IP</b>: source IP hash algorithm</li> </ul> <p><b>NOTE</b></p> <p>If this parameter is set to <b>SOURCE_IP</b>, the weight setting (<b>weight</b> field) of backend servers bound to the backend server group is invalid, and sticky session cannot be enabled.</p>

Parameter	Mandatory	Type	Description
kubernetes.io/elb.session-affinity-mode	No	String	Source IP address-based sticky session means that access requests from the same IP address are forwarded to the same backend server. <ul style="list-style-type: none"> <li>Disabling sticky session: Do not configure this parameter.</li> <li>Enabling sticky session: Set this parameter to <b>SOURCE_IP</b>, indicating that the sticky session is based on the source IP address.</li> </ul> <b>NOTE</b> When <b>kubernetes.io/elb.lb-algorithm</b> is set to <b>SOURCE_IP</b> (source IP hash), sticky session cannot be enabled.
kubernetes.io/elb.session-affinity-option	No	<a href="#">Table 7-17</a> object	Sticky session timeout.
kubernetes.io/elb.health-check-flag	No	String	Whether to enable the ELB health check. <ul style="list-style-type: none"> <li>Enabling health check: Leave blank this parameter or set it to <b>on</b>.</li> <li>Disabling health check: Set this parameter to <b>off</b>.</li> </ul> If this parameter is enabled, the <a href="#">kubernetes.io/elb.health-check-option</a> field must also be specified at the same time.
kubernetes.io/elb.health-check-option	No	<a href="#">Table 7-18</a> object	ELB health check configuration items.

**Table 7-17** elb.session-affinity-option data structure

Parameter	Mandatory	Type	Description
persistenc e_timeout	Yes	String	Sticky session timeout, in minutes. This parameter is valid only when <b>elb.session-affinity-mode</b> is set to <b>SOURCE_IP</b> . Value range: 1 to 60. Default value: <b>60</b>

**Table 7-18** elb.health-check-option data structure

Parameter	Mandatory	Type	Description
delay	No	String	Health check interval (s) Value range: 1 to 50. Default value: <b>5</b>
timeout	No	String	Health check timeout, in seconds. Value range: 1 to 50. Default value: <b>10</b>
max_retries	No	String	Maximum number of health check retries. Value range: 1 to 10. Default value: <b>3</b>
protocol	No	String	Health check protocol. Value options: TCP or HTTP
path	No	String	Health check URL. This parameter needs to be configured when the protocol is <b>HTTP</b> . Default value: / Value range: 1-80 characters

**Step 3** Create a workload.

**kubectl create -f nginx-deployment.yaml**

If information similar to the following is displayed, the workload has been created.

```
deployment/nginx created
```

**kubectl get pod**

If information similar to the following is displayed, the workload is running.

```
NAME                READY   STATUS    RESTARTS   AGE
nginx-2601814895-c1xhw 1/1     Running   0           6s
```

**Step 4** Create a Service.

**kubectl create -f nginx-elb-svc.yaml**

If information similar to the following is displayed, the Service has been created.

```
service/nginx created
```

**kubectl get svc**

If information similar to the following is displayed, the workload's access mode has been configured.

```
NAME      TYPE        CLUSTER-IP   EXTERNAL-IP  PORT(S)    AGE
kubernetes ClusterIP  10.247.0.1   <none>       443/TCP    3d
nginx     LoadBalancer 10.247.130.196 10.78.42.242 80:31540/TCP 51s
```

**Step 5** Enter the URL in the address box of the browser, for example, **10.78.42.242:80**. **10.78.42.242** indicates the IP address of the load balancer, and **80** indicates the access port displayed on the CCE console.

The Nginx is accessible.

**Figure 7-29** Accessing Nginx through the LoadBalancer Service

## Welcome to nginx!

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 [nginx.org](https://nginx.org).  
Commercial support is available at [nginx.com](https://nginx.com).

*Thank you for using nginx.*

----End

### Using kubectl to Create a Service (Automatically Creating a Load Balancer)

You can configure Service access using kubectl when creating a workload. This section uses an Nginx workload as an example to describe how to add a LoadBalancer Service using kubectl.

**Step 1** Use kubectl to access the cluster. For details, see [Connecting to a Cluster Using kubectl](#).

**Step 2** Create and edit the `nginx-deployment.yaml` and `nginx-elb-svc.yaml` files.

The file names are user-defined. `nginx-deployment.yaml` and `nginx-elb-svc.yaml` are merely example file names.

#### vi nginx-deployment.yaml

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx
spec:
  replicas: 1
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
        - image: nginx
          name: nginx
      imagePullSecrets:
        - name: default-secret
```

#### vi nginx-elb-svc.yaml

##### NOTE

To enable sticky session, ensure anti-affinity is configured for the workload pods so that the pods are deployed onto different nodes. For details, see [Configuring Workload Affinity or Anti-affinity Scheduling \(podAffinity or podAntiAffinity\)](#).

Example Service using a public network dedicated load balancer (only for clusters of v1.17 and later):

```
apiVersion: v1
kind: Service
metadata:
  name: nginx
  labels:
    app: nginx
  namespace: default
  annotations:
    kubernetes.io/elb.class: performance
    kubernetes.io/elb.autocreate: '{
      "type": "public",
      "bandwidth_name": "cce-bandwidth-1626694478577",
      "bandwidth_chargemode": "bandwidth",
      "bandwidth_size": 5,
      "bandwidth_sharetype": "PER",
      "eip_type": "5_bgp",
      "vip_subnet_cidr_id": "*****",
      "vip_address": "***.**.**.**",

      "available_zone": [
        ""
      ],
      "l4_flavor_name": "L4_flavor.elb.s1.small"
    }'
    kubernetes.io/elb.enterpriseID: '0' # ID of the enterprise project to which the load balancer belongs
    kubernetes.io/elb.lb-algorithm: ROUND_ROBIN # Load balancer algorithm
    kubernetes.io/elb.session-affinity-mode: SOURCE_IP # The sticky session type is source IP address.
    kubernetes.io/elb.session-affinity-option: '{"persistence_timeout": "30"}' # Stickiness duration (min)
    kubernetes.io/elb.health-check-flag: 'on' # Enable ELB health check.
    kubernetes.io/elb.health-check-option: '{
      "protocol": "TCP",
      "delay": "5",
      "timeout": "10",
      "max_retries": "3"
    }'
spec:
  selector:
    app: nginx
  ports:
    - name: cce-service-0
      targetPort: 80
      nodePort: 0
      port: 80
      protocol: TCP
  type: LoadBalancer
```

The preceding example uses annotations to implement some advanced functions of load balancing, such as sticky session and health check. For details, see [Table 7-19](#).

For more annotations and examples related to advanced functions, see [Configuring LoadBalancer Services Using Annotations](#).

**Table 7-19** annotations parameters

Parameter	Mandatory	Type	Description
kubernetes.io/elb.class	Yes	String	Select a proper load balancer type. The value can be: <ul style="list-style-type: none"> <li><b>performance</b>: dedicated load balancer, which can be used only in clusters of v1.17 and later.</li> </ul>
kubernetes.io/elb.autocreate	Yes	<a href="#">elb.autocreate</a> object	Whether to automatically create a load balancer associated with the Service. <b>Example</b> <ul style="list-style-type: none"> <li>Automatically created shared load balancer with an EIP bound: <code>{ "type": "public", "bandwidth_name": "cceb-1551163379627", "bandwidth_charge_mode": "bandwidth", "bandwidth_size": 5, "bandwidth_share_type": "PER", "eip_type": "5_bgp", "name": "james" }</code></li> <li>Automatically created shared load balancer with no EIP bound: <code>{ "type": "inner", "name": "A-location-d-test" }</code></li> </ul>
kubernetes.io/elb.subnet-id	None	String	ID of the subnet where the cluster is located. The value can contain 1 to 100 characters. <ul style="list-style-type: none"> <li>Mandatory when a cluster of v1.11.7-r0 or earlier is to be automatically created.</li> <li>Optional for clusters later than v1.11.7-r0.</li> </ul>
kubernetes.io/elb.enterpriseID	No	String	<b>Clusters of v1.15 and later versions support this field. In clusters earlier than v1.15, load balancers are created in the default project by default.</b> This parameter indicates the ID of the enterprise project in which the ELB load balancer will be created. If this parameter is not specified or is set to <b>0</b> , resources will be bound to the default enterprise project. <b>How to obtain:</b> Log in to the management console and choose <b>Enterprise &gt; Project Management</b> on the top menu bar. In the list displayed, click the name of the target enterprise project, and copy the ID on the enterprise project details page.



Parameter	Mandatory	Type	Description
kubernetes.io/elb.lb-algorithm	No	String	<p>Specifies the load balancing algorithm of the backend server group. The default value is <b>ROUND_ROBIN</b>.</p> <p>Options:</p> <ul style="list-style-type: none"> <li>• <b>ROUND_ROBIN</b>: weighted round robin algorithm</li> <li>• <b>LEAST_CONNECTIONS</b>: weighted least connections algorithm</li> <li>• <b>SOURCE_IP</b>: source IP hash algorithm</li> </ul> <p><b>NOTE</b> If this parameter is set to <b>SOURCE_IP</b>, the weight setting (<b>weight</b> field) of backend servers bound to the backend server group is invalid, and sticky session cannot be enabled.</p>
kubernetes.io/elb.session-affinity-mode	No	String	<p>Source IP address-based sticky session means that access requests from the same IP address are forwarded to the same backend server.</p> <ul style="list-style-type: none"> <li>• Disabling sticky session: Do not configure this parameter.</li> <li>• Enabling sticky session: Set this parameter to <b>SOURCE_IP</b>, indicating that the sticky session is based on the source IP address.</li> </ul> <p><b>NOTE</b> When <b>kubernetes.io/elb.lb-algorithm</b> is set to <b>SOURCE_IP</b> (source IP hash), sticky session cannot be enabled.</p>
kubernetes.io/elb.session-affinity-option	No	<a href="#">Table 7-17</a> object	Sticky session timeout.
kubernetes.io/elb.health-check-flag	No	String	<p>Whether to enable the ELB health check.</p> <ul style="list-style-type: none"> <li>• Enabling health check: Leave blank this parameter or set it to <b>on</b>.</li> <li>• Disabling health check: Set this parameter to <b>off</b>.</li> </ul> <p>If this parameter is enabled, the <a href="#">kubernetes.io/elb.health-check-option</a> field must also be specified at the same time.</p>

Parameter	Mandatory	Type	Description
kubernetes.io/elb.health-check-option	No	<a href="#">Table 7-18</a> object	ELB health check configuration items.

**Table 7-20** elb.autocreate data structure

Parameter	Mandatory	Type	Description
name	No	String	Name of the automatically created load balancer. The value can contain 1 to 64 characters. Only letters, digits, underscores (_), hyphens (-), and periods (.) are allowed. Default: <b>cce-lb+service.UID</b>
type	No	String	Network type of the load balancer. <ul style="list-style-type: none"> <li>• <b>public</b>: public network load balancer</li> <li>• <b>inner</b>: private network load balancer</li> </ul> Default: <b>inner</b>
bandwidth_name	Yes for public network load balancers	String	Bandwidth name. The default value is <b>cce-bandwidth-*****</b> . The value can contain 1 to 64 characters. Only letters, digits, underscores (_), hyphens (-), and periods (.) are allowed.
bandwidth_chargemode	No	String	Bandwidth mode. <ul style="list-style-type: none"> <li>• <b>bandwidth</b>: billed by bandwidth</li> <li>• <b>traffic</b>: billed by traffic</li> </ul> Default: <b>bandwidth</b>

Parameter	Mandatory	Type	Description
bandwidth_size	Yes for public network load balancers	Integer	<p>Bandwidth size. The value ranges from 1 Mbit/s to 2000 Mbit/s by default. Configure this parameter based on the bandwidth range allowed in your region.</p> <p>The minimum increment for bandwidth adjustment varies depending on the bandwidth range.</p> <ul style="list-style-type: none"> <li>• The minimum increment is 1 Mbit/s if the allowed bandwidth does not exceed 300 Mbit/s.</li> <li>• The minimum increment is 50 Mbit/s if the allowed bandwidth ranges from 300 Mbit/s to 1000 Mbit/s.</li> <li>• The minimum increment is 500 Mbit/s if the allowed bandwidth exceeds 1000 Mbit/s.</li> </ul>
bandwidth_sharetype	Yes for public network load balancers	String	<p>Bandwidth sharing mode.</p> <ul style="list-style-type: none"> <li>• <b>PER</b>: dedicated bandwidth</li> </ul>
eip_type	Yes for public network load balancers	String	<p>EIP type.</p> <ul style="list-style-type: none"> <li>• <b>5_bgp</b>: dynamic BGP</li> <li>• <b>5_sbgp</b>: static BGP</li> </ul> <p>The specific type varies with regions. For details, see the EIP console.</p>
vip_subnet_cidr_id	No	String	<p>Subnet where a load balancer is located. The subnet must belong to the VPC where the cluster resides.</p> <p>If this parameter is not specified, the ELB load balancer and the cluster are in the same subnet.</p> <p>This field can be specified only for clusters of v1.21 or later.</p>

Parameter	Mandatory	Type	Description
vip_address	No	String	<p>Private IP address of the load balancer. Only IPv4 addresses are supported.</p> <p>The IP address must be in the ELB CIDR block. If this parameter is not specified, an IP address will be automatically assigned from the ELB CIDR block.</p> <p>This parameter is available only in clusters of v1.23.11-r0, v1.25.6-r0, v1.27.3-r0, or later versions.</p>

**Step 3** Create a workload.

**kubectl create -f nginx-deployment.yaml**

If information similar to the following is displayed, the workload is being created.

```
deployment/nginx created
```

**kubectl get pod**

If information similar to the following is displayed, the workload is running.

```
NAME                READY   STATUS    RESTARTS   AGE
nginx-2601814895-c1xhw 1/1     Running   0           6s
```

**Step 4** Create a Service.

**kubectl create -f nginx-elb-svc.yaml**

If information similar to the following is displayed, the Service has been created.

```
service/nginx created
```

**kubectl get svc**

If information similar to the following is displayed, the workload's access mode has been configured.

```
NAME      TYPE        CLUSTER-IP   EXTERNAL-IP   PORT(S)          AGE
kubernetes ClusterIP  10.247.0.1   <none>        443/TCP          3d
nginx     LoadBalancer 10.247.130.196 10.78.42.242 80:31540/TCP     51s
```

**Step 5** Enter the URL in the address box of the browser, for example, **10.78.42.242:80**. **10.78.42.242** indicates the IP address of the load balancer, and **80** indicates the access port displayed on the CCE console.

The Nginx is accessible.

**Figure 7-30** Accessing Nginx through the LoadBalancer Service

# Welcome to nginx!

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 [nginx.org](http://nginx.org). Commercial support is available at [nginx.com](http://nginx.com).

*Thank you for using nginx.*

----End

## 7.3.4.2 Configuring LoadBalancer Services Using Annotations

You can add annotations to a YAML file to use some CCE advanced functions. This section describes the available annotations when a LoadBalancer service is created.

- [Interconnection with ELB](#)
- [Sticky Session](#)
- [Health Check](#)
- [HTTP or HTTPS](#)
- [SNI](#)
- [Dynamic Adjustment of the Weight of the Backend ECS](#)
- [Passthrough Capability](#)
- [Host Network](#)
- [Timeout](#)
- [HTTP/2](#)

### Interconnection with ELB

**Table 7-21** Annotations for interconnecting with ELB

Parameter	Type	Description	Supported Cluster Version
kubernetes.io/elb.class	String	Select a proper load balancer type. Options: <ul style="list-style-type: none"> <li>• <b>performance</b>: dedicated load balancer, which can be used only in clusters of v1.17 and later.</li> </ul>	v1.9 or later

Parameter	Type	Description	Supported Cluster Version
kubernetes.io/elb.id	String	<p>Mandatory <b>when an existing load balancer is to be associated.</b></p> <p>ID of a load balancer.</p> <p><b>How to obtain:</b></p> <p>On the management console, click <b>Service List</b>, and choose <b>Networking &gt; Elastic Load Balance</b>. Click the name of the target load balancer. On the <b>Summary</b> tab page, find and copy the ID.</p> <p><b>NOTE</b></p> <p>The system preferentially connects to the load balancer based on the <b>kubernetes.io/elb.id</b> field. If this field is not specified, the <b>spec.loadBalancerIP</b> field is used (optional and available only in 1.23 and earlier versions).</p> <p>Do not use the <b>spec.loadBalancerIP</b> field to connect to the load balancer. This field will be discarded by Kubernetes. For details, see <a href="#">Deprecation</a>.</p>	v1.9 or later
kubernetes.io/elb.auto create	<a href="#">Table 7-31</a>	<p>Mandatory <b>when load balancers are automatically created.</b></p> <p><b>Example</b></p> <ul style="list-style-type: none"> <li>Automatically created shared load balancer with an EIP bound:  <pre>{ "type": "public", "bandwidth_name": "cce - bandwidth-1551163379627", "bandwidth_chargemode": "bandwidth", "bandwidth_size": 5, "bandwidth_sharetype": "PER", "eip_type": "5_bgp", "name": "james" }</pre> </li> <li>Automatically created shared load balancer with no EIP bound:  <pre>{ "type": "inner", "name": "A-location-d-test" }</pre> </li> </ul>	v1.9 or later

Parameter	Type	Description	Supported Cluster Version
kubernetes.io/elb.enterpriseID	String	<p>Optional <b>when load balancers are automatically created.</b></p> <p><b>Clusters of v1.15 and later versions support this field. In clusters earlier than v1.15, load balancers are created in the default project by default.</b></p> <p>This parameter indicates the ID of the enterprise project in which the ELB load balancer will be created.</p> <p>If this parameter is not specified or is set to <b>0</b>, resources will be bound to the default enterprise project.</p> <p><b>How to obtain:</b></p> <p>Log in to the management console and choose <b>Enterprise &gt; Project Management</b> on the top menu bar. In the list displayed, click the name of the target enterprise project, and copy the ID on the enterprise project details page.</p>	v1.15 or later
kubernetes.io/elb.subnet-id	String	<p>Optional <b>when load balancers are automatically created.</b></p> <p>ID of the subnet where the cluster is located. The value can contain 1 to 100 characters.</p> <ul style="list-style-type: none"> <li>• Mandatory when a cluster of v1.11.7-r0 or earlier is to be automatically created.</li> <li>• Optional for clusters later than v1.11.7-r0.</li> </ul>	<p>Mandatory for clusters earlier than v1.11.7-r0</p> <p>Discarded in clusters later than v1.11.7-r0</p>
kubernetes.io/elb.lb-algorithm	String	<p>Specifies the load balancing algorithm of the backend server group. The default value is <b>ROUND_ROBIN</b>.</p> <p>Options:</p> <ul style="list-style-type: none"> <li>• <b>ROUND_ROBIN</b>: weighted round robin algorithm</li> <li>• <b>LEAST_CONNECTIONS</b>: weighted least connections algorithm</li> <li>• <b>SOURCE_IP</b>: source IP hash algorithm</li> </ul> <p><b>NOTE</b></p> <p>If this parameter is set to <b>SOURCE_IP</b>, the weight setting (<b>weight</b> field) of backend servers bound to the backend server group is invalid, and sticky session cannot be enabled.</p>	v1.9 or later

The following shows how to use the preceding annotations:

- Associate an existing load balancer. For details, see [Using kubectl to Create a Service \(Using an Existing Load Balancer\)](#).

```

apiVersion: v1
kind: Service
metadata:
  name: nginx
  annotations:
    kubernetes.io/elb.id: <your_elb_id>           # Load balancer ID. Replace it with the
actual value.
    kubernetes.io/elb.class: performance         # Load balancer type
    kubernetes.io/elb.lb-algorithm: ROUND_ROBIN # Load balancer algorithm
spec:
  selector:
    app: nginx
  ports:
  - name: service0
    port: 80
    protocol: TCP
    targetPort: 80
  type: LoadBalancer
  
```

- Automatically create a load balancer. For details, see [Using kubectl to Create a Service \(Automatically Creating a Load Balancer\)](#).

## Sticky Session

**Table 7-22** Annotations for sticky session

Parameter	Type	Description	Supported Cluster Version
kubernetes.io/elb.session-affinity-mode	String	<p>Source IP address-based sticky session means that access requests from the same IP address are forwarded to the same backend server.</p> <ul style="list-style-type: none"> <li>• Disabling sticky session: Do not configure this parameter.</li> <li>• Enabling sticky session: Set this parameter to <b>SOURCE_IP</b>, indicating that the sticky session is based on the source IP address.</li> </ul> <p><b>NOTE</b> When <b>kubernetes.io/elb.lb-algorithm</b> is set to <b>SOURCE_IP</b> (source IP hash), sticky session cannot be enabled.</p>	v1.9 or later
kubernetes.io/elb.session-affinity-option	<a href="#">Table 7-34</a>	Sticky session timeout.	v1.9 or later

The following shows how to use the preceding annotations:



```

apiVersion: v1
kind: Service
metadata:
  name: nginx
  annotations:
    kubernetes.io/elb.id: <your_elb_id>          # Load balancer ID. Replace it with the actual value.
    kubernetes.io/elb.class:                   # Load balancer type
    kubernetes.io/elb.session-affinity-mode: SOURCE_IP    # The sticky session type is source IP
address:
  kubernetes.io/elb.session-affinity-option: '{"persistence_timeout": "30"}' # Stickiness duration
(min)
spec:
  selector:
    app: nginx
  ports:
  - name: service0
    port: 80
    protocol: TCP
    targetPort: 80
  type: LoadBalancer

```

## Health Check

**Table 7-23** Annotations for health check

Parameter	Type	Description	Supported Cluster Version
kubernetes.io/elb.health-check-flag	String	Whether to enable the ELB health check. <ul style="list-style-type: none"> <li>Enabling health check: Leave blank this parameter or set it to <b>on</b>.</li> <li>Disabling health check: Set this parameter to <b>off</b>.</li> </ul> If this parameter is enabled, the <a href="#">kubernetes.io/elb.health-check-option</a> field must also be specified at the same time.	v1.9 or later
kubernetes.io/elb.health-check-option	<a href="#">Table 7-32</a>	ELB health check configuration items.	v1.9 or later
kubernetes.io/elb.health-check-options	<a href="#">Table 7-33</a>	ELB health check configuration items. Each Service port can be configured separately, and you can configure only some ports. <b>NOTE</b> Either <a href="#">kubernetes.io/elb.health-check-option</a> or <a href="#">kubernetes.io/elb.health-check-options</a> can be configured.	v1.19.16-r5 or later v1.21.8-r0 or later v1.23.6-r0 or later v1.25.2-r0 or later

- The following shows how to use **kubernetes.io/elb.health-check-option**:

```

apiVersion: v1
kind: Service
metadata:
  name: nginx
  annotations:
    kubernetes.io/elb.id: <your_elb_id>           # Load balancer ID. Replace it with the actual
value.
    kubernetes.io/elb.class:                    # Load balancer type
    kubernetes.io/elb.health-check-flag: 'on'    # Enable ELB health check.
    kubernetes.io/elb.health-check-option: '{
  "protocol": "TCP",
  "delay": "5",
  "timeout": "10",
  "max_retries": "3"
}'
spec:
  selector:
    app: nginx
  ports:
    - name: service0
      port: 80
      protocol: TCP
      targetPort: 80
  type: LoadBalancer
  
```

- For details about how to use **kubernetes.io/elb.health-check-options**, see [Configuring Health Check on Multiple Ports of a LoadBalancer Service](#).

## HTTP or HTTPS

**Table 7-24** Annotations for using HTTP or HTTPS

Parameter	Type	Description	Supported Cluster Version
kubernetes.io/elb.protocol-port	String	If a Service is HTTP/HTTPS-compliant, configure the protocol and port number in the format of "protocol:port". where, <ul style="list-style-type: none"> <li><b>protocol</b>: specifies the protocol used by the listener port. The value can be <b>http</b> or <b>https</b>.</li> <li><b>ports</b>: Service ports specified by <b>spec.ports[].port</b>.</li> </ul>	v1.19.16 or later
kubernetes.io/elb.cert-id	String	ID of an ELB certificate, which is used as the HTTPS server certificate. To obtain the certificate, log in to the CCE console, choose <b>Service List &gt; Networking &gt; Elastic Load Balance</b> , and click <b>Certificates</b> in the navigation pane. In the load balancer list, copy the ID under the target certificate name.	v1.19.16 or later

For details, see [Configuring HTTP/HTTPS for a LoadBalancer Service](#).

## SNI

**Table 7-25** Annotations for using SNIs

Parameter	Type	Description	Supported Cluster Version
kubernetes.io/elb.tls-certificate-ids	String	In ELB, the IDs of SNI certificates that must contain a domain name are separated by commas (.). To obtain the certificate, log in to the CCE console, choose <b>Service List</b> > <b>Networking</b> > <b>Elastic Load Balance</b> , and click <b>Certificates</b> in the navigation pane. In the load balancer list, copy the ID under the target certificate name.	v1.23.13-r0, v1.25.8-r0, v1.27.5-r0, v1.28.3-r0, or later

HTTPS must be enabled. For details, see [Configuring SNI for a LoadBalancer Service](#).

## Dynamic Adjustment of the Weight of the Backend ECS

**Table 7-26** Annotations for dynamically adjusting the weight of the backend ECS

Parameter	Type	Description	Supported Cluster Version
kubernetes.io/elb.adaptive-weight	String	Dynamically adjust the weight of the load balancer backend server based on the number pods on the server. In this way, the requests received by each pod are more balanced. <ul style="list-style-type: none"><li>• <b>true</b>: enabled</li><li>• <b>false</b>: disabled</li></ul>	v1.21 or later

### NOTE

This parameter is invalid in passthrough networking, where dedicated load balancers are used in a CCE Turbo cluster.

The following shows how to use the preceding annotations:

```
apiVersion: v1
kind: Service
metadata:
  name: nginx
annotations:
  kubernetes.io/elb.id: <your_elb_id> # Load balancer ID. Replace it with the actual value.
  kubernetes.io/elb.class: # Load balancer type
  kubernetes.io/elb.adaptive-weight: 'true' # Enable dynamic adjustment of the weight of
```

```

the backend ECS.
spec:
  selector:
    app: nginx
  ports:
  - name: service0
    port: 80
    protocol: TCP
    targetPort: 80
  type: LoadBalancer

```

## Passthrough Capability

**Table 7-27** Annotations for passthrough capability

Parameter	Type	Description	Supported Cluster Version
kubernetes.io/elb.pass-through	String	Whether the access requests from within the cluster to the Service pass through the ELB load balancer.	v1.19 or later

For details, see [Configuring Passthrough Networking for a LoadBalancer Service](#).

## Host Network

**Table 7-28** Annotations for host network

Parameter	Type	Description	Supported Cluster Version
kubernetes.io/hws-hostNetwork	String	If the pod uses <b>hostNetwork</b> , the ELB forwards the request to the host network after this annotation is used. Options: <ul style="list-style-type: none"> <li><b>true</b>: enabled</li> <li><b>false</b> (default): disabled</li> </ul>	v1.9 or later

The following shows how to use the preceding annotations:

```

apiVersion: v1
kind: Service
metadata:
  name: nginx
  annotations:
    kubernetes.io/elb.id: <your_elb_id> # Load balancer ID. Replace it with the actual value.
    kubernetes.io/elb.class: # Load balancer type
    kubernetes.io/hws-hostNetwork: 'true' # The load balancer forwards the request to the
host network.
spec:

```

```
selector:
  app: nginx
ports:
- name: service0
  port: 80
  protocol: TCP
  targetPort: 80
type: LoadBalancer
```

## Timeout

**Table 7-29** Annotation for configuring timeout

Parameter	Type	Description	Supported Cluster Version
kubernetes.io/elb.keepalive_timeout	String	<p>Timeout for client connections. If there are no requests reaching the load balancer during the timeout duration, the load balancer will disconnect the connection from the client and establish a new connection when there is a new request.</p> <p>Value:</p> <ul style="list-style-type: none"> <li>For TCP listeners, the value ranges from <b>10</b> to <b>4000</b> (in seconds). The default value is <b>300</b>.</li> <li>For HTTP, HTTPS, and TERMINATED_HTTPS listeners, the value ranges from <b>0</b> to <b>4000</b> (in seconds). The default value is <b>60</b>.</li> <li>For UDP listeners, this parameter does not take effect.</li> </ul>	<p>Dedicated load balancers: v1.19.16-r30, v1.21.10-r10, v1.23.8-r10, v1.25.3-r10, or later</p> <p>Shared load balancers: v1.23.13-r0, v1.25.8-r0, v1.27.5-r0, v1.28.3-r0, or later</p>

For details, see [Configuring Timeout for a LoadBalancer Service](#).

## HTTP/2

**Table 7-30** Annotations of using HTTP/2

Parameter	Type	Description	Supported Cluster Version
kubernetes.io/elb.http2-enable	String	<p>Whether HTTP/2 is enabled. Request forwarding using HTTP/2 improves the access performance between your application and the load balancer. However, the load balancer still uses HTTP/1.x to forward requests to the backend server.</p> <p>Options:</p> <ul style="list-style-type: none"> <li>• <b>true</b>: enabled</li> <li>• <b>false</b>: disabled (default value)</li> </ul> <p>Note: <b>HTTP/2 can be enabled or disabled only when the listener uses HTTPS</b>. This parameter is invalid and defaults to <b>false</b> when the listener protocol is HTTP.</p>	v1.23.13-r0, v1.25.8-r0, v1.27.5-r0, v1.28.3-r0, or later

For details, see [Configuring HTTP/2 for a LoadBalancer Service](#).

## Parameters for Automatically Creating a Load Balancer

**Table 7-31** elb.autocreate data structure

Parameter	Mandatory	Type	Description
name	No	String	<p>Name of the automatically created load balancer.</p> <p>The value can contain 1 to 64 characters. Only letters, digits, underscores (_), hyphens (-), and periods (.) are allowed.</p> <p>Default: <b>cce-lb+service.UID</b></p>
type	No	String	<p>Network type of the load balancer.</p> <ul style="list-style-type: none"> <li>• <b>public</b>: public network load balancer</li> <li>• <b>inner</b>: private network load balancer</li> </ul> <p>Default: <b>inner</b></p>

Parameter	Mandatory	Type	Description
bandwidth_name	Yes for public network load balancers	String	Bandwidth name. The default value is <b>cce-bandwidth-*****</b> . The value can contain 1 to 64 characters. Only letters, digits, underscores (_), hyphens (-), and periods (.) are allowed.
bandwidth_chargemode	No	String	Bandwidth mode. <ul style="list-style-type: none"> <li>• <b>bandwidth</b>: billed by bandwidth</li> <li>• <b>traffic</b>: billed by traffic</li> </ul> Default: <b>bandwidth</b>
bandwidth_size	Yes for public network load balancers	Integer	Bandwidth size. The value ranges from 1 Mbit/s to 2000 Mbit/s by default. Configure this parameter based on the bandwidth range allowed in your region. The minimum increment for bandwidth adjustment varies depending on the bandwidth range. <ul style="list-style-type: none"> <li>• The minimum increment is 1 Mbit/s if the allowed bandwidth does not exceed 300 Mbit/s.</li> <li>• The minimum increment is 50 Mbit/s if the allowed bandwidth ranges from 300 Mbit/s to 1000 Mbit/s.</li> <li>• The minimum increment is 500 Mbit/s if the allowed bandwidth exceeds 1000 Mbit/s.</li> </ul>
bandwidth_sharetype	Yes for public network load balancers	String	Bandwidth sharing mode. <ul style="list-style-type: none"> <li>• <b>PER</b>: dedicated bandwidth</li> </ul>
eip_type	Yes for public network load balancers	String	EIP type. <ul style="list-style-type: none"> <li>• <b>5_bgp</b>: dynamic BGP</li> <li>• <b>5_sbgp</b>: static BGP</li> </ul> The specific type varies with regions. For details, see the EIP console.

Parameter	Mandatory	Type	Description
vip_subnet_cidr_id	No	String	Subnet where a load balancer is located. The subnet must belong to the VPC where the cluster resides.  If this parameter is not specified, the ELB load balancer and the cluster are in the same subnet.  This field can be specified only for clusters of v1.21 or later.
vip_address	No	String	Private IP address of the load balancer. Only IPv4 addresses are supported.  The IP address must be in the ELB CIDR block. If this parameter is not specified, an IP address will be automatically assigned from the ELB CIDR block.  This parameter is available only in clusters of v1.23.11-r0, v1.25.6-r0, v1.27.3-r0, or later versions.

**Table 7-32** elb.health-check-option data structure

Parameter	Mandatory	Type	Description
delay	No	String	Health check interval (s) Value range: 1 to 50. Default value: <b>5</b>
timeout	No	String	Health check timeout, in seconds. Value range: 1 to 50. Default value: <b>10</b>
max_retries	No	String	Maximum number of health check retries. Value range: 1 to 10. Default value: <b>3</b>
protocol	No	String	Health check protocol. Value options: TCP or HTTP
path	No	String	Health check URL. This parameter needs to be configured when the protocol is <b>HTTP</b> . Default value: / Value range: 1-80 characters



**Table 7-33** elb.health-check-options

Parameter	Mandatory	Type	Description
target_service_port	Yes	String	Port for health check specified by spec.ports. The value consists of the protocol and port number, for example, TCP:80.
monitor_port	No	String	Re-specified port for health check. If this parameter is not specified, the service port is used by default. <b>NOTE</b> Ensure that the port is in the listening state on the node where the pod is located. Otherwise, the health check result will be affected.
delay	No	String	Health check interval (s) Value range: 1 to 50. Default value: <b>5</b>
timeout	No	String	Health check timeout, in seconds. Value range: 1 to 50. Default value: <b>10</b>
max_retries	No	String	Maximum number of health check retries. Value range: 1 to 10. Default value: <b>3</b>
protocol	No	String	Health check protocol. Default value: protocol of the associated Service Value options: TCP, UDP, or HTTP
path	No	String	Health check URL. This parameter needs to be configured when the protocol is <b>HTTP</b> . Default value: / Value range: 1-80 characters

**Table 7-34** elb.session-affinity-option data structure

Parameter	Mandatory	Type	Description
persistenc e_timeout	Yes	String	Sticky session timeout, in minutes. This parameter is valid only when <b>elb.session-affinity-mode</b> is set to <b>SOURCE_IP</b> . Value range: 1 to 60. Default value: <b>60</b>

### 7.3.4.3 Configuring HTTP/HTTPS for a LoadBalancer Service

#### Notes and Constraints

- Only clusters of v1.19.16 or later support HTTP or HTTPS.

**Table 7-35** Scenarios where a load balancer supports HTTP or HTTPS

ELB Type	Application scenario	Whether to Support HTTP or HTTPS	Description
Dedicated load balancer	Interconnecting with an existing load balancer	Supported	<ul style="list-style-type: none"> <li>• For versions earlier than v1.19.16-r50, v1.21.11-r10, v1.23.9-r10, v1.25.4-r10 and v1.27.1-r10, the load balancer flavor must <b>support both the layer-4 and layer-7 routing</b>.</li> <li>• For v1.19.16-r50, v1.21.11-r10, v1.23.9-r10, v1.25.4-r10, v1.27.1-r10, and later versions, the load balancer flavor <b>must support layer-7 routing</b>.</li> </ul>
	Automatically creating a load balancer	Supported	<ul style="list-style-type: none"> <li>• For versions earlier than v1.19.16-r50, v1.21.11-r10, v1.23.9-r10, v1.25.4-r10 and v1.27.1-r10, the load balancer flavor must <b>support both the layer-4 and layer-7 routing</b>.</li> <li>• For v1.19.16-r50, v1.21.11-r10, v1.23.9-r10, v1.25.4-r10, v1.27.1-r10, and later versions, the load balancer flavor <b>must support layer-7 routing</b>.</li> </ul>

- Do not connect an ingress and a Service that uses HTTP or HTTPS to the same listener of the same load balancer. Otherwise, a port conflict occurs.

#### Using kubectl

If a Service uses the HTTP or HTTPS protocol, it is important to take note of the following configuration requirements:

- Different ELB types and cluster versions have different requirements on flavors. For details, see [Table 7-35](#).
- The two ports in **spec.ports** must correspond to those in **kubernetes.io/elb.protocol-port**. In this example, ports 443 and 80 are enabled with HTTPS and HTTP, respectively.

The following is a configuration example for automatically creating a dedicated load balancer, in which key configurations are marked in red:

```

apiVersion: v1
kind: Service
metadata:
  annotations:
# Specify the Layer 4 and Layer 7 flavors in the parameters for automatically creating a load balancer.
  kubernetes.io/elb.autocreate: '
    {
      "type": "public",
      "bandwidth_name": "cce-bandwidth-1634816602057",
      "bandwidth_chargemode": "bandwidth",
      "bandwidth_size": 5,
      "bandwidth_sharetype": "PER",
      "eip_type": "5_bgp",
      "available_zone": [
        ""
      ],
      "l7_flavor_name": "L7_flavor.elb.s2.small",
      "l4_flavor_name": "L4_flavor.elb.s1.medium"
    }
  kubernetes.io/elb.class: performance # Dedicated load balancer
  kubernetes.io/elb.protocol-port: "https:443,http:80" # HTTP/HTTPS and port number, which must be the
same as the port numbers in spec.ports
  kubernetes.io/elb.cert-id: "17e3b4f4bc40471c86741dc3aa211379" # Certificate ID of the LoadBalancer
Service
labels:
  app: nginx
  name: test
  name: test
  namespace: default
spec:
  ports:
  - name: cce-service-0
    port: 443
    protocol: TCP
    targetPort: 80
  - name: cce-service-1
    port: 80
    protocol: TCP
    targetPort: 80
  selector:
    app: nginx
    version: v1
  sessionAffinity: None
  type: LoadBalancer

```

**Table 7-36** Key parameters

Parameter	Type	Description
kubernetes.io/elb.protocol-port	String	<p>If a Service is TLS/HTTP/HTTPS-compliant, configure the protocol and port number in the format of "protocol:port".</p> <p>Specifically:</p> <ul style="list-style-type: none"> <li>• <b>protocol</b>: specifies the protocol used by the listener port. The value can be <b>tls</b>, <b>http</b>, or <b>https</b>.</li> <li>• <b>port</b>: Service port specified by <b>spec.ports[].port</b>.</li> </ul> <p>In this example, ports 443 and 80 are enabled with HTTPS and HTTP, respectively. Therefore, the parameter value is <i>https:443,http:80</i>.</p>

Parameter	Type	Description
kubernetes.io/ elb.cert-id	String	ID of an ELB certificate, which is used as the TLS/HTTPS server certificate.  To obtain the certificate, log in to the CCE console, choose <b>Service List &gt; Networking &gt; Elastic Load Balance</b> , and click <b>Certificates</b> in the navigation pane. In the load balancer list, copy the ID under the target certificate name.

### 7.3.4.4 Configuring SNI for a LoadBalancer Service

An SNI certificate is an extended server certificate that allows the same IP address and port number to provide multiple access domain names for external systems. Different security certificates can be used based on the domain names requested by clients to ensure HTTPS communication security.

When configuring SNI, you need to add a certificate associated with a domain name. The client submits the requested domain name information when initiating an SSL handshake request. After receiving the SSL request, the load balancer searches for the certificate based on the domain name. If the certificate is found, the load balancer will return it to the client. If the certificate is not found, the load balancer will return the default server certificate.

#### NOTE

After SNI is configured, if you delete the SNI configuration on the CCE console or delete the target annotation from the YAML file, the configuration on the ELB will be retained.

### Prerequisites

- A Kubernetes cluster is available and the cluster version meets the following requirements:
  - v1.23: v1.23.13-r0 or later
  - v1.25: v1.25.8-r0 or later
  - v1.27: v1.27.5-r0 or later
  - v1.28: v1.28.3-r0 or later
  - Other clusters of later versions
- You have created one or more SNI certificates in ELB and specified a domain name in these certificates.
- To create a cluster using commands, ensure kubectl is used. For details, see [Connecting to a Cluster Using kubectl](#).

### Using the CCE Console

**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 **Services & Ingresses**. In the upper right corner, click **Create Service**.

**Step 3** Configure Service parameters. In this example, only mandatory parameters required for using SNI are listed. For details about how to configure other parameters, see [Using the Console](#).

- **Service Name:** Specify a Service name, which can be the same as the workload name.
- **Service Type:** Select **LoadBalancer**.
- **Selector:** Add a label and click **Confirm**. The Service will use this label to select pods. You can also click **Reference Workload Label** to use the label of an existing workload. In the dialog box that is displayed, select a workload and click **OK**.
- **Load Balancer:** Select a load balancer type and creation mode.
  - A load balancer can be dedicated or shared. To enable HTTP/HTTPS on the listener port of a dedicated load balancer, the type of the load balancer must be **Application (HTTP/HTTPS)** or **Network (TCP/UDP) & Application (HTTP/HTTPS)**.
  - This section uses an existing load balancer as an example. For details about the parameters for automatically creating a load balancer, see [Table 7-14](#).
- **Ports**
  - **Protocol:** Select **TCP**. If you select **UDP**, HTTP and HTTPS will be unavailable.
  - **Service Port:** port used by the Service. The port number ranges from 1 to 65535.
  - **Container Port:** listener port of the workload. For example, Nginx uses port 80 by default.
  - **Frontend Protocol:** In this example, HTTPS must be enabled for the Service to use SNI. For a [dedicated load balancer](#), to use HTTP/HTTPS, the type of the load balancer must be **Application (HTTP/HTTPS)**.
- **Listener**
  - **Server Certificate:** Select a server certificate as the default certificate.
  - **SNI:** Add an SNI certificate containing a domain name.  
If an SNI certificate cannot be found based on the domain name requested by the client, the server certificate will be returned by default.

**Step 4** Click **OK**.

----End

## Using kubectl

This section uses an existing load balancer as an example. An example YAML file of a SNI-compliant Service is as follows:

```
apiVersion: v1
kind: Service
metadata:
  name: test
  labels:
    app: test
  namespace: default
annotations:
  kubernetes.io/elb.class: performance # Load balancer type
```

```
kubernetes.io/elb.id: 65318265-4f01-4541-a654-fa74e439dfd3 # ID of an existing load balancer
kubernetes.io/elb.protocol-port: https:80 # Port where SNI is to be enabled
kubernetes.io/elb.cert-id: b64ab636f1614e1a960b5249c497a880 # HTTPS server certificate
kubernetes.io/elb.tls-certificate-ids:
5196aa70b0f143189e4cb54991ba2286,8125d71fcc124aabb007610cba42d60 # SNI certificate IDs
kubernetes.io/elb.lb-algorithm: ROUND_ROBIN
spec:
  selector:
    app: test
  externalTrafficPolicy: Cluster
  ports:
    - name: cce-service-0
      targetPort: 80
      nodePort: 0
      port: 80
      protocol: TCP
  type: LoadBalancer
  loadBalancerIP: **.**.**. # Private IP address of the load balancer
```

**Table 7-37** Key parameters

Parameter	Type	Description
kubernetes.io/elb.protocol-port	String	<p>If a Service is HTTP/HTTPS-compliant, configure the protocol and port number in the format of "protocol:port".</p> <p>where,</p> <ul style="list-style-type: none"> <li><b>protocol</b>: specifies the protocol used by the listener port. The value can be <b>http</b> or <b>https</b>.</li> <li><b>ports</b>: Service ports specified by <b>spec.ports[].port</b>.</li> </ul> <p>For example, to use SNI, the Service protocol must be <b>https</b> and the Service port must be <b>80</b>. Therefore, the parameter value is <b>https:80</b>.</p>
kubernetes.io/elb.cert-id	String	<p>ID of an ELB certificate, which is used as the HTTPS server certificate.</p> <p>To obtain the certificate, log in to the CCE console, choose <b>Service List &gt; Networking &gt; Elastic Load Balance</b>, and click <b>Certificates</b> in the navigation pane. In the load balancer list, copy the ID under the target certificate name.</p>

Parameter	Type	Description
kubernetes.io/elb.tls-certificate-ids	String	<p>In ELB, the IDs of SNI certificates that must contain a domain name are separated by commas (,).</p> <p>If an SNI certificate cannot be found based on the domain name requested by the client, the server certificate will be returned by default.</p> <p>To obtain the certificate, log in to the CCE console, choose <b>Service List &gt; Networking &gt; Elastic Load Balance</b>, and click <b>Certificates</b> in the navigation pane. In the load balancer list, copy the ID under the target certificate name.</p>

### 7.3.4.5 Configuring HTTP/2 for a LoadBalancer Service

Services can be exposed via HTTP/2. By default, HTTP/1.x is used between clients and load balancers. HTTP/2 can improve access performance between clients and load balancers, but HTTP/1.x is still used between load balancers and backend servers.

#### NOTE

- An HTTPS-compliant load balancer supports HTTP/2.
- After HTTP/2 is configured, if you delete the advanced configuration for enabling HTTP/2 on the CCE console or delete the target annotation from the YAML file, the configuration on the ELB will be retained.

### Prerequisites

- A Kubernetes cluster is available and the cluster version meets the following requirements:
  - v1.23: v1.23.13-r0 or later
  - v1.25: v1.25.8-r0 or later
  - v1.27: v1.27.5-r0 or later
  - v1.28: v1.28.3-r0 or later
  - Other clusters of later versions
- To create a cluster using commands, ensure kubectl is used. For details, see [Connecting to a Cluster Using kubectl](#).

### Using the CCE Console

**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 **Services & Ingresses**. In the upper right corner, click **Create Service**.

**Step 3** Configure Service parameters. In this example, only mandatory parameters are listed. For details about how to configure other parameters, see [Using the Console](#).

- **Service Name:** Specify a Service name, which can be the same as the workload name.
- **Service Type:** Select **LoadBalancer**.
- **Selector:** Add a label and click **Confirm**. The Service will use this label to select pods. You can also click **Reference Workload Label** to use the label of an existing workload. In the dialog box that is displayed, select a workload and click **OK**.
- **Load Balancer:** Select a load balancer type and creation mode.
  - A load balancer can be dedicated or shared. To enable HTTP/HTTPS on the listener port of a dedicated load balancer, the type of the load balancer must be **Application (HTTP/HTTPS)** or **Network (TCP/UDP) & Application (HTTP/HTTPS)**.
  - This section uses an existing load balancer as an example. For details about the parameters for automatically creating a load balancer, see [Table 7-14](#).
- **Ports**
  - **Protocol:** Select **TCP**. If you select **UDP**, HTTP and HTTPS will be unavailable.
  - **Service Port:** port used by the Service. The port number ranges from 1 to 65535.
  - **Container Port:** listener port of the workload. For example, Nginx uses port 80 by default.
  - **Frontend Protocol:** In this example, HTTPS must be enabled for the Service to use HTTP/2. For a [dedicated load balancer](#), to use HTTP/HTTPS, the type of the load balancer must be **Application (HTTP/HTTPS)**.
- **Listener**
  - SSL authentication is available only in clusters of v1.23.14-r0, v1.25.9-r0, v1.27.6-r0, v1.28.4-r0, or later versions.
    - **One-way authentication:** Only the backend server is authenticated. If you also need to authenticate the identity of the client, select mutual authentication.
    - **Mutual authentication:** If you want the clients and the load balancer to authenticate each other, select this option. Only authenticated clients will be allowed to access the load balancer.
  - **CA Certificate:** If **SSL Authentication** is set to **Mutual authentication**, add a CA certificate to authenticate the client. A CA certificate is issued by a certificate authority (CA) and used to verify the certificate issuer. If HTTPS mutual authentication is required, HTTPS connections can be established only when the client provides a certificate issued by a specific CA.
  - **Server Certificate:** Select a server certificate when HTTPS is used.
  - **SNI:** Add an SNI certificate containing a domain name.



- **Advanced Options:** Click **Add custom container network settings** and enable HTTP/2.

**Step 4** Click **OK**.

----End

## Using kubectl

To enable HTTP/2, add the following annotation:

```
kubernetes.io/elb.http2-enable: 'true'
```

The following shows an example YAML file where an existing load balancer is associated:

```
apiVersion: v1
kind: Service
metadata:
  name: test
  labels:
    app: test
    version: v1
  namespace: default
  annotations:
    kubernetes.io/elb.class: performance
    kubernetes.io/elb.id: 35cb350b-23e6-4551-ac77-10d5298f5204
    kubernetes.io/elb.protocol-port: https:443
    kubernetes.io/elb.cert-id: b64ab636f1614e1a960b5249c497a880
    kubernetes.io/elb.http2-enable: 'true'
    kubernetes.io/elb.lb.algorithm: ROUND_ROBIN
spec:
  selector:
    app: test
    version: v1
  externalTrafficPolicy: Cluster
  ports:
    - name: cce-service-0
      targetPort: 80
      nodePort: 0
      port: 443
      protocol: TCP
  type: LoadBalancer
  loadBalancerIP: **.*.*.*
```

**Table 7-38** HTTP/2 parameters

Parameter	Type	Description
kubernetes.io/elb.protocol-port	String	<p>If a Service is HTTP/HTTPS-compliant, configure the protocol and port number in the format of "protocol:port".</p> <p>where,</p> <ul style="list-style-type: none"> <li>• <b>protocol:</b> specifies the protocol used by the listener port. The value can be <b>http</b> or <b>https</b>.</li> <li>• <b>ports:</b> Service ports specified by <b>spec.ports[].port</b>.</li> </ul> <p>For example, to use HTTPS, the Service port must be <b>443</b>. Therefore, the parameter value is <b>https:443</b>.</p>

Parameter	Type	Description
kubernetes.io/elb.cert-id	String	ID of an ELB certificate, which is used as the HTTPS server certificate.  To obtain the certificate, log in to the CCE console, choose <b>Service List &gt; Networking &gt; Elastic Load Balance</b> , and click <b>Certificates</b> in the navigation pane. In the load balancer list, copy the ID under the target certificate name.
kubernetes.io/elb.http2-enable	String	Whether HTTP/2 is enabled. Request forwarding using HTTP/2 improves the access performance between your application and the load balancer. However, the load balancer still uses HTTP/1.x to forward requests to the backend server.  Options: <ul style="list-style-type: none"> <li>• <b>true</b>: enabled</li> <li>• <b>false</b>: disabled (default value)</li> </ul> Note: <b>HTTP/2 can be enabled or disabled only when the listener uses HTTPS</b> . This parameter is invalid and defaults to <b>false</b> when the listener protocol is HTTP.

### 7.3.4.6 Configuring Timeout for a LoadBalancer Service

LoadBalancer Services allow you to configure timeout, which is the maximum duration for keeping a connection if no request is received from the client. If no request is received during this period, the load balancer closes the connection and establishes a new one with the client when the next request arrives.

 **NOTE**

After timeout is configured, if you delete the timeout configuration on the CCE console or delete the target annotation from the YAML file, the configuration on the ELB will be retained.

### Notes and Constraints

The following table lists the scenarios where timeout can be configured for a Service.

Timeout Type	Load Balancer Type	Restrictions	Supported Cluster Version
Idle timeout	Dedicated	None	<ul style="list-style-type: none"> <li>• v1.19: v1.19.16-r30 or later</li> <li>• v1.21: v1.21.10-r10 or later</li> <li>• v1.23: v1.23.8-r10 or later</li> <li>• v1.25: v1.25.3-r10 or later</li> <li>• Other clusters of later versions</li> </ul>

## Using kubectl

Use annotations to configure timeout. The following shows an example:

```

apiVersion: v1
kind: Service
metadata:
  annotations:
    kubernetes.io/elb.id: <your_elb_id> # In this example, an existing dedicated load balancer is used.
    Replace its ID with the ID of your dedicated load balancer.
    kubernetes.io/elb.class: performance # Load balancer type
    kubernetes.io/elb.protocol-port: http:80 # The HTTP port 80 is used.
    kubernetes.io/elb.keepalive_timeout: '300' # Timeout setting for client connections

name: nginx
spec:
  ports:
  - name: service0
    port: 80
    protocol: TCP
    targetPort: 80
  selector:
    app: nginx
  type: LoadBalancer
  
```

**Table 7-39** Key annotation parameters

Parameter	Mandatory	Type	Description
kubernetes.io/elb.keepalive_timeout	No	String	<p>Timeout for client connections. If there are no requests reaching the load balancer during the timeout duration, the load balancer will disconnect the connection from the client and establish a new connection when there is a new request.</p> <p>Value:</p> <ul style="list-style-type: none"> <li>For TCP listeners, the value ranges from <b>10</b> to <b>4000</b> (in seconds). The default value is <b>300</b>.</li> <li>For HTTP, HTTPS, and TERMINATED_HTTPS listeners, the value ranges from <b>0</b> to <b>4000</b> (in seconds). The default value is <b>60</b>.</li> <li>For UDP listeners, the value ranges from <b>10</b> to <b>4000</b> (in seconds). The default value is <b>300</b>.</li> </ul>

### 7.3.4.7 Configuring Health Check on Multiple Ports of a LoadBalancer Service

The annotation field related to the health check of the LoadBalancer Service is upgraded from **kubernetes.io/elb.health-check-option** to **kubernetes.io/elb.health-check-options**. Each Service port can be configured separately, and you can configure only some ports. If the port protocol does not need to be configured separately, the original annotation field is still available and does not need to be modified.

#### Notes and Constraints

- This feature is available in the following versions:
  - v1.19: v1.19.16-r5 or later
  - v1.21: v1.21.8-r0 or later
  - v1.23: v1.23.6-r0 or later
  - v1.25: v1.25.2-r0 or later
  - Versions later than v1.25
- Either **kubernetes.io/elb.health-check-option** or **kubernetes.io/elb.health-check-options** can be configured.
- The **target\_service\_port** field is mandatory and must be unique.
- For a TCP port, the health check protocol can only be TCP or HTTP. For a UDP port, the health check protocol must be UDP.

## Procedure

The following is an example of using the `kubernetes.io/elb.health-check-options` annotation:

```
apiVersion: v1
kind: Service
metadata:
  name: nginx
  namespace: default
  labels:
    app: nginx
    version: v1
  annotations:
    kubernetes.io/elb.class: union # Load balancer type
    kubernetes.io/elb.id: <your_elb_id> # Load balancer ID. Replace it with the actual value.
    kubernetes.io/elb.lb-algorithm: ROUND_ROBIN # Load balancer algorithm
    kubernetes.io/elb.health-check-flag: 'on' # Enable ELB health check.
    kubernetes.io/elb.health-check-options: '[
  {
    "protocol": "TCP",
    "delay": "5",
    "timeout": "10",
    "max_retries": "3",
    "target_service_port": "TCP:1",
    "monitor_port": "22"
  },
  {
    "protocol": "HTTP",
    "delay": "5",
    "timeout": "10",
    "max_retries": "3",
    "path": "/",
    "target_service_port": "TCP:2",
    "monitor_port": "22",
    "expected_codes": "200-399,401,404"
  }
]'
spec:
  selector:
    app: nginx
    version: v1
  externalTrafficPolicy: Cluster
  ports:
    - name: cce-service-0
      targetPort: 1
      nodePort: 0
      port: 1
      protocol: TCP
    - name: cce-service-1
      targetPort: 2
      nodePort: 0
      port: 2
      protocol: TCP
  type: LoadBalancer
  loadBalancerIP: *.*.*.*
```

**Table 7-40** elb.health-check-options

Parameter	Mandatory	Type	Description
target_service_port	Yes	String	Port for health check specified by spec.ports. The value consists of the protocol and port number, for example, TCP:80.

Parameter	Mandatory	Type	Description
monitor_port	No	String	Re-specified port for health check. If this parameter is not specified, the service port is used by default. <b>NOTE</b> Ensure that the port is in the listening state on the node where the pod is located. Otherwise, the health check result will be affected.
delay	No	String	Health check interval (s) Value range: 1 to 50. Default value: <b>5</b>
timeout	No	String	Health check timeout, in seconds. Value range: 1 to 50. Default value: <b>10</b>
max_retries	No	String	Maximum number of health check retries. Value range: 1 to 10. Default value: <b>3</b>
protocol	No	String	Health check protocol. Default value: protocol of the associated Service Value options: TCP, UDP, or HTTP
path	No	String	Health check URL. This parameter needs to be configured when the protocol is <b>HTTP</b> . Default value: / Value range: 1-80 characters

### 7.3.4.8 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.

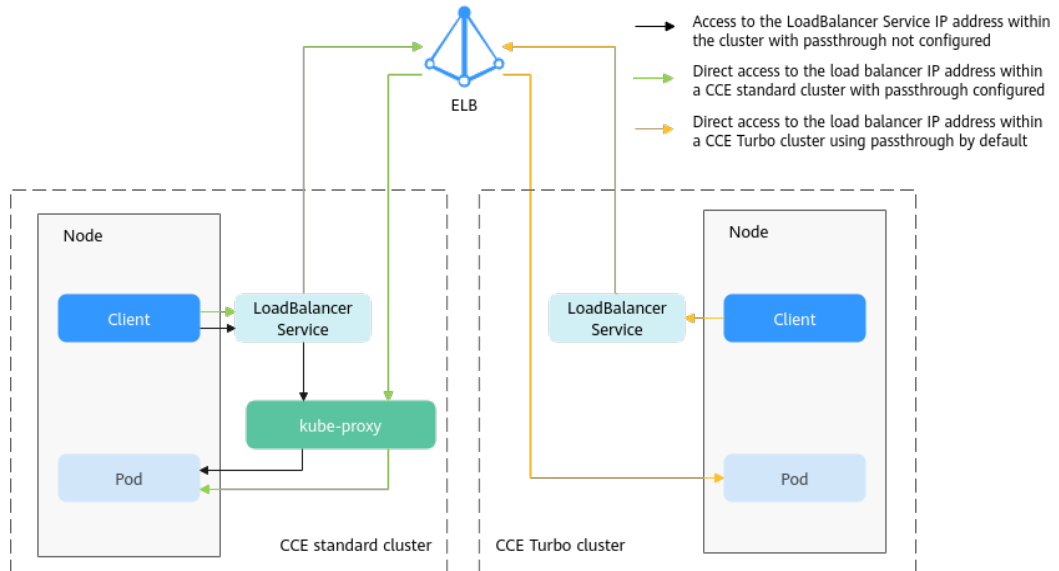
If node-level affinity is configured for a Service (with **externalTrafficPolicy** set to **Local**), the Service will forward traffic only to pods on the node that run these pods. When a node or pod accesses another pod in the same cluster, if the node where the client runs does not have the corresponding 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 7-31** 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.

## Constraints

- 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**.

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":"cce-bandwidth","bandwidth_chargemode":"bandwidth","bandwidth_size":5,"bandwidth_sharetype":"PER","eip_type":"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

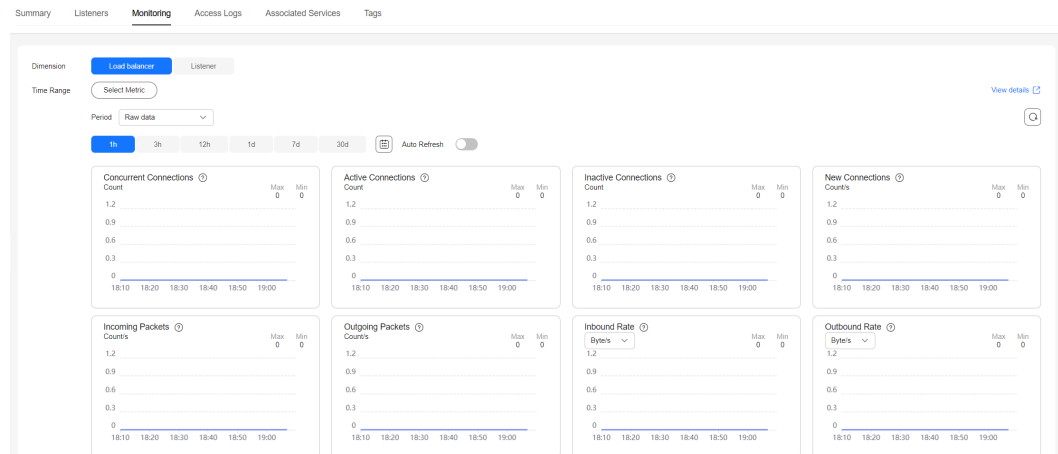
## Verification

**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.





**Step 3** Log in to an Nginx container in the cluster using kubectl and access the IP address of the load balancer.

1. 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 container.  
`kubectl exec -it nginx-7c4c5cc6b5-vpncx -- /bin/sh`
3. Access the load balancer IP address.  
`curl **.*.*.*`

**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

### 7.3.4.9 Setting the Pod Ready Status Through the ELB Health Check

The ready status of the pod is associated with the ELB health check. After the health check is successful, the pod is ready. This association works with the **strategy.rollingUpdate.maxSurge** and **strategy.rollingUpdate.maxUnavailable** parameters of the pod to implement graceful rolling upgrade.

### Notes and Constraints

- This feature is available in the following versions:
  - v1.19: v1.19.16-r5 or later
  - v1.21: v1.21.8-r0 or later
  - v1.23: v1.23.6-r0 or later
  - v1.25: v1.25.2-r0 or later
  - Versions later than v1.25
- This function applies only to passthrough scenarios, that is, scenarios where dedicated load balancers are used in CCE Turbo clusters.

- To use this function, configure the `readinessGates` field in the pod and specify the label `target-health.elb.k8s.cce/{serviceName}`, where `{serviceName}` indicates the service name.
- The pod ready status takes effect only when the ELB backend is initially connected. The subsequent health check status does not affect the pod ready status.

## Setting the Pod Ready Status Through the ELB Health Check

To use pod readiness gates, perform the following steps:

- 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 **Workloads**. In the upper right corner, click **Create from YAML**.

YAML content:

```
kind: Deployment
apiVersion: apps/v1
metadata:
  name: nginx
  namespace: default
  labels:
    version: v1
spec:
  replicas: 1
  selector:
    matchLabels:
      app: nginx
      version: v1
  template:
    metadata:
      labels:
        app: nginx
        version: v1
    spec:
      containers:
        - name: container-1
          image: nginx:latest
          imagePullSecrets:
            - name: default-secret
          readinessGates:
            - conditionType: target-health.elb.k8s.cce/specific-service-name # Specifies the ServiceName.
      strategy:
        type: RollingUpdate
        rollingUpdate:
          maxUnavailable: 25% # Works with the following two parameters to control the number of ELB
          maxSurge: 25%
backends and implement graceful rolling upgrade.
```

- Step 3** Click **OK**. On the workload list, you can check the workload status and find that the pod is not ready.

An event is displayed as follows:

```
Pod not ready: corresponding condition of pod readinessgate "target-health.elb.k8s.cce/specific-service-name" does not exist.
```

- Step 4** In the navigation pane, choose **Services & Ingresses**. In the upper right corner, click **Create Service** and configure the following parameters:
  - **Service Name:** The value must be the same as the value of `readinessGates` in the pod.

- **Service Type:** Select **LoadBalancer**.
- **Selector:** Click **Reference Workload Label**, select the workload created in the previous step, and click **OK**.
- **Load Balancer:** Dedicated load balancers must be used. You can select an existing load balancer or automatically create a load balancer.
- **Health Check:** Whether to enable health check. (If it is not configured, the health check is enabled by default.)

**Figure 7-32** Configuring a load balancer

**Step 5** Go to the ELB console and check the backend server group. The health check status is normal.

**Step 6** On the CCE console, the workload is in the **Running** status.

----End

### 7.3.5 Headless Services

Services allow internal and external pod access, but not the following scenarios:

- Accessing all pods at the same time
- Pods in a Service accessing each other

This is where headless Service come into service. A headless Service does not create a cluster IP address, and the DNS records of all pods are returned during query. In this way, the IP addresses of all pods can be queried. **StatefulSets** use headless Services to support mutual access between pods.

```
apiVersion: v1
kind: Service # Object type (Service)
metadata:
```

```
name: nginx-headless
labels:
  app: nginx
spec:
  ports:
    - name: nginx # - 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 that a headless Service is to be created.
```

Run the following command to create a headless Service:

```
# kubectl create -f headless.yaml
service/nginx-headless created
```

After the Service is created, you can query the Service.

```
# kubectl get svc
NAME          TYPE          CLUSTER-IP  EXTERNAL-IP  PORT(S)  AGE
nginx-headless ClusterIP  None         <none>       80/TCP   5s
```

Create a pod to query the DNS. You can view the records of all pods. In this way, all pods can be accessed.

```
$ kubectl run -i --tty --image tutum/dnsutils dnsutils --restart=Never --rm /bin/sh
If you do not see a command prompt, try pressing Enter.
/# nslookup nginx-0.nginx
Server:      10.247.3.10
Address:    10.247.3.10#53
Name:   nginx-0.nginx.default.svc.cluster.local
Address: 172.16.0.31

/# nslookup nginx-1.nginx
Server:      10.247.3.10
Address:    10.247.3.10#53
Name:   nginx-1.nginx.default.svc.cluster.local
Address: 172.16.0.18

/# nslookup nginx-2.nginx
Server:      10.247.3.10
Address:    10.247.3.10#53
Name:   nginx-2.nginx.default.svc.cluster.local
Address: 172.16.0.19
```

## 7.4 Ingresses

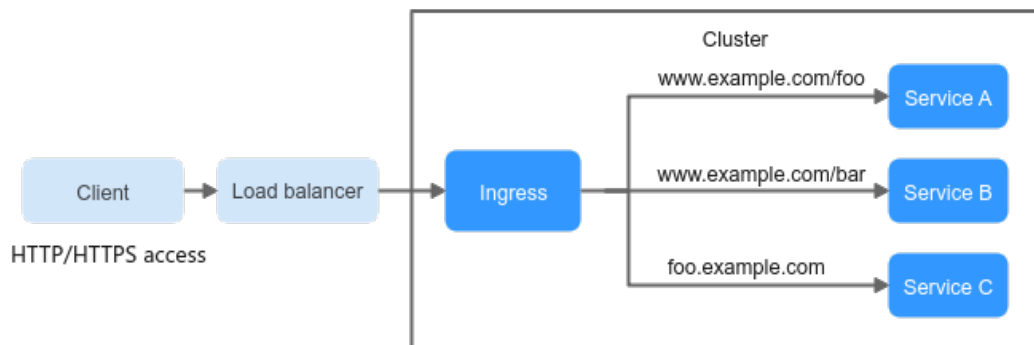
### 7.4.1 Overview

#### Why We Need Ingresses

A Service is generally used to forward access requests based on TCP and UDP and provide layer-4 load balancing for clusters. However, in actual scenarios, if there is a large number of HTTP/HTTPS access requests on the application layer, the Service cannot meet the forwarding requirements. Therefore, the Kubernetes cluster provides an HTTP-based access mode, ingress.

An ingress is an independent resource in the Kubernetes cluster and defines rules for forwarding external access traffic. As shown in [Figure 7-33](#), you can customize forwarding rules based on domain names and URLs to implement fine-grained distribution of access traffic.

**Figure 7-33** Ingress diagram



## Ingress Overview

Kubernetes uses ingress resources to define how incoming traffic should be handled, while the Ingress Controller is responsible for processing the actual traffic.

- **Ingress object:** a set of access rules that forward requests to specified Services based on domain names or paths. It can be added, deleted, modified, and queried by calling APIs.
- **Ingress Controller:** an executor for forwarding requests. It monitors the changes of resource objects such as ingresses, Services, endpoints, secrets (mainly TLS certificates and keys), nodes, and ConfigMaps in real time, parses rules defined by ingresses, and forwards requests to the target backend Services.

The way of implementing Ingress Controllers varies depending on their vendors. CCE supports LoadBalancer Ingress Controllers and NGINX Ingress Controllers.

- LoadBalancer Ingress Controllers are deployed on master nodes and they forward traffic based on the ELB. All policy configurations and forwarding behaviors are handled by the ELB.
- NGINX Ingress Controllers are deployed in clusters using charts and images maintained by the Kubernetes community. They provide external access through NodePort and forward external traffic to other services in the cluster through Nginx. All traffic forwarding behaviors and forwarding objects are within the cluster.

## Ingress Feature Comparison

**Table 7-41** Comparison between ingress features

Feature	ELB Ingress Controller	Nginx Ingress Controller
O&M	O&M-free	Self-installation, upgrade, and maintenance
Performance	One ingress supports only one load balancer.	Multiple ingresses support one load balancer.

Feature	ELB Ingress Controller	Nginx Ingress Controller
	Enterprise-grade load balancers are used to provide high performance and high availability. Service forwarding is not affected in upgrade and failure scenarios.	Performance varies depending on the resource configuration of pods.
	Dynamic loading is supported.	<ul style="list-style-type: none"> <li>Processes must be reloaded for non-backend endpoint changes. This causes loss to persistent connections.</li> <li>Lua supports hot updates of endpoint changes.</li> <li>Processes must be reloaded for a Lua modification.</li> </ul>
Component deployment	Deployed on the master node	Deployed on worker nodes, and operations costs required for the Nginx component
Route redirection	Supported	Supported
SSL configuration	Supported	Supported
Using ingress as a proxy for backend services	Supported	Supported, which can be implemented through backend-protocol: HTTPS annotations.

The LoadBalancer ingress is essentially different from the open source Nginx Ingress. Therefore, their supported Service types are different. For details, see [Services Supported by LoadBalancer Ingresses](#).

LoadBalancer Ingress Controllers are deployed on master nodes. All policy configurations and forwarding behaviors are configured on the ELB. Load balancers outside the cluster can connect to nodes in the cluster only through the IP address of the VPC. Therefore, LoadBalancer ingresses support only NodePort Services.

NGINX Ingress Controller runs in a cluster and is exposed as a Service through NodePort. Traffic is forwarded to other Services in the cluster through Nginx-ingress. The traffic forwarding behavior and forwarding object are in the cluster. Therefore, both ClusterIP and NodePort Services are supported.

In conclusion, LoadBalancer ingresses use enterprise-grade load balancers to forward traffic and delivers high performance and stability. NGINX Ingress Controller is deployed on cluster nodes, which consumes cluster resources but has better configurability.

## Working Rules of LoadBalancer Ingress Controller

LoadBalancer Ingress Controller developed by CCE implements layer-7 network access for the internet and intranet (in the same VPC) based on ELB and distributes access traffic to the target Services using different paths.

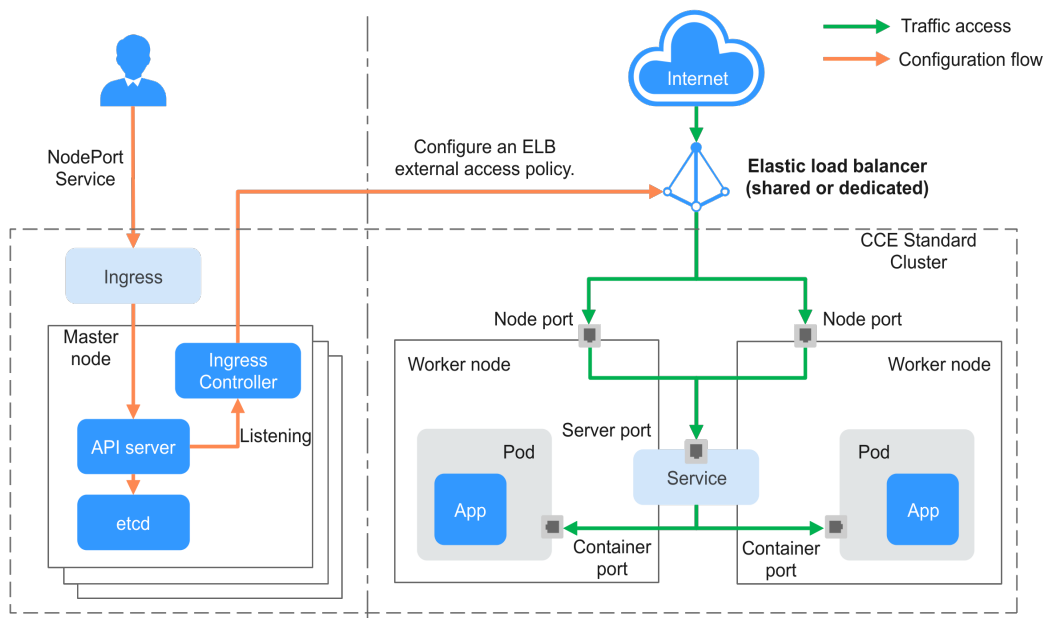
LoadBalancer Ingress Controller is deployed on the master node and bound to the load balancer in the VPC where the cluster resides. Different domain names, ports, and forwarding policies can be configured for the same load balancer (with the same IP address). The working rules of LoadBalancer Ingress Controller are as follows:

1. A user creates an ingress and configures a traffic access rule in the ingress, including the load balancer, access path, SSL, and backend Service port.
2. When Ingress Controller detects that the ingress changes, it reconfigures the listener and backend server route on the ELB according to the traffic access rule.
3. When a user attempts to access a workload, the ELB forwards the traffic to the target workload according to the configured forwarding rule.

The way LoadBalancer Ingress Controller works depends on the type of cluster and ELB being used. The following section describes the configuration process and network flow in various scenarios.

## CCE Standard Clusters

**Figure 7-34** Working flow of a LoadBalancer ingress in a CCE standard cluster



## Services Supported by Ingresses

LoadBalancer and the open-source Nginx ingresses support different Services due to their implementation principles.

## Services Supported by LoadBalancer Ingresses

**Table 7-42** Services supported by LoadBalancer ingresses

Cluster Type	ELB Type	ClusterIP	NodePort
CCE standard cluster	Shared load balancer	Not supported	Supported
	Dedicated load balancer	Not supported (Failed to access the dedicated load balancers because no ENI is bound to the associated pod of the ClusterIP Service.)	Supported
CCE Turbo cluster	Shared load balancer	Not supported	Supported
	Dedicated load balancer	Supported	Not supported (Failed to access the dedicated load balancers because an ENI has been bound to the associated pod of the NodePort Service.)

## 7.4.2 LoadBalancer Ingresses

### 7.4.2.1 Creating a LoadBalancer Ingress on the Console

In Kubernetes, an ingress is a resource object that controls how Services within a cluster can be accessed from outside the cluster. You can use ingresses to configure different forwarding rules to access pods in a cluster. The following uses an [Nginx workload](#) as an example to describe how to create a LoadBalancer ingress on the console.

#### Prerequisites

- An available workload has been deployed in the cluster for external access. If no workload is available, deploy a workload by referring to [Creating a Deployment](#), [Creating a StatefulSet](#), or [Creating a DaemonSet](#).
- A Service for external access has been configured for the workload. [Services Supported by LoadBalancer Ingresses](#) lists the Service types supported by LoadBalancer ingresses.

#### Notes and Constraints

- It is recommended that other resources not use the load balancer automatically created by an ingress. Otherwise, the load balancer will be occupied when the ingress is deleted, resulting in residual resources.



- After an ingress is created, upgrade and maintain the configuration of the selected load balancers on the CCE console. Do not modify the configuration on the ELB console. Otherwise, the ingress service may be abnormal.
- The URL registered in an ingress forwarding policy must be the same as the URL used to access the backend Service. Otherwise, a 404 error will be returned.
- In a cluster using the IPVS proxy mode, if the ingress and Service use the same ELB load balancer, the ingress cannot be accessed from the nodes and containers in the cluster because kube-proxy mounts the LoadBalancer Service address to the ipvs-0 bridge. This bridge intercepts the traffic of the load balancer connected to the ingress. Use different load balancers for the ingress and Service.
- If multiple ingresses access the same ELB port in a cluster, the listener configuration items (such as the certificate associated with the listener and the HTTP/2 attribute of the listener) are subject to the configuration of the first ingress.

## Adding a LoadBalancer Ingress

This section uses an Nginx workload as an example to describe how to add a LoadBalancer ingress.

- Step 1** Log in to the CCE console and click the cluster name to access the cluster console.
- Step 2** Choose **Services & Ingresses** in the navigation pane, click the **Ingresses** tab, and click **Create Ingress** in the upper right corner.
- Step 3** Configure ingress parameters.

- **Name:** Customize the name of an ingress, for example, **ingress-demo**.
- **Load Balancer:** Select a load balancer type and creation mode.

A load balancer can be dedicated or shared. A dedicated load balancer must be of the application type (HTTP/HTTPS) and support private networks.

You can select **Use existing** or **Auto create** to obtain a load balancer. For details about the configuration of different creation modes, see [Table 7-43](#).

**Table 7-43** Load balancer configurations

How to Create	Configuration
Use existing	Only the load balancers in the same VPC as the cluster can be selected. If no load balancer is available, click <b>Create Load Balancer</b> to create one on the ELB console.

How to Create	Configuration
Auto create	<ul style="list-style-type: none"> <li>- <b>Instance Name:</b> Enter a load balancer name.</li> <li>- <b>EIP:</b> If you select <b>Auto create</b>, you can configure the size of the public network bandwidth.</li> <li>- <b>Resource Tag:</b> You can add resource tags to classify resources. You can create <b>predefined tags</b> on the TMS console. The predefined tags are available to all resources that support tags. You can use these tags to improve the tag creation and resource migration efficiency.</li> </ul>

- **Listener:** An ingress configures a listener for the load balancer, which listens to requests from the load balancer and distributes traffic. After the configuration is complete, a listener is created on the load balancer. The default listener name is *k8s\_<Protocol type>\_<Port number>*, for example, *k8s\_HTTP\_80*.
  - **External Protocol:** HTTP and HTTPS are available.
  - **External Port:** port number that is open to the ELB service address. The port number is configurable.
  - **Certificate Source:** TLS secret and ELB server certificates are supported.
    - **TLS secret:** For details about how to create a secret certificate, see [Creating a Secret](#).
    - **ELB server certificate:** Use a certificate created in the ELB service.
  - **Server Certificate:** When an HTTPS listener is created for a load balancer, bind a certificate to the load balancer to support encrypted authentication for HTTPS data transmission.

 NOTE

If there is already an HTTPS ingress for the chosen port on the load balancer, the certificate of the new HTTPS ingress must be the same as the certificate of the existing ingress. This means that a listener has only one certificate. If two certificates, each with a different ingress, are added to the same listener of the same load balancer, only the certificate added earliest takes effect on the load balancer.

- **SNI:** stands for Server Name Indication (SNI), which is an extended protocol of TLS. SNI allows multiple TLS-compliant domain names for external access using the same IP address and port number, and different domain names can use different security certificates. After SNI is enabled, the client is allowed to submit the requested domain name when initiating a TLS handshake request. After receiving the TLS request, the load balancer searches for the certificate based on the domain name in the request. If the certificate corresponding to the domain name is found, the load balancer returns the certificate for authorization. Otherwise, the default certificate (server certificate) is returned for authorization.

 **NOTE**

- The **SNI** option is available only when **HTTPS** is used.
  - This function is supported only in clusters of v1.15.11 or later.
  - Only one domain name can be specified for each SNI certificate. Wildcard-domain certificates are supported.
  - For ingresses connected to the same ELB port, do not configure SNIs with the same domain name but different certificates. Otherwise, the SNIs will be overwritten.
- **Security Policy:** combinations of different TLS versions and supported cipher suites available to HTTPS listeners.


For details about security policies, see .

 **NOTE**

- **Security Policy** is available only when **HTTPS** is selected.
  - This function is supported only in clusters of v1.17.9 or later.
- **Backend Protocol:**  
When the **listener** is HTTP-compliant, only **HTTP** can be selected.  
If it is an **HTTPS listener**, this parameter can be set to **HTTP** or **HTTPS**.
- **Advanced Options**

Configuration	Description	Restrictions
Idle Timeout	Timeout for an idle client connection. If there are no requests reaching the load balancer during the timeout duration, the load balancer will disconnect the connection from the client and establish a new connection when there is a new request.	None

Configuration	Description	Restrictions
Request Timeout	<p>Timeout for waiting for a request from a client. There are two cases:</p> <ul style="list-style-type: none"> <li>▪ If the client fails to send a request header to the load balancer during the timeout duration, the request will be interrupted.</li> <li>▪ If the interval between two consecutive request bodies reaching the load balancer is greater than the timeout duration, the connection will be disconnected.</li> </ul>	None
Response Timeout	<p>Timeout for waiting for a response from a backend server. After a request is forwarded to the backend server, if the backend server does not respond during the timeout duration, the load balancer will stop waiting and return HTTP 504 Gateway Timeout.</p>	None
HTTP2	<p>Whether to use HTTP/2 for a client to communicate with a load balancer. Request forwarding using HTTP/2 improves the access performance between your application and the load balancer. However, the load balancer still uses HTTP/1.x to forward requests to the backend server.</p>	<p>This function is available only when the <b>listener</b> is HTTPS-compliant.</p>

- **Forwarding Policy:** When the access address of a request matches the forwarding policy (a forwarding policy consists of a domain name and URL, for example, 10.117.117.117:80/helloworld), the request is forwarded to the corresponding target Service for processing. You can click  to add multiple forwarding policies.
  - **Domain Name:** Enter an actual domain name to be accessed. If it is left blank, the ingress can be accessed through the IP address. Ensure that the domain name has been registered and licensed. Once a forwarding policy is configured with a domain name specified, you must use the domain name for access.
  - **Path Matching Rule:**

- **Prefix match:** If the URL is set to `/healthz`, the URL that meets the prefix can be accessed, for example, `/healthz/v1` and `/healthz/v2`.
- **Exact match:** The URL can be accessed only when it is fully matched. For example, if the URL is set to `/healthz`, only `/healthz` can be accessed.
- **RegEX match:** The URL is matched based on the regular expression. For example, if the regular expression is `/[A-Za-z0-9_-]+/test`, all URLs that comply with this rule can be accessed, for example, `/abcA9/test` and `/v1-Ab/test`. Two regular expression standards are supported: POSIX and Perl.

- **Path:** access path, for example, `/healthz`

 **NOTE**

The access path added here must exist in the backend application. Otherwise, the forwarding fails.

For example, the default access URL of the Nginx application is `/usr/share/nginx/html`. When adding `/test` to the ingress forwarding policy, ensure the access URL of your Nginx application contains `/usr/share/nginx/html/test`. Otherwise, error 404 will be returned.

- **Destination Service:** Select an existing Service or create a Service. Any Services that do not match the search criteria will be filtered out automatically.
- **Destination Service Port:** Select the access port of the destination Service.
- **Set ELB:**
  - **Algorithm:** Three algorithms are available: weighted round robin, weighted least connections algorithm, or source IP hash.

 **NOTE**

- **Weighted round robin:** Requests are forwarded to different servers based on their weights, which indicate server processing performance. Backend servers with higher weights receive proportionately more requests, whereas equal-weighted servers receive the same number of requests. This algorithm is often used for short connections, such as HTTP services.
  - **Weighted least connections:** In addition to the weight assigned to each server, the number of connections processed by each backend server is considered. Requests are forwarded to the server with the lowest connections-to-weight ratio. Building on **least connections**, the **weighted least connections** algorithm assigns a weight to each server based on their processing capability. This algorithm is often used for persistent connections, such as database connections.
  - **Source IP hash:** The source IP address of each request is calculated using the hash algorithm to obtain a unique hash key, and all backend servers are numbered. The generated key allocates the client to a particular server. This enables requests from different clients to be distributed in load balancing mode and ensures that requests from the same client are forwarded to the same server. This algorithm applies to TCP connections without cookies.
- **Sticky Session:** This function is disabled by default. Options are as follows:

- **Load balancer cookie:** Enter the **Stickiness Duration** , which ranges from 1 to 1440 minutes.
- **Application cookie:** This parameter is available only for shared load balancers. In addition, enter **Cookie Name**, which ranges from 1 to 64 characters.

 **NOTE**

- When the **distribution policy** uses the source IP hash, sticky session cannot be set.
  - Dedicated load balancers in the clusters of a version earlier than v1.21 do not support sticky sessions. If sticky sessions are required, use shared load balancers.
- **Health Check:** Set the health check configuration of the load balancer. If this function is enabled, the following configurations are supported:

Parameter	Description
Protocol	When the protocol of the target Service port is TCP, more protocols including HTTP are supported. <ul style="list-style-type: none"> <li>○ <b>Check Path</b> (supported only by HTTP for health check): specifies the health check URL. The check path must start with a slash (/) and contain 1 to 80 characters.</li> </ul>
Port	By default, the service port (NodePort or container port of the Service) is used for health check. You can also specify another port for health check. After the port is specified, a service port named <b>cce-healthz</b> will be added for the Service. <ul style="list-style-type: none"> <li>○ <b>Node Port:</b> If a shared load balancer is used or no ENI instance is associated, the node port is used as the health check port. If this parameter is not specified, a random port is used. The value ranges from 30000 to 32767.</li> <li>○ <b>Container Port:</b> When a dedicated load balancer is associated with an ENI instance, the container port is used for health check. The value ranges from 1 to 65535.</li> </ul>
Check Period (s)	Specifies the maximum interval between health checks. The value ranges from 1 to 50.
Timeout (s)	Specifies the maximum timeout duration for each health check. The value ranges from 1 to 50.
Max. Retries	Specifies the maximum number of health check retries. The value ranges from 1 to 10.

- **Operation:** Click **Delete** to delete the configuration.
- **Annotation:** Ingresses provide some advanced CCE functions, which are implemented by annotations. When you use kubectl to create a container, annotations will be used. For details, see [Automatically Creating a Load Balancer While Creating an Ingress](#) or [Associating an Existing Load Balancer to an Ingress While Creating the Ingress](#).

**Step 4** Click **OK**. After the ingress is created, it is displayed in the ingress list.

On the ELB console, you can check the load balancer automatically created through CCE. The default name is **cce-lb-`<ingress.UID>`**. Click the load balancer name to go to the details page. On the **Listeners** tab page, check the listener and forwarding policy of the target ingress.

---

**NOTICE**

After an ingress is created, upgrade and maintain the selected load balancer on the CCE console. Do not modify the configuration on the ELB console. Otherwise, the ingress service may be abnormal.

---

**Step 5** Access the `/healthz` interface of the workload, for example, workload **defaultbackend**.

1. Obtain the access address of the `/healthz` interface of the workload. The access address consists of the load balancer IP address, external port, and mapping URL, for example, `10.**.**.**:80/healthz`.
2. Enter the URL of the `/healthz` interface, for example, `http://10.**.**.**:80/healthz`, in the address box of the browser to access the workload, as shown in [Figure 7-35](#).

**Figure 7-35** Accessing the `/healthz` interface of defaultbackend



----End

### 7.4.2.2 Creating a LoadBalancer Ingress Using kubectl

This section uses an [Nginx workload](#) as an example to describe how to create a LoadBalancer ingress using kubectl.

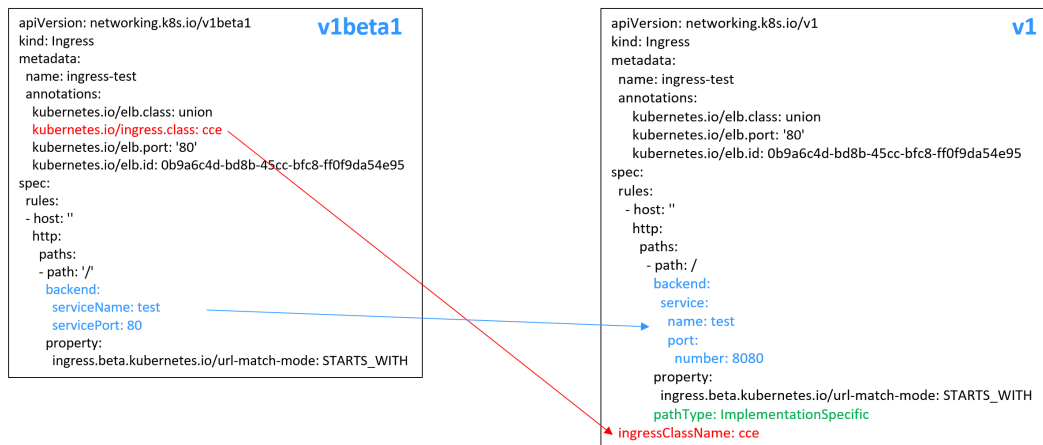
- If no load balancer is available in the same VPC, CCE can automatically create a load balancer when creating an ingress. For details, see [Automatically Creating a Load Balancer While Creating an Ingress](#).
- If a load balancer is available in the same VPC, perform the operation by referring to [Associating an Existing Load Balancer to an Ingress While Creating the Ingress](#).

## Ingress API Version Upgrade in CCE Clusters v1.23

In CCE clusters of v1.23 or later, the ingress version is switched to **networking.k8s.io/v1**.

Compared with v1beta1, v1 has the following differences in parameters:

- The ingress type is specified by **spec.ingressClassName** instead of **kubernetes.io/ingress.class** in **annotations**.
- The format of **backend** has changed.
- The **pathType** parameter must be specified for each path. The options are as follows:
  - **ImplementationSpecific**: The matching method depends on Ingress Controller. The matching method defined by **ingress.beta.kubernetes.io/url-match-mode** is used in CCE, which is the same as v1beta1.
  - **Exact**: exact matching of the URL, which is case-sensitive.
  - **Prefix**: matching based on the URL prefix separated by a slash (/). The match is case-sensitive, and elements in the path are matched one by one. A path element refers to a list of labels in the path separated by a slash (/).



### Prerequisites

- An available workload has been deployed in the cluster for external access. If no workload is available, deploy a workload by referring to [Creating a Deployment](#), [Creating a StatefulSet](#), or [Creating a DaemonSet](#).
- A Service for external access has been configured for the workload. [Services Supported by LoadBalancer Ingresses](#) lists the Service types supported by LoadBalancer ingresses.

### Creating an Ingress Using kubectl

You can determine whether to automatically create a load balancer or use an existing one when creating an ingress.

### Automatically Creating a Load Balancer While Creating an Ingress

The following describes how to run the kubectl command to automatically create a load balancer when creating an ingress.



**Step 1** Use kubectl to access the cluster. For details, see [Connecting to a Cluster Using kubectl](#).

**Step 2** Create a YAML file named **ingress-test.yaml**. The file name can be customized.

```
vi ingress-test.yaml
```

 **NOTE**

Starting from cluster v1.23, the ingress version is switched from **networking.k8s.io/v1beta1** to **networking.k8s.io/v1**. For details about the differences between v1 and v1beta1, see [Ingress API Version Upgrade in CCE Clusters v1.23](#).

**Example of a dedicated load balancer (public network access) for clusters of v1.23 or later:**

```
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.autocreate:
      '{
        "type": "public",
        "bandwidth_name": "cce-bandwidth-*****",
        "bandwidth_chargemode": "bandwidth",
        "bandwidth_size": 5,
        "bandwidth_sharetype": "PER",
        "eip_type": "5_bgp",
        "vip_subnet_cidr_id": "*****",
        "vip_address": "**.*.*.*",
        "elb_virsubnet_ids": ["*****"],
        "available_zone": [
          "eu-west-101a"
        ],
        "l7_flavor_name": "L7_flavor.elb.s1.small"
      }'
spec:
  rules:
    - host: ""
      http:
        paths:
          - path: '/'
            backend:
              service:
                name: <your_service_name> # Replace it with the name of your target Service.
                port:
                  number: <your_service_port> # Replace it with the port number of your target Service.
              property:
                ingress.beta.kubernetes.io/url-match-mode: STARTS_WITH
                pathType: ImplementationSpecific
            ingressClassName: cce
```

**Example of a dedicated load balancer (public network access) for clusters of v1.21 or earlier:**

```
apiVersion: networking.k8s.io/v1beta1
kind: Ingress
metadata:
  name: ingress-test
  namespace: default
  annotations:
    kubernetes.io/elb.class: performance
    kubernetes.io/ingress.class: cce
    kubernetes.io/elb.port: '80'
    kubernetes.io/elb.autocreate:
      '{
```

```

"type": "public",
"bandwidth_name": "cce-bandwidth-*****",
"bandwidth_chargemode": "bandwidth",
"bandwidth_size": 5,
"bandwidth_sharetype": "PER",
"eip_type": "5_bgp",
"elb_virsubnet_ids":["*****"],
"available_zone": [
  "eu-west-101a"
],
"l7_flavor_name": "L7_flavor.elb.s1.small"
}
spec:
rules:
- host: "
  http:
  paths:
  - path: '/'
  backend:
  serviceName: <your_service_name> # Replace it with the name of your target Service.
  servicePort: <your_service_port> # Replace it with the port number of your target Service.
property:
ingress.beta.kubernetes.io/url-match-mode: STARTS_WITH

```

**Table 7-44** Key parameters

Parameter	Mandatory	Type	Description
kubernetes.io/elb.class	Yes	String	Select a proper load balancer type. <ul style="list-style-type: none"> <li><b>performance:</b> dedicated load balancer</li> </ul>
kubernetes.io/ingress.class	Yes (only for clusters of v1.21 or earlier)	String	<b>cce:</b> A proprietary LoadBalancer ingress is used. This parameter is mandatory when an ingress is created by calling the API.
ingressClassName	Yes (only for clusters of v1.23 or later)	String	<b>cce:</b> A proprietary LoadBalancer ingress is used. This parameter is mandatory when an ingress is created by calling the API.
kubernetes.io/elb.port	Yes	String	This parameter indicates the external port registered with the address of the LoadBalancer Service. The value ranges from 1 to 65535. <p><b>NOTE</b> Some ports are high-risk ports and are blocked by default, for example, port 21.</p>

Parameter	Mandatory	Type	Description
kubernetes.io/elb.subnet-id	None	String	<p>ID of the subnet where the cluster is located. The value can contain 1 to 100 characters.</p> <ul style="list-style-type: none"> <li>• Mandatory when a cluster of v1.11.7-r0 or earlier is to be automatically created.</li> <li>• Optional for clusters later than v1.11.7-r0. It is left blank by default.</li> </ul>
kubernetes.io/elb.enterpriseID	No	String	<p><b>Kubernetes clusters of v1.15 and later versions support this field. In Kubernetes clusters earlier than v1.15, load balancers are created in the default project by default.</b></p> <p>ID of the enterprise project in which the load balancer will be created.</p> <p>The value contains 1 to 100 characters.</p> <p><b>How to obtain:</b></p> <p>Log in to the management console and choose <b>Enterprise &gt; Project Management</b> on the top menu bar. In the list displayed, click the name of the target enterprise project, and copy the ID on the enterprise project details page.</p>

Parameter	Mandatory	Type	Description
kubernetes.io/elb.autocreate	Yes	elb.autocreate object	<p>Whether to automatically create a load balancer associated with an ingress. For details about the field description, see <a href="#">Table 7-45</a>.</p> <p><b>Example</b></p> <ul style="list-style-type: none"> <li>Automatically created shared load balancer with an EIP bound:  <pre>{ "type": "public", "bandwidth_name": "cce-bandwidth-*****", "bandwidth_chargemode": "bandwidth", "bandwidth_size": 5, "bandwidth_sharetype": "P", "eip_type": "5_bgp", "name": "james" }</pre> </li> <li>Automatically created shared load balancer:  <pre>{ "type": "inner", "name": "A-location-d-test" }</pre> </li> </ul>
host	No	String	<p>Domain name for accessing the Service. By default, this parameter is left blank, and the domain name needs to be fully matched. Ensure that the domain name has been registered and licensed. Once a forwarding policy is configured with a domain name specified, you must use the domain name for access.</p>
path	Yes	String	<p>User-defined route path. All external access requests must match <b>host</b> and <b>path</b>.</p> <p><b>NOTE</b></p> <p>The access path added here must exist in the backend application. Otherwise, the forwarding fails.</p> <p>For example, the default access URL of the Nginx application is <b>/usr/share/nginx/html</b>. When adding <b>/test</b> to the ingress forwarding policy, ensure the access URL of your Nginx application contains <b>/usr/share/nginx/html/test</b>. Otherwise, error 404 will be returned.</p>

Parameter	Mandatory	Type	Description
ingress.beta.kubernetes.io/url-match-mode	No	String	Route matching policy. Default: <b>STARTS_WITH</b> (prefix match) Options: <ul style="list-style-type: none"> <li>• <b>EQUAL_TO</b>: exact match</li> <li>• <b>STARTS_WITH</b>: prefix match</li> <li>• <b>REGEX</b>: regular expression match</li> </ul>

Parameter	Mandatory	Type	Description
pathType	Yes	String	<p>Path type. This field is supported only by clusters of v1.23 or later.</p> <ul style="list-style-type: none"> <li>• <b>ImplementationSpecific:</b> The matching method depends on Ingress Controller. The matching method defined by <b>ingress.beta.kubernetes.io/url-match-mode</b> is used in CCE.</li> <li>• <b>Exact:</b> exact matching of the URL, which is case-sensitive.</li> <li>• <b>Prefix:</b> prefix matching, which is case-sensitive. With this method, the URL path is separated into multiple elements by slashes (/) and the elements are matched one by one. If each element in the URL matches the path, the subpaths of the URL can be routed normally.</li> </ul> <p><b>NOTE</b></p> <ul style="list-style-type: none"> <li>- During prefix matching, each element must be exactly matched. If the last element of the URL is the substring of the last element in the request path, no matching is performed. For example, <b>/foo/bar</b> matches <b>/foo/bar/baz</b> but does not match <b>/foo/barbaz</b>.</li> <li>- When elements are separated by slashes (/), if the URL or request path ends with a slash (/), the slash (/) at the end is ignored. For example, <b>/foo/bar</b> matches <b>/foo/bar/</b>.</li> </ul> <p>See <a href="#">examples</a> of ingress path matching.</p>

**Table 7-45** elb.autocreate data structure

Parameter	Mandatory	Type	Description
name	No	String	Name of the automatically created load balancer. The value can contain 1 to 64 characters. Only letters, digits, underscores (_), hyphens (-), and periods (.) are allowed. Default: <b>cce-lb+service.UID</b>
type	No	String	Network type of the load balancer. <ul style="list-style-type: none"> <li>• <b>public</b>: public network load balancer</li> <li>• <b>inner</b>: private network load balancer</li> </ul> Default: <b>inner</b>
bandwidth_name	Yes for public network load balancers	String	Bandwidth name. The default value is <b>cce-bandwidth-*****</b> . The value can contain 1 to 64 characters. Only letters, digits, underscores (_), hyphens (-), and periods (.) are allowed.
bandwidth_chargemode	No	String	Bandwidth mode. <ul style="list-style-type: none"> <li>• <b>bandwidth</b>: billed by bandwidth</li> <li>• <b>traffic</b>: billed by traffic</li> </ul> Default: <b>bandwidth</b>
bandwidth_size	Yes for public network load balancers	Integer	Bandwidth size. The value ranges from 1 Mbit/s to 2000 Mbit/s by default. Configure this parameter based on the bandwidth range allowed in your region. The minimum increment for bandwidth adjustment varies depending on the bandwidth range. <ul style="list-style-type: none"> <li>• The minimum increment is 1 Mbit/s if the allowed bandwidth does not exceed 300 Mbit/s.</li> <li>• The minimum increment is 50 Mbit/s if the allowed bandwidth ranges from 300 Mbit/s to 1000 Mbit/s.</li> <li>• The minimum increment is 500 Mbit/s if the allowed bandwidth exceeds 1000 Mbit/s.</li> </ul>

Parameter	Mandatory	Type	Description
bandwidth_sharetype	Yes for public network load balancers	String	Bandwidth sharing mode. <ul style="list-style-type: none"> <li>• <b>PER</b>: dedicated bandwidth</li> </ul>
eip_type	Yes for public network load balancers	String	EIP type. <ul style="list-style-type: none"> <li>• <b>5_bgp</b>: dynamic BGP</li> <li>• <b>5_sbgp</b>: static BGP</li> </ul> The specific type varies with regions. For details, see the EIP console.
vip_subnet_cidr_id	No	String	Subnet where a load balancer is located. The subnet must belong to the VPC where the cluster resides. <p>If this parameter is not specified, the ELB load balancer and the cluster are in the same subnet.</p> <p>This field can be specified only for clusters of v1.21 or later.</p>
vip_address	No	String	Private IP address of the load balancer. Only IPv4 addresses are supported. <p>The IP address must be in the ELB CIDR block. If this parameter is not specified, an IP address will be automatically assigned from the ELB CIDR block.</p> <p>This parameter is available only in clusters of v1.23.11-r0, v1.25.6-r0, v1.27.3-r0, or later versions.</p>

**Step 3** Create an ingress.

```
kubectl create -f ingress-test.yaml
```

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

```
ingress/ingress-test created
```

**Step 4** Check the created ingress.

```
kubectl get ingress
```

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

```
NAME      CLASS  HOSTS      ADDRESS      PORTS  AGE
ingress-test  cce    *          121.**.**.**  80     10s
```

**Step 5** Enter **http://121.\*\*.\*\*.\*\*80** in the address box of the browser to access the workload (for example, [Nginx workload](#)).



*121.\*\*.\*\** indicates the IP address of the unified load balancer.

----End

## Associating an Existing Load Balancer to an Ingress While Creating the Ingress

CCE allows you to connect to an existing load balancer when creating an ingress.

**Step 1** Use `kubectl` to access the cluster. For details, see [Connecting to a Cluster Using kubectl](#).

**Step 2** Create a YAML file named `ingress-test.yaml`. The file name can be customized.

```
vi ingress-test.yaml
apiVersion: networking.k8s.io/v1beta1
kind: Ingress
metadata:
  name: ingress-test
  annotations:
    kubernetes.io/elb.id: <your_elb_id> # Replace it with the ID of your existing load balancer.
    kubernetes.io/elb.ip: <your_elb_ip> # Replace it with the IP of your existing load balancer.
    kubernetes.io/elb.class: performance # Load balancer type
    kubernetes.io/elb.port: '80'
    kubernetes.io/ingress.class: cce
spec:
  rules:
  - host: ""
    http:
      paths:
      - path: '/'
        backend:
          serviceName: <your_service_name> # Replace it with the name of your target Service.
          servicePort: 80
    property:
      ingress.beta.kubernetes.io/url-match-mode: STARTS_WITH
```

**Table 7-46** Key parameters

Parameter	Mandatory	Type	Description
kubernetes.io/elb.id	Yes	String	ID of a load balancer. The value can contain 1 to 100 characters. <b>How to obtain:</b> On the management console, click <b>Service List</b> , and choose <b>Networking &gt; Elastic Load Balance</b> . Click the name of the target load balancer. On the <b>Summary</b> tab page, find and copy the ID.
kubernetes.io/elb.ip	No	String	Service address of a load balancer. The value can be the public IP address of a public network load balancer or the private IP address of a private network load balancer.

Parameter	Mandatory	Type	Description
kubernetes.io/elb.class	Yes	String	<p>Load balancer type.</p> <ul style="list-style-type: none"> <li><b>performance:</b> dedicated load balancer, which can be used only in clusters of v1.17 and later.</li> </ul> <p><b>NOTE</b> If a LoadBalancer ingress accesses an existing dedicated load balancer, the dedicated load balancer must be of the application load balancing (HTTP/HTTPS) type.</p>

**Step 3** Create an ingress.

```
kubectl create -f ingress-test.yaml
```

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

```
ingress/ingress-test created
```

**Step 4** Check the created ingress.

```
kubectl get ingress
```

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

```
NAME          CLASS  HOSTS  ADDRESS      PORTS  AGE
ingress-test  cce    *      121.**.**.**   80     10s
```

**Step 5** Enter **http://121.\*\*.\*\*.\*\*:80** in the address box of the browser to access the workload (for example, [Nginx workload](#)).

**121.\*\*.\*\*.\*\*** indicates the IP address of the unified load balancer.

----End

### 7.4.2.3 Annotations for Configuring LoadBalancer Ingresses

You can add annotations to a YAML file for more advanced ingress functions. This section describes the annotations that can be used when you create a LoadBalancer ingress.

## Indexes

Category	Ingress Annotation
Load balancer configuration	<ul style="list-style-type: none"> <li><a href="#">Basic Configurations for Interconnecting with ELB</a></li> </ul>
Port or protocol configuration	<ul style="list-style-type: none"> <li><a href="#">Configuring ELB Certificates</a></li> <li><a href="#">Using HTTP/2</a></li> <li><a href="#">Interconnecting with HTTPS Backend Services</a></li> </ul>
Advanced features of ELB listeners	<ul style="list-style-type: none"> <li><a href="#">Configuring Timeout for an Ingress</a></li> <li><a href="#">Configuring a Slow Start</a></li> </ul>

## Basic Configurations for Interconnecting with ELB

Application scenarios and use cases:

- Associate an existing load balancer. For details, see [Associating an Existing Load Balancer to an Ingress While Creating the Ingress](#).
- Automatically create a load balancer. For details, see [Automatically Creating a Load Balancer While Creating an Ingress](#).

**Table 7-47** Annotations for interconnecting with ELB

Parameter	Type	Description	Supported Cluster Version
kubernetes.io/elb.class	String	Select a proper load balancer type. Options: <ul style="list-style-type: none"> <li>• <b>performance</b>: dedicated load balancer, which can be used only in clusters of v1.17 and later.</li> </ul>	v1.9 or later
kubernetes.io/ingress.class	String	<ul style="list-style-type: none"> <li>• <b>cce</b>: A proprietary LoadBalancer ingress is used.</li> <li>• <b>nginx</b>: Nginx Ingress is used.</li> </ul> This parameter is mandatory when an ingress is created by calling the API. For clusters of v1.23 or later, use the parameter <b>ingressClassName</b> . For details, see <a href="#">Creating a LoadBalancer Ingress Using kubectl</a> .	Only clusters of v1.21 or earlier
kubernetes.io/elb.port	String	This parameter indicates the external port registered with the address of the LoadBalancer Service. The value ranges from 1 to 65535. <b>NOTE</b> Some ports are high-risk ports and are blocked by default, for example, port 21.	v1.9 or later
kubernetes.io/elb.id	String	Mandatory <b>when an existing load balancer is to be interconnected</b> . ID of a load balancer. <b>How to obtain:</b> On the management console, click <b>Service List</b> , and choose <b>Networking &gt; Elastic Load Balance</b> . Click the name of the target load balancer. On the <b>Summary</b> tab page, find and copy the ID.	v1.9 or later

Parameter	Type	Description	Supported Cluster Version
kubernetes.io/elb.ip	String	<p><b>Mandatory when an existing load balancer is to be interconnected.</b></p> <p>Service address of a load balancer. The value can be the public IP address of a public network load balancer or the private IP address of a private network load balancer.</p>	v1.9 or later
kubernetes.io/elb.autocreate	<a href="#">Table 7-53</a> Object	<p><b>Mandatory when load balancers are automatically created.</b></p> <p><b>Example</b></p> <ul style="list-style-type: none"> <li>Automatically created shared load balancer with an EIP bound: <code>'{"type":"public","bandwidth_name":"cce-bandwidth-1551163379627","bandwidth_chargemode":"bandwidth","bandwidth_size":5,"bandwidth_sharetype":"PER","eip_type":"5_bgp","name":"james"}</code></li> <li>Automatically created shared load balancer with no EIP bound: <code>{"type":"inner","name":"A-location-d-test"}</code></li> </ul>	v1.9 or later
kubernetes.io/elb.enterpriseID	String	<p><b>Optional when load balancers are automatically created.</b></p> <p><b>Clusters of v1.15 and later versions support this field. In clusters earlier than v1.15, load balancers are created in the default project by default.</b></p> <p>This parameter indicates the ID of the enterprise project in which the ELB load balancer will be created.</p> <p>If this parameter is not specified or is set to <b>0</b>, resources will be bound to the default enterprise project.</p> <p><b>How to obtain:</b></p> <p>Log in to the management console and choose <b>Enterprise &gt; Project Management</b> on the top menu bar. In the list displayed, click the name of the target enterprise project, and copy the ID on the enterprise project details page.</p>	v1.15 or later

Parameter	Type	Description	Supported Cluster Version
kubernetes.io/elb.subnet-id	String	<p>Optional <b>when load balancers are automatically created</b>.</p> <p>ID of the subnet where the cluster is located. The value can contain 1 to 100 characters.</p> <ul style="list-style-type: none"> <li>• Mandatory when a cluster of v1.11.7-r0 or earlier is to be automatically created.</li> <li>• Optional for clusters of a version later than v1.11.7-r0.</li> </ul>	<p>Mandatory for clusters earlier than v1.11.7-r0</p> <p>Discarded in clusters of a version later than v1.11.7-r0</p>

## Configuring ELB Certificates

For details about application scenarios and use cases, see [Using kubectl](#).

**Table 7-48** ELB certificate annotations

Parameter	Type	Description	Supported Cluster Version
kubernetes.io/elb.tls-certificate-ids	String	<p>ELB certificate IDs, which are separated by comma (.). The list length is greater than or equal to 1. The first ID in the list is the server certificate, and the other IDs are SNI certificates in which a domain name must be contained.</p> <p>To obtain the certificate, log in to the CCE console, choose <b>Service List &gt; Networking &gt; Elastic Load Balance</b>, and click <b>Certificates</b> in the navigation pane. In the load balancer list, copy the ID under the target certificate name.</p>	v1.19.16-r2, v1.21.5-r0, v1.23.3-r0, or later

## Using HTTP/2

For details about application scenarios and use cases, see [Configuring HTTP/2 for a LoadBalancer Ingress](#).

**Table 7-49** Annotations for using HTTP/2

Parameter	Type	Description	Supported Cluster Version
kubernetes.io/elb.http2-enable	String	<p>Whether HTTP/2 is enabled. Request forwarding using HTTP/2 improves the access performance between your application and the load balancer. However, the load balancer still uses HTTP/1.x to forward requests to the backend server.</p> <p>Options:</p> <ul style="list-style-type: none"> <li>• <b>true</b>: enabled</li> <li>• <b>false</b>: disabled (default value)</li> </ul> <p>Note: <b>HTTP/2 can be enabled or disabled only when the listener uses HTTPS</b>. This parameter is invalid and defaults to <b>false</b> when the listener protocol is HTTP.</p>	v1.23.13-r0, v1.25.8-r0, v1.27.5-r0, v1.28.3-r0, or later

## Interconnecting with HTTPS Backend Services

For details about application scenarios and use cases, see [Configuring HTTPS Backend Services for a LoadBalancer Ingress](#).

**Table 7-50** Annotations for interconnecting with HTTPS backend services

Parameter	Type	Description	Supported Cluster Version
kubernetes.io/elb.pool-protocol	String	To interconnect with HTTPS backend services, set this parameter to <b>https</b> .	v1.23.8, v1.25.3, or later

## Configuring Timeout for an Ingress

For details about application scenarios and use cases, see [Configuring Timeout for a LoadBalancer Ingress](#).

**Table 7-51** Annotations for configuring timeout of an ingress

Parameter	Type	Description	Supported Cluster Version
kubernetes.io/elb.KeepAliveTimeout	String	<p>Timeout for client connections. If there are no requests reaching the load balancer during the timeout duration, the load balancer will disconnect the connection from the client and establish a new connection when there is a new request.</p> <p>Value:</p> <ul style="list-style-type: none"> <li>For TCP listeners, the value ranges from <b>10</b> to <b>4000</b> (in seconds). The default value is <b>300</b>.</li> <li>For HTTP or HTTPS listeners, the value ranges from <b>0</b> to <b>4000</b> (in seconds). The default value is <b>60</b>.</li> </ul> <p>For UDP listeners, this parameter does not take effect.</p>	<p>Dedicated load balancers: v1.19.16-r30, v1.21.10-r10, v1.23.8-r10, v1.25.3-r10, or later</p> <p>Shared load balancers: v1.23.13-r0, v1.25.8-r0, v1.28.3-r0, or later</p>
kubernetes.io/elb.clientTimeout	String	<p>Timeout for waiting for a request from a client. There are two cases:</p> <ul style="list-style-type: none"> <li>If the client fails to send a request header to the load balancer during the timeout duration, the request will be interrupted.</li> <li>If the interval between two consecutive request bodies reaching the load balancer is greater than the timeout duration, the connection will be disconnected.</li> </ul> <p>The value ranges from <b>1</b> to <b>300</b> (in seconds). The default value is <b>60</b>.</p> <p>This parameter is available only for HTTP and HTTPS listeners.</p> <p>Minimum value: <b>1</b> Maximum value: <b>300</b> Default value: <b>60</b></p>	

Parameter	Type	Description	Supported Cluster Version
kubernetes.io/elb.member_timeout	String	<p>Timeout for waiting for a response from a backend server. After a request is forwarded to the backend server, if the backend server does not respond within the duration specified by <b>member_timeout</b>, the load balancer will stop waiting and return HTTP 504 Gateway Timeout.</p> <p>The value ranges from <b>1</b> to <b>300</b> (in seconds). The default value is <b>60</b>.</p> <p>This parameter is available only for HTTP and HTTPS listeners.</p> <p>Minimum value: <b>1</b> Maximum value: <b>300</b> Default value: <b>60</b></p>	

## Configuring a Slow Start

For details about application scenarios and use cases, see [Configuring a Slow Start for a LoadBalancer Ingress](#).

**Table 7-52** Annotations for configuring a slow start

Parameter	Type	Description	Supported Cluster Version
kubernetes.io/elb.slowstart	String	<p>Duration of slow start, in seconds. The slow start duration ranges from 30 to 1200.</p> <ul style="list-style-type: none"> <li>This configuration applies only to dedicated load balancers.</li> <li>Valid only when the allocation policy of the target Service is weighted round robin (WRR) and sticky session is disabled.</li> </ul> <p><b>NOTE</b> The load balancer linearly increases the proportion of requests to backend servers in slow start mode. When the configured slow start duration elapses, the load balancer sends full share of requests to backend servers and exits the slow start mode.</p>	v1.23 or later



## Parameters for Automatically Creating a Load Balancer

**Table 7-53** elb.autocreate data structure

Parameter	Mandatory	Type	Description
name	No	String	Name of the automatically created load balancer. The value can contain 1 to 64 characters. Only letters, digits, underscores (_), hyphens (-), and periods (.) are allowed. Default: <b>cce-lb+service.UID</b>
type	No	String	Network type of the load balancer. <ul style="list-style-type: none"> <li><b>public</b>: public network load balancer</li> <li><b>inner</b>: private network load balancer</li> </ul> Default: <b>inner</b>
bandwidth_name	Yes for public network load balancers	String	Bandwidth name. The default value is <b>cce-bandwidth-*****</b> . The value can contain 1 to 64 characters. Only letters, digits, underscores (_), hyphens (-), and periods (.) are allowed.
bandwidth_chargemode	No	String	Bandwidth mode. <ul style="list-style-type: none"> <li><b>bandwidth</b>: billed by bandwidth</li> <li><b>traffic</b>: billed by traffic</li> </ul> Default: <b>bandwidth</b>
bandwidth_size	Yes for public network load balancers	Integer	Bandwidth size. The value ranges from 1 Mbit/s to 2000 Mbit/s by default. Configure this parameter based on the bandwidth range allowed in your region. The minimum increment for bandwidth adjustment varies depending on the bandwidth range. <ul style="list-style-type: none"> <li>The minimum increment is 1 Mbit/s if the allowed bandwidth does not exceed 300 Mbit/s.</li> <li>The minimum increment is 50 Mbit/s if the allowed bandwidth ranges from 300 Mbit/s to 1000 Mbit/s.</li> <li>The minimum increment is 500 Mbit/s if the allowed bandwidth exceeds 1000 Mbit/s.</li> </ul>

Parameter	Mandatory	Type	Description
bandwidth_sharetype	Yes for public network load balancers	String	Bandwidth sharing mode. <ul style="list-style-type: none"> <li>• <b>PER</b>: dedicated bandwidth</li> </ul>
eip_type	Yes for public network load balancers	String	EIP type. <ul style="list-style-type: none"> <li>• <b>5_bgp</b>: dynamic BGP</li> <li>• <b>5_sbgp</b>: static BGP</li> </ul> The specific type varies with regions. For details, see the EIP console.
vip_subnet_cidr_id	No	String	Subnet where a load balancer is located. The subnet must belong to the VPC where the cluster resides. <p>If this parameter is not specified, the ELB load balancer and the cluster are in the same subnet.</p> <p>This field can be specified only for clusters of v1.21 or later.</p>
vip_address	No	String	Private IP address of the load balancer. Only IPv4 addresses are supported. <p>The IP address must be in the ELB CIDR block. If this parameter is not specified, an IP address will be automatically assigned from the ELB CIDR block.</p> <p>This parameter is available only in clusters of v1.23.11-r0, v1.25.6-r0, v1.27.3-r0, or later versions.</p>

## 7.4.2.4 Advanced Setting Examples of LoadBalancer Ingresses

### 7.4.2.4.1 Configuring an HTTPS Certificate for a LoadBalancer Ingress

Ingresses support SSL or TLS certificates, allowing you to secure your Services with HTTPS.

You are allowed to use either of the following ways to configure an ingress certificate in a cluster:

- **Using a TLS Certificate:** You need to first import a certificate to a Secret. CCE will then automatically handle the certificate configurations on the ELB console and give a name to the certificate (started with **k8s\_plb\_default**). This certificate, which is generated by CCE, **cannot be modified or deleted from the ELB console**. To modify the certificate, update the secret where the certificate is imported to on CCE.

- **Using a Certificate Created on the ELB Console:** You are allowed to directly use certificates created on the ELB console. There is no need to manually configure the cluster Secrets, and you can modify the certificates on the ELB console.

 **NOTE**

If HTTPS is enabled for the same port of the same load balancer of multiple ingresses, you must select the same certificate. For details, see [Configuring a Same Port for Multiple Ingresses](#).

## Prerequisites

- An available workload has been deployed in the cluster for external access. If no workload is available, deploy a workload by referring to [Creating a Deployment](#), [Creating a StatefulSet](#), or [Creating a DaemonSet](#).
- A Service for external access has been configured for the workload. [Services Supported by LoadBalancer Ingresses](#) lists the Service types supported by LoadBalancer ingresses.
- You can obtain a trusted certificate from a certificate provider.

## Using a TLS Certificate

You can configure a TLS certificate using either the CCE console or kubectl.

## Using the CCE Console

- Step 1** Log in to the CCE console and click the cluster name to access the cluster console.
- Step 2** Choose **Services & Ingresses** in the navigation pane, click the **Ingresses** tab, and click **Create Ingress** in the upper right corner.
- Step 3** Configure ingress parameters.

 **NOTE**

This example explains only key parameters for configuring HTTPS certificates. You can configure other parameters as required. For details, see [Creating a LoadBalancer Ingress on the Console](#).

**Table 7-54** Key parameters

Parameter	Description	Example
Name	Enter an ingress name.	ingress-test
Load Balancer	Select a load balancer to be associated with the ingress or automatically create a load balancer.	Shared

Parameter	Description	Example
Listener	<ul style="list-style-type: none"> <li>● <b>External Protocol:</b> Select <b>HTTPS</b> when configuring a certificate for an ingress.</li> <li>● <b>External Port:</b> specifies the port of the load balancer listener. The default HTTPS port is 443.</li> <li>● <b>Certificate Source:</b> Select <b>TLS secret</b>.</li> <li>● <b>Server Certificate:</b> <b>kubernetes.io/tls</b> and <b>IngressTLS</b> are supported. If no certificate is available, you can create a TLS certificate. For details about the configuration parameters, see <a href="#">Creating a Secret</a>.</li> </ul>	<ul style="list-style-type: none"> <li>● External Protocol: HTTPS</li> <li>● External Port: 443</li> <li>● Certificate Source: TLS secret</li> <li>● Server Certificate: test</li> </ul>
Forwarding Policy	<ul style="list-style-type: none"> <li>● <b>Domain Name:</b> Enter an actual domain name to be accessed. If it is left blank, the ingress can be accessed through the IP address. Ensure that the domain name has been registered and licensed. Once a forwarding policy is configured with a domain name specified, you must use the domain name for access.</li> <li>● <b>Path Matching Rule:</b> Select <b>Prefix match</b>, <b>Exact match</b>, or <b>RegEx match</b>.</li> <li>● <b>Path:</b> Enter the path provided by a backend application for external access. The path added must be valid in the backend application, or the forwarding cannot take effect.</li> <li>● <b>Destination Service:</b> Select an existing Service or create a Service. Any Services that do not match the search criteria will be filtered out automatically.</li> <li>● <b>Destination Service Port:</b> Select the access port of the destination Service.</li> </ul>	<ul style="list-style-type: none"> <li>● Domain Name: You do not need to configure this parameter.</li> <li>● Path Matching Rule: Prefix match</li> <li>● Path: /</li> <li>● Destination Service: nginx</li> <li>● Destination Service Port: 80</li> </ul>

**Figure 7-36** Selecting a TLS certificate

The screenshot shows the 'Listeners' configuration interface. The 'Certificate Source' section is highlighted with a red box, showing 'ELB server certificate' selected and 'TLS secret' as an available option. Other visible settings include: External Protocol (HTTPS), External Port (443), Access Control (Not configured), Server Certificate (test), SNI (empty table), Security Policy (--Select--), and Backend Protocol (HTTP).

**Step 4** Click **OK**.

----End

## Using kubectl

**Step 1** Use kubectl to access the cluster. For details, see [Connecting to a Cluster Using kubectl](#).

**Step 2** Ingress supports two TLS secret types: kubernetes.io/tls and IngressTLS. IngressTLS is used as an example. For details, see [Creating a Secret](#). For details about examples of the kubernetes.io/tls secret and its description, see [TLS secrets](#).

Create a YAML file named **ingress-test-secret.yaml**. The file name can be customized.

```
vi ingress-test-secret.yaml
```

**The YAML file is configured as follows:**

```
apiVersion: v1
data:
  tls.crt: LS0*****tLS0tCg==
  tls.key: LS0tL*****0tLS0K
kind: Secret
metadata:
  annotations:
    description: test for ingressTLS secrets
    name: ingress-test-secret
    namespace: default
type: IngressTLS
```

 NOTE

In the preceding information, **tls.crt** and **tls.key** are only examples. Replace them with the actual files. The values of **tls.crt** and **tls.key** are Base64-encoded.

**Step 3** Create a secret.

```
kubectl create -f ingress-test-secret.yaml
```

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

```
secret/ingress-test-secret created
```

**Step 4** Check the created secret.

```
kubectl get secrets
```

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

NAME	TYPE	DATA	AGE
ingress-test-secret	IngressTLS	2	13s

**Step 5** Create a YAML file named **ingress-test.yaml**. The file name can be customized.

```
vi ingress-test.yaml
```

 NOTE

Default security policy (kubernetes.io/elb.tls-ciphers-policy) is supported only in clusters of v1.17.17 or later.

An example YAML file of an ingress associated with an automatically created load balancer is as follows:

```
apiVersion: networking.k8s.io/v1beta1
kind: Ingress
metadata:
  name: ingress-test
  annotations:
    kubernetes.io/elb.class: performance
    kubernetes.io/ingress.class: cce
    kubernetes.io/elb.port: '443'
    kubernetes.io/elb.autocreate:
      '{
        "type": "public",
        "bandwidth_name": "cce-bandwidth-*****",
        "bandwidth_chargemode": "bandwidth",
        "bandwidth_size": 5,
        "bandwidth_sharetype": "PER",
        "eip_type": "5_bgp",
        "available_zone": [
          "eu-west-101a"
        ],
        "elb_virsubnet_ids":["b4bf8152-6c36-4c3b-9f74-2229f8e640c9"],
        "l7_flavor_name": "L7_flavor.elb.s1.small"
      }'
    kubernetes.io/elb.tls-ciphers-policy: tls-1-2
spec:
  tls:
  - secretName: ingress-test-secret
  rules:
  - host: ""
    http:
      paths:
      - path: '/'
        backend:
          serviceName: <your_service_name> # Replace it with the name of your target Service.
          servicePort: 80
  property:
    ingress.beta.kubernetes.io/url-match-mode: STARTS_WITH
```

**Table 7-55** Key parameters

Parameter	Mandatory	Type	Description
kubernetes.io/elb.tls-ciphers-policy	No	String	The default value is <b>tls-1-2</b> , which is the default security policy used by the listener and takes effect only when HTTPS is used.  Options: <ul style="list-style-type: none"> <li>• tls-1-0</li> <li>• tls-1-1</li> <li>• tls-1-2</li> <li>• tls-1-2-strict</li> </ul> For details of cipher suites for each security policy, see <a href="#">Table 7-56</a> .
tls	No	Array of strings	When HTTPS is used, this parameter must be added to specify the secret certificate.  Multiple independent domain names and certificates can be added. For details, see <a href="#">Configuring SNI for a LoadBalancer Ingress</a> .
secretName	No	String	This parameter is mandatory if HTTPS is used. Set this parameter to the name of the created secret.

**Table 7-56** `tls_ciphers_policy` parameter description

Security Policy	TLS Version	Cipher Suite
tls-1-0	TLS 1.2 TLS 1.1 TLS 1.0	ECDHE-RSA-AES256-GCM-SHA384:ECDHE-RSA-AES128-GCM-SHA256:ECDHE-ECDSA-AES256-GCM-SHA384:ECDHE-ECDSA-AES128-GCM-SHA256:AES128-GCM-SHA256:AES256-GCM-SHA384:ECDHE-ECDSA-AES128-SHA256:ECDHE-RSA-AES128-SHA256:AES128-SHA256:AES256-SHA256:ECDHE-ECDSA-AES256-SHA384:ECDHE-RSA-AES256-SHA384:ECDHE-ECDSA-AES128-SHA:ECDHE-RSA-AES128-SHA:ECDHE-RSA-AES256-SHA:ECDHE-ECDSA-AES256-SHA:AES128-SHA:AES256-SHA
tls-1-1	TLS 1.2 TLS 1.1	ECDHE-RSA-AES128-SHA256:AES128-SHA256:AES256-SHA256:ECDHE-ECDSA-AES256-SHA384:ECDHE-RSA-AES256-SHA384:ECDHE-ECDSA-AES128-SHA:ECDHE-RSA-AES128-SHA:ECDHE-RSA-AES256-SHA:ECDHE-ECDSA-AES256-SHA:AES128-SHA:AES256-SHA
tls-1-2	TLS 1.2	ECDHE-RSA-AES256-SHA384:ECDHE-ECDSA-AES128-SHA:ECDHE-RSA-AES128-SHA:ECDHE-RSA-AES256-SHA:ECDHE-ECDSA-AES256-SHA:AES128-SHA:AES256-SHA

Security Policy	TLS Version	Cipher Suite
tls-1-2-strict	TLS 1.2	ECDHE-RSA-AES256-GCM-SHA384:ECDHE-RSA-AES128-GCM-SHA256:ECDHE-ECDSA-AES256-GCM-SHA384:ECDHE-ECDSA-AES128-GCM-SHA256:AES128-GCM-SHA256:AES256-GCM-SHA384:ECDHE-ECDSA-AES128-SHA256:ECDHE-RSA-AES128-SHA256:AES128-SHA256:AES256-SHA256:ECDHE-ECDSA-AES256-SHA384:ECDHE-RSA-AES256-SHA384
TLS-1-0-WITH-1-3 (dedicated load balancer)	TLS 1.3 TLS 1.2 TLS 1.1 TLS 1.0	ECDHE-RSA-AES256-GCM-SHA384:ECDHE-RSA-AES128-GCM-SHA256:ECDHE-ECDSA-AES256-GCM-SHA384:ECDHE-ECDSA-AES128-GCM-SHA256:AES128-GCM-SHA256:AES256-GCM-SHA384:ECDHE-ECDSA-AES128-SHA256:ECDHE-RSA-AES128-SHA256:AES128-SHA256:AES256-SHA256:ECDHE-ECDSA-AES256-SHA384:ECDHE-RSA-AES256-SHA384:ECDHE-ECDSA-AES128-SHA:ECDHE-RSA-AES128-SHA:ECDHE-RSA-AES256-SHA:ECDHE-ECDSA-AES256-SHA:AES128-SHA:AES256-SHA:TLS_AES_256_GCM_SHA384:TLS_CHACHA20_POLY1305_SHA256:TLS_AES_128_GCM_SHA256:TLS_AES_128_CCM_8_SHA256:TLS_AES_128_CCM_SHA256
TLS-1-2-FS (dedicated load balancer)	TLS 1.3 TLS 1.2	ECDHE-RSA-AES256-GCM-SHA384:ECDHE-RSA-AES128-GCM-SHA256:ECDHE-ECDSA-AES256-GCM-SHA384:ECDHE-ECDSA-AES128-GCM-SHA256:ECDHE-ECDSA-AES128-SHA256:ECDHE-RSA-AES128-SHA256:ECDHE-ECDSA-AES256-SHA384:ECDHE-RSA-AES256-SHA384
TLS-1-2-FS-WITH-1-3 (dedicated load balancer)	TLS 1.3 TLS 1.2	ECDHE-RSA-AES256-GCM-SHA384:ECDHE-RSA-AES128-GCM-SHA256:ECDHE-ECDSA-AES256-GCM-SHA384:ECDHE-ECDSA-AES128-GCM-SHA256:ECDHE-ECDSA-AES128-SHA256:ECDHE-RSA-AES128-SHA256:ECDHE-ECDSA-AES256-SHA384:ECDHE-RSA-AES256-SHA384:TLS_AES_256_GCM_SHA384:TLS_CHACHA20_POLY1305_SHA256:TLS_AES_128_GCM_SHA256:TLS_AES_128_CCM_8_SHA256:TLS_AES_128_CCM_SHA256

**Step 6** Create an ingress.

```
kubectl create -f ingress-test.yaml
```

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

```
ingress/ingress-test created
```

**Step 7** Check the created ingress.



```
kubectl get ingress
```

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

NAME	CLASS	HOSTS	ADDRESS	PORTS	AGE
ingress-test	cce	*	121.**.**.*	80,443	10s

**Step 8** Enter **https://121.\*\*.\*\*.\*:443** in the address box of the browser to access the workload (for example, [Nginx workload](#)).

**121.\*\*.\*\*.\*** indicates the IP address of the unified load balancer.

----End

## Using a Certificate Created on the ELB Console

You can configure a certificate created on the ELB console for an ingress using either the CCE console or kubectl.

### NOTE

- If both an ELB certificate and a TLS certificate are specified for the same ingress, the ingress will use the ELB certificate.
- CCE does not check whether an ELB certificate is valid. It only checks whether the certificate is present.
- Only ingresses in clusters of v1.19.16-r2, v1.21.5-r0, v1.23.3-r0, or later support ELB certificates.

## Using the CCE Console

**Step 1** Log in to the CCE console and click the cluster name to access the cluster console.

**Step 2** Choose **Services & Ingresses** in the navigation pane, click the **Ingresses** tab, and click **Create Ingress** in the upper right corner.

**Step 3** Configure ingress parameters.

### NOTE

This example explains only key parameters for configuring HTTPS certificates. You can configure other parameters as required. For details, see [Creating a LoadBalancer Ingress on the Console](#).

**Table 7-57** Key parameters

Parameter	Description	Example
Name	Enter an ingress name.	ingress-test
Load Balancer	Select a load balancer to be associated with the ingress or automatically create a load balancer.	Shared

Parameter	Description	Example
Listener	<ul style="list-style-type: none"> <li>● <b>External Protocol:</b> Select <b>HTTPS</b>.</li> <li>● <b>External Port:</b> specifies the port of the load balancer listener. The default HTTPS port is 443.</li> <li>● <b>Certificate Source:</b> Select <b>ELB server certificate</b>.</li> <li>● <b>Server Certificate:</b> Use a certificate created on ELB. If no certificate is available, go to the ELB console and create one.</li> </ul>	<ul style="list-style-type: none"> <li>● External Protocol: HTTPS</li> <li>● External Port: 443</li> <li>● Certificate Source: ELB server certificate</li> <li>● Server Certificate: cert-test</li> </ul>
Forwarding Policy	<ul style="list-style-type: none"> <li>● <b>Domain Name:</b> Enter an actual domain name to be accessed. If it is left blank, the ingress can be accessed through the IP address. Ensure that the domain name has been registered and licensed. Once a forwarding policy is configured with a domain name specified, you must use the domain name for access.</li> <li>● <b>Path Matching Rule:</b> Select <b>Prefix match</b>, <b>Exact match</b>, or <b>RegEx match</b>.</li> <li>● <b>Path:</b> Enter the path provided by a backend application for external access. The path added must be valid in the backend application, or the forwarding cannot take effect.</li> <li>● <b>Destination Service:</b> Select an existing Service or create a Service. Any Services that do not match the search criteria will be filtered out automatically.</li> <li>● <b>Destination Service Port:</b> Select the access port of the destination Service.</li> </ul>	<ul style="list-style-type: none"> <li>● Domain Name: You do not need to configure this parameter.</li> <li>● Path Matching Rule: Prefix match</li> <li>● Path: /</li> <li>● Destination Service: nginx</li> <li>● Destination Service Port: 80</li> </ul>

**Figure 7-37** Selecting a certificate from the ELB service

The screenshot shows the configuration for an ELB Listener. The 'Certificate Source' is set to 'ELB server certificate', which is highlighted with a red box. Other visible settings include: External Protocol: HTTPS; External Port: 443; Access Control: Not configured; Server Certificate: cert-test; SNI: --Select--; Security Policy: --Select--; Backend Protocol: HTTP.

**Step 4** Click **OK**.

----End

## Using kubectl

To use an ELB certificate, you can specify the `kubernetes.io/elb.tls-certificate-ids` annotation.

**Step 1** Use `kubectl` to access the cluster. For details, see [Connecting to a Cluster Using kubectl](#).

**Step 2** Create a YAML file named `ingress-test.yaml`. The file name can be customized.

```
vi ingress-test.yaml
```

An example YAML file of an ingress associated with an existing load balancer is as follows:

### For clusters of v1.21 or earlier:

```
apiVersion: networking.k8s.io/v1beta1
kind: Ingress
metadata:
  name: ingress-test
  annotations:
    kubernetes.io/ingress.class: cce
    kubernetes.io/elb.port: '443'
    kubernetes.io/elb.id: 0b9a6c4d-bd8b-45cc-bfc8-ff0f9da54e95
    kubernetes.io/elb.class: union
    kubernetes.io/elb.tls-certificate-ids:
058cc023690d48a3867ad69dbe9cd6e5,b98382b1f01c473286653afd1ed9ab63
spec:
  rules:
  - host: ""
    http:
```

```

paths:
- path: '/'
  backend:
    serviceName: <your_service_name> # Replace it with the name of your target Service.
    servicePort: 80
property:
  ingress.beta.kubernetes.io/url-match-mode: STARTS_WITH
    
```

**For clusters of v1.23 or later:**

```

apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: ingress-test
  namespace: default
  annotations:
    kubernetes.io/elb.port: '443'
    kubernetes.io/elb.id: 0b9a6c4d-bd8b-45cc-bfc8-ff0f9da54e95
    kubernetes.io/elb.class: union
    kubernetes.io/elb.tls-certificate-ids:
058cc023690d48a3867ad69dbe9cd6e5,b98382b1f01c473286653afd1ed9ab63
spec:
  rules:
  - host: ""
    http:
      paths:
      - path: '/'
        backend:
          service:
            name: <your_service_name> # Replace it with the name of your target Service.
            port:
              number: 80 # Replace 80 with the port number of your target Service.
          property:
            ingress.beta.kubernetes.io/url-match-mode: STARTS_WITH
            pathType: ImplementationSpecific
          ingressClassName: cce
    
```

**Table 7-58** Key parameters

Parameter	Type	Description
kubernetes.io/elb.tls-certificate-ids	String	<p>ELB certificate IDs, which are separated by comma (,). The list length is greater than or equal to 1. The first ID in the list is the server certificate, and the other IDs are SNI certificates in which a domain name must be contained.</p> <p>If an SNI certificate cannot be found based on the domain name requested by the client, the server certificate will be returned by default.</p> <p>To obtain the certificate, log in to the CCE console, choose <b>Service List &gt; Networking &gt; Elastic Load Balance</b>, and click <b>Certificates</b> in the navigation pane. In the load balancer list, copy the ID under the target certificate name.</p>

**Step 3** Create an ingress.

```
kubectl create -f ingress-test.yaml
```

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

```
ingress/ingress-test created
```

**Step 4** Check the created ingress.

```
kubectl get ingress
```

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

```
NAME      CLASS  HOSTS      ADDRESS      PORTS  AGE
ingress-test  cce    *          121.**.**.**    80,443 10s
```

----End

## Configuring a Same Port for Multiple Ingresses

In a cluster, multiple ingresses can share a listener, allowing them to use the same port on a single load balancer. When two ingresses are set up with HTTPS certificates, the server certificate that is used will be based on the configuration of the earliest ingress.

Starting from v1.21.15-r0, v1.23.14-r0, v1.25.9-r0, v1.27.6-r0, v1.28.4-r0, v1.29.1-r0, and later, the YAML file of ingresses includes **annotation: kubernetes.io/elb.listener-master-ingress**. This annotation specifies the server certificate configured and used by ingresses.

For example, the configurations of two ingresses are as follows:

<b>Ingress Name</b>	ingress1	ingress2
<b>Namespace</b>	default	default
<b>Creation Time</b>	2024-04-01	2024-04-02
<b>Protocol</b>	HTTPS	HTTPS
<b>Load Balancer</b>	elb1	elb1
<b>Port</b>	443	443
<b>kubernetes.io/elb.listener-master-ingress</b>	default/ingress1	default/ingress1

In the configurations of the two ingresses, **annotation: kubernetes.io/elb.listener-master-ingress** is present and the value is **default/ingress1**, indicating that the server certificates that take effect for the two ingresses are the server certificates configured for **ingress1** in the **default** namespace.

 **NOTE**

If ingresses in separate namespaces use the same listener and TLS certificates, the secrets associated with the TLS certificates may not display normal for the later-created ingress due to namespace isolation.

To update a server certificate, modify the certificate in the **ingress1** configuration within the **default** namespace. Once the modification is made, the certificates used by both **ingress1** and **ingress2** will be updated accordingly.

To find the ingress with the active certificate configuration, run the following command:

```
kubectl get ingress -n {namespace} {ingress_name} -oyaml | grep kubernetes.io/elb.listener-master-ingress
```

#### 7.4.2.4.2 Configuring SNI for a LoadBalancer Ingress

An SNI certificate is an extended server certificate that allows the same IP address and port number to provide multiple access domain names for external systems. Different security certificates can be used based on the domain names requested by clients to ensure HTTPS communication security.

When configuring SNI, you need to add a certificate associated with a domain name. The client submits the requested domain name information when initiating an SSL handshake request. After receiving the SSL request, the load balancer searches for the certificate based on the domain name. If the certificate is found, the load balancer will return it to the client. If the certificate is not found, the load balancer will return the default server certificate.

You are allowed to use either of the following ways to configure an ingress certificate in a cluster:

- **Configuring SNI Using a TLS Certificate:** You need to first import a certificate to a Secret. CCE will then automatically handle the certificate configurations on the ELB console and give a name to the certificate (started with `k8s_plb_default`). This certificate, which is generated by CCE, cannot be modified or deleted from the ELB console.
- **Configuring SNI Using an ELB Certificate:** You are allowed to directly use certificates created on the ELB console. There is no need to manually configure the cluster Secrets, and you can modify the certificates on the ELB console.

#### Prerequisites

- An available workload has been deployed in the cluster for external access. If no workload is available, deploy a workload by referring to [Creating a Deployment](#), [Creating a StatefulSet](#), or [Creating a DaemonSet](#).
- A Service for external access has been configured for the workload. [Services Supported by LoadBalancer Ingresses](#) lists the Service types supported by LoadBalancer ingresses.
- You can obtain a trusted certificate from a certificate provider.

#### Configuring SNI Using a TLS Certificate

You can configure SNI through a TLS certificate using either the CCE console or kubectl.

##### NOTE

- The **SNI** option is available only when HTTPS is used.
- Only one domain name can be specified for each SNI certificate. Wildcard-domain certificates are supported.
- Security policy (`kubernetes.io/elb.tls-ciphers-policy`) is supported only in clusters of v1.17.11 or later.

## Using the CCE Console

**Step 1** Log in to the CCE console and click the cluster name to access the cluster console.

**Step 2** Choose **Services & Ingresses** in the navigation pane, click the **Ingresses** tab, and click **Create Ingress** in the upper right corner.

**Step 3** Configure ingress parameters.

 **NOTE**

This example explains only key parameters for configuring SNI certificates. You can configure other parameters as required. For details, see [Creating a LoadBalancer Ingress on the Console](#).

**Table 7-59** Key parameters

Parameter	Description	Example
Name	Enter an ingress name.	ingress-test
Load Balancer	Select a load balancer to be associated with the ingress or automatically create a load balancer.	Shared
Listener	<ul style="list-style-type: none"> <li>● <b>External Protocol:</b> Select <b>HTTPS</b> when configuring a certificate for an ingress.</li> <li>● <b>External Port:</b> specifies the port of the load balancer listener. The default HTTPS port is 443.</li> <li>● <b>Certificate Source:</b> Select <b>TLS secret</b>.</li> <li>● <b>Server Certificate:</b> <b>kubernetes.io/tls</b> and <b>IngressTLS</b> are supported. If no certificate is available, you can create a TLS certificate. For details about the configuration parameters, see <a href="#">Creating a Secret</a>.</li> <li>● <b>SNI:</b> Enter a domain name and select a certificate. The SNI certificate must contain the domain name information. SNI certificates support two TLS secret types: <b>kubernetes.io/tls</b> and <b>IngressTLS</b>.</li> </ul>	<ul style="list-style-type: none"> <li>● External Protocol: HTTPS</li> <li>● External Port: 443</li> <li>● Certificate Source: TLS secret</li> <li>● Server Certificate: test</li> <li>● SNI: <ul style="list-style-type: none"> <li>– Domain Name: example.com</li> <li>– Certificate: example-test</li> </ul> </li> </ul>

Parameter	Description	Example
Forwarding Policy	<ul style="list-style-type: none"> <li>• <b>Domain Name:</b> Enter an actual domain name to be accessed. If it is left blank, the ingress can be accessed through the IP address. Ensure that the domain name has been registered and licensed. Once a forwarding policy is configured with a domain name specified, you must use the domain name for access.</li> <li>• <b>Path Matching Rule:</b> Select <b>Prefix match</b>, <b>Exact match</b>, or <b>RegEx match</b>.</li> <li>• <b>Path:</b> Enter the path provided by a backend application for external access. The path added must be valid in the backend application, or the forwarding cannot take effect.</li> <li>• <b>Destination Service:</b> Select an existing Service or create a Service. Any Services that do not match the search criteria will be filtered out automatically.</li> <li>• <b>Destination Service Port:</b> Select the access port of the destination Service.</li> </ul>	<ul style="list-style-type: none"> <li>• Domain Name: example.com</li> <li>• Path Matching Rule: Prefix match</li> <li>• Path: /</li> <li>• Destination Service: nginx</li> <li>• Destination Service Port: 80</li> </ul>

**Figure 7-38** Configuring SNI using a TLS certificate

Listener

External Protocol  HTTP  HTTPS

External Port

Access Control

Certificate Source  ELB server certificate  TLS secret

Server Certificate  [Create TLS Secret](#)

SNI	Domain Name	Certificate	Operation
	<input type="text" value="example.com"/>	<input type="text" value="example-test"/>	<a href="#">Delete</a>

+

Security Policy

Backend Protocol  HTTP  HTTPS

Advanced Options [+ Add advanced options](#)



**Step 4** Click **OK**.

----End

## Using kubectl

In this example, the **sni-test-secret** SNI certificate is used as an example. The specified domain name must be the same as that of the SNI certificate.

**Step 1** Use kubectl to access the cluster. For details, see [Connecting to a Cluster Using kubectl](#).

**Step 2** Create a YAML file named **ingress-test.yaml**. The file name can be customized.

```
vi ingress-test.yaml
```

An example YAML file of an ingress associated with an automatically created load balancer is as follows:

### For clusters of v1.21 or earlier:

```
apiVersion: networking.k8s.io/v1beta1
kind: Ingress
metadata:
  name: ingress-test
  annotations:
    kubernetes.io/elb.class: performance
    kubernetes.io/ingress.class: cce
    kubernetes.io/elb.port: '443'
    kubernetes.io/elb.autocreate:
      '{
        "type": "public",
        "bandwidth_name": "cce-bandwidth-*****",
        "bandwidth_chargemode": "bandwidth",
        "bandwidth_size": 5,
        "bandwidth_sharetype": "PER",
        "eip_type": "5_bgp",
        "available_zone": [
          "eu-west-101a"
        ],
        "elb_virsubnet_ids": ["b4bf8152-6c36-4c3b-9f74-2229f8e640c9"],
        "l7_flavor_name": "L7_flavor.elb.s1.small"
      }'
    kubernetes.io/elb.tls-ciphers-policy: tls-1-2
spec:
  tls:
  - secretName: ingress-test-secret
  - hosts:
    - example.com # Domain name specified when a certificate is issued
    secretName: sni-test-secret #SNI certificate
  rules:
  - host: example.com #The domain name must be the same as the value of hosts in the tls field.
    http:
      paths:
      - path: '/'
        backend:
          serviceName: <your_service_name> # Replace it with the name of your target Service.
          servicePort: 80
        property:
          ingress.beta.kubernetes.io/url-match-mode: STARTS_WITH
```

### For clusters of v1.23 or later:

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: ingress-test
  annotations:
    kubernetes.io/elb.class: performance
```

```
kubernetes.io/elb.port: '443'
kubernetes.io/elb.autocreate:
  '{
    "type": "public",
    "bandwidth_name": "cce-bandwidth-*****",
    "bandwidth_chargemode": "bandwidth",
    "bandwidth_size": 5,
    "bandwidth_sharetype": "PER",
    "eip_type": "5_bgp",
    "available_zone": [
      "eu-west-101a"
    ],
    "elb_virsubnet_ids": ["b4bf8152-6c36-4c3b-9f74-2229f8e640c9"],
    "l7_flavor_name": "L7_flavor.elb.s1.small"
  }'
kubernetes.io/elb.tls-ciphers-policy: tls-1-2
spec:
  tls:
    - secretName: ingress-test-secret
  - hosts:
    - example.com # Domain name specified when a certificate is issued
    secretName: sni-test-secret #SNI certificate
  rules:
  - host: example.com #The domain name must be the same as the value of hosts in the tls field.
    http:
      paths:
      - path: '/'
        backend:
          service:
            name: <your_service_name> # Replace it with the name of your target Service.
            port:
              number: 80 # Replace 80 with the port number of your target Service.
          property:
            ingress.beta.kubernetes.io/url-match-mode: STARTS_WITH
            pathType: ImplementationSpecific
          ingressClassName: cce
```

### Step 3 Create an ingress.

```
kubectl create -f ingress-test.yaml
```

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

```
ingress/ingress-test created
```

### Step 4 Check the created ingress.

```
kubectl get ingress
```

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

NAME	CLASS	HOSTS	ADDRESS	PORTS	AGE
ingress-test	cce	example.com	121.**.**.*	80,443	10s

### Step 5 Use HTTPS to access the ingress. $\${ELB\_IP}$ specifies the IP address accessed by the target ingress.

```
curl -H "Host:example.com" -k https:// $\${ELB\_IP}$ :443
```

If the ingress can be accessed, the certificate is configured.

----End

## Configuring SNI Using an ELB Certificate

You can configure SNI using an ELB certificate for an ingress through either the CCE console or kubectl.

 **NOTE**

- If both an ELB certificate and a TLS certificate are specified for the same ingress, the ingress will use the ELB certificate.
- CCE does not check whether an ELB certificate is valid. It only checks whether the certificate is present.
- Only ingresses in clusters of v1.19.16-r2, v1.21.5-r0, v1.23.3-r0, or later support ELB certificates.

## Using the CCE Console

**Step 1** Log in to the CCE console and click the cluster name to access the cluster console.

**Step 2** Choose **Services & Ingresses** in the navigation pane, click the **Ingresses** tab, and click **Create Ingress** in the upper right corner.

**Step 3** Configure ingress parameters.

 **NOTE**

This example explains only key parameters for configuring SNI certificates. You can configure other parameters as required. For details, see [Creating a LoadBalancer Ingress on the Console](#).

**Table 7-60** Key parameters

Parameter	Description	Example
Name	Enter an ingress name.	ingress-test
Load Balancer	Select a load balancer to be associated with the ingress or automatically create a load balancer.	Shared
Listener	<ul style="list-style-type: none"> <li>• <b>External Protocol:</b> Select <b>HTTPS</b>.</li> <li>• <b>External Port:</b> specifies the port of the load balancer listener. The default HTTPS port is 443.</li> <li>• <b>Certificate Source:</b> Select <b>ELB server certificate</b>.</li> <li>• <b>Server Certificate:</b> Use a certificate created on ELB. If no certificate is available, go to the ELB console and create one.</li> <li>• <b>SNI:</b> Select the corresponding SNI certificate, which must contain the domain name information. If no certificate is available, go to the ELB console and create one.</li> </ul>	<ul style="list-style-type: none"> <li>• External Protocol: HTTPS</li> <li>• External Port: 443</li> <li>• Certificate Source: ELB server certificate</li> <li>• Server Certificate: cert-test</li> <li>• SNI: cert-example</li> </ul>

Parameter	Description	Example
Forwarding Policy	<ul style="list-style-type: none"> <li>• <b>Domain Name:</b> Enter an actual domain name to be accessed. If it is left blank, the ingress can be accessed through the IP address. Ensure that the domain name has been registered and licensed. Once a forwarding policy is configured with a domain name specified, you must use the domain name for access.</li> <li>• <b>Path Matching Rule:</b> Select <b>Prefix match</b>, <b>Exact match</b>, or <b>RegEx match</b>.</li> <li>• <b>Path:</b> Enter the path provided by a backend application for external access. The path added must be valid in the backend application, or the forwarding cannot take effect.</li> <li>• <b>Destination Service:</b> Select an existing Service or create a Service. Any Services that do not match the search criteria will be filtered out automatically.</li> <li>• <b>Destination Service Port:</b> Select the access port of the destination Service.</li> </ul>	<ul style="list-style-type: none"> <li>• Domain Name: You do not need to configure this parameter.</li> <li>• Path Matching Rule: Prefix match</li> <li>• Path: /</li> <li>• Destination Service: nginx</li> <li>• Destination Service Port: 80</li> </ul>

**Figure 7-39** Configuring SNI using a certificate from the ELB service

Listener

External Protocol  HTTP  HTTPS

External Port

Access Control

Certificate Source  ELB server certificate  TLS secret

Server Certificate  [View ELB Certificate](#)

**SNI**  [?](#)

Security Policy  [?](#)

Backend Protocol  HTTP  HTTPS

Advanced Options [+ Add advanced options](#)

**Step 4** Click **OK**.

----End

## Using kubectl

To use an ELB certificate for an ingress, you can specify the **kubernetes.io/elb.tls-certificate-ids** annotation.

**Step 1** Use kubectl to access the cluster. For details, see [Connecting to a Cluster Using kubectl](#).

**Step 2** Create a YAML file named **ingress-test.yaml**. The file name can be customized.

```
vi ingress-test.yaml
```

An example YAML file of an ingress associated with an existing load balancer is as follows:

### For clusters of v1.21 or earlier:

```
apiVersion: networking.k8s.io/v1beta1
kind: Ingress
metadata:
  name: ingress-test
  annotations:
    kubernetes.io/ingress.class: cce
    kubernetes.io/elb.port: '443'
    kubernetes.io/elb.id: 0b9a6c4d-bd8b-45cc-bfc8-ff0f9da54e95
    kubernetes.io/elb.class: union
    kubernetes.io/elb.tls-certificate-ids:
058cc023690d48a3867ad69dbe9cd6e5,b98382b1f01c473286653afd1ed9ab63
spec:
  rules:
  - host: ""
    http:
      paths:
      - path: '/'
        backend:
          serviceName: <your_service_name> # Replace it with the name of your target Service.
          servicePort: 80
  property:
    ingress.beta.kubernetes.io/url-match-mode: STARTS_WITH
```

### For clusters of v1.23 or later:

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: ingress-test
  namespace: default
  annotations:
    kubernetes.io/elb.port: '443'
    kubernetes.io/elb.id: 0b9a6c4d-bd8b-45cc-bfc8-ff0f9da54e95
    kubernetes.io/elb.class: union
    kubernetes.io/elb.tls-certificate-ids:
058cc023690d48a3867ad69dbe9cd6e5,b98382b1f01c473286653afd1ed9ab63
spec:
  rules:
  - host: ""
    http:
      paths:
      - path: '/'
        backend:
          service:
            name: <your_service_name> # Replace it with the name of your target Service.
            port:
              number: 80 # Replace 80 with the port number of your target Service.
```

```
property:
  ingress.beta.kubernetes.io/url-match-mode: STARTS_WITH
  pathType: ImplementationSpecific
ingressClassName: cce
```

**Table 7-61** Key parameters

Parameter	Type	Description
kubernetes.io/elb.tls-certificate-ids	String	<p>ELB certificate IDs, which are separated by comma (.). The list length is greater than or equal to 1. The first ID in the list is the server certificate, and the other IDs are SNI certificates in which a domain name must be contained.</p> <p>If an SNI certificate cannot be found based on the domain name requested by the client, the server certificate will be returned by default.</p> <p>To obtain the certificate, log in to the CCE console, choose <b>Service List &gt; Networking &gt; Elastic Load Balance</b>, and click <b>Certificates</b> in the navigation pane. In the load balancer list, copy the ID under the target certificate name.</p>

**Step 3** Create an ingress.

```
kubectl create -f ingress-test.yaml
```

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

```
ingress/ingress-test created
```

**Step 4** Check the created ingress.

```
kubectl get ingress
```

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

```
NAME      CLASS  HOSTS      ADDRESS      PORTS  AGE
ingress-test  cce    *          121.**.**.**  80,443  10s
```

**Step 5** Use HTTPS to access the ingress.  $\{ELB\_IP\}$  specifies the IP address accessed by the target ingress.

```
curl -H "Host:example.com" -k https://{ELB_IP}:443
```

If the ingress can be accessed, the certificate is configured.

----End

### 7.4.2.4.3 Configuring Multiple Forwarding Policies for a LoadBalancer Ingress

An ingress can route requests to multiple backend Services based on different matching policies. For example, requests can be routed to three different backend Services separately by accessing **www.example.com/foo**, **www.example.com/bar**, and **foo.example.com/**.

**NOTICE**

The URL registered in an ingress forwarding policy must be the same as the URL used to access the backend Service. Otherwise, a 404 error will be returned.

For example, the default access URL of the Nginx application is `/usr/share/nginx/html`. When adding `/test` to the ingress forwarding policy, ensure the access URL of your Nginx application contains `/usr/share/nginx/html/test`. Otherwise, error 404 will be returned.

## Prerequisites

- An available workload has been deployed in the cluster for external access. If no workload is available, deploy a workload by referring to [Creating a Deployment](#), [Creating a StatefulSet](#), or [Creating a DaemonSet](#).
- A Service for external access has been configured for the workload. [Services Supported by LoadBalancer Ingresses](#) lists the Service types supported by LoadBalancer ingresses.

## Routing an Ingress to Multiple Services

You can an ingress route to multiple services using either the CCE console or kubectl.

## Using the CCE Console

- Step 1** Log in to the CCE console and click the cluster name to access the cluster console.
- Step 2** Choose **Services & Ingresses** in the navigation pane, click the **Ingresses** tab, and click **Create Ingress** in the upper right corner.
- Step 3** Configure ingress parameters.

 **NOTE**

This example explains only key parameters for configuring forwarding policies. You can configure other parameters as required. For details, see [Creating a LoadBalancer Ingress on the Console](#).

**Table 7-62** Key parameters

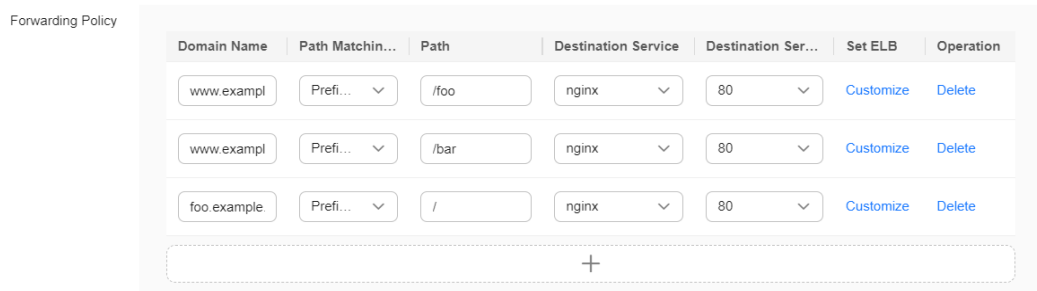
Parameter	Description	Example
Name	Enter an ingress name.	ingress-test
Load Balancer	Select a load balancer to be associated with the ingress or automatically create a load balancer.	Shared
Listener	<ul style="list-style-type: none"> <li>• <b>External Protocol:</b> <b>HTTP</b> and <b>HTTPS</b> are available.</li> <li>• <b>External Port:</b> specifies the port of the load balancer listener.</li> </ul>	<ul style="list-style-type: none"> <li>• External Protocol: HTTP</li> <li>• External Port: 80</li> </ul>

Parameter	Description	Example
Forwarding Policy	<ul style="list-style-type: none"> <li>● <b>Domain Name:</b> Enter an actual domain name to be accessed. If it is left blank, the ingress can be accessed through the IP address. Ensure that the domain name has been registered and licensed. Once a forwarding policy is configured with a domain name specified, you must use the domain name for access.</li> <li>● <b>Path Matching Rule:</b> Select <b>Prefix match</b>, <b>Exact match</b>, or <b>RegEx match</b>.</li> <li>● <b>Path:</b> Enter the path provided by a backend application for external access. The path added must be valid in the backend application, or the forwarding cannot take effect.</li> <li>● <b>Destination Service:</b> Select an existing Service or create a Service. Any Services that do not match the search criteria will be filtered out automatically.</li> <li>● <b>Destination Service Port:</b> Select the access port of the destination Service.</li> </ul>	<p>Forwarding rule 1:</p> <ul style="list-style-type: none"> <li>● Domain Name: www.example.com</li> <li>● Path Matching Rule: Prefix match</li> <li>● Path: /foo</li> <li>● Destination Service: nginx</li> <li>● Destination Service Port: 80</li> </ul> <p>Forwarding rule 2:</p> <ul style="list-style-type: none"> <li>● Domain Name: www.example.com</li> <li>● Path Matching Rule: Prefix match</li> <li>● Path: /bar</li> <li>● Destination Service: nginx</li> <li>● Destination Service Port: 80</li> </ul> <p>Forwarding rule 3:</p> <ul style="list-style-type: none"> <li>● Domain Name: foo.example.com</li> <li>● Path Matching Rule: Prefix match</li> <li>● Path: /</li> <li>● Destination Service: nginx</li> </ul>



Parameter	Description	Example
		<ul style="list-style-type: none"> <li>Destination Service Port: 80</li> </ul>

**Figure 7-40** Routing an ingress to multiple Services



**Step 4** Click **OK**.

----End

## Using kubectl

**Step 1** Use `kubectl` to access the cluster. For details, see [Connecting to a Cluster Using kubectl](#).

**Step 2** Create a YAML file named **ingress-test.yaml**. The file name can be customized.

```
vi ingress-test.yaml
```

An example YAML file of an ingress associated with an existing load balancer is as follows:

**For clusters of v1.23 or later:**

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: ingress-test
  annotations:
    kubernetes.io/elb.id: <your_elb_id> #Replace it with the ID of your existing load balancer.
    kubernetes.io/elb.class: performance # Load balancer type
    kubernetes.io/elb.port: '80'
spec:
  rules:
  - host: 'www.example.com'
    http:
      paths:
      - path: '/foo'
        backend:
          service:
            name: <your_service_name> # Replace it with the name of your target Service.
            port:
              number: 80 # Replace 80 with the port number of your target Service.
          property:
            ingress.beta.kubernetes.io/url-match-mode: STARTS_WITH
            pathType: ImplementationSpecific
      - path: '/bar'
        backend:
```

```

service:
  name: <your_service_name> # Replace it with the name of your target Service.
  port:
    number: 80 # Replace 80 with the port number of your target Service.
  property:
    ingress.beta.kubernetes.io/url-match-mode: STARTS_WITH
    pathType: ImplementationSpecific
- host: 'foo.example.com'
  http:
    paths:
    - path: '/'
      backend:
        service:
          name: <your_service_name> # Replace it with the name of your target Service.
          port:
            number: 80 # Replace 80 with the port number of your target Service.
        property:
          ingress.beta.kubernetes.io/url-match-mode: STARTS_WITH
          pathType: ImplementationSpecific
    ingressClassName: cce

```

### For clusters of v1.21 or earlier:

```

apiVersion: networking.k8s.io/v1beta1
kind: Ingress
metadata:
  name: ingress-test
  annotations:
    kubernetes.io/ingress.class: cce
    kubernetes.io/elb.port: '80'
    kubernetes.io/elb.id: <your_elb_id> #Replace it with the ID of your existing load balancer.
    kubernetes.io/elb.class: performance # Load balancer type
spec:
  rules:
  - host: 'www.example.com'
    http:
      paths:
      - path: '/foo'
        backend:
          serviceName: <your_service_name> # Replace it with the name of your target Service.
          servicePort: 80
        property:
          ingress.beta.kubernetes.io/url-match-mode: STARTS_WITH
      - path: '/bar'
        backend:
          serviceName: <your_service_name> # Replace it with the name of your target Service.
          servicePort: 80
        property:
          ingress.beta.kubernetes.io/url-match-mode: STARTS_WITH
  - host: 'foo.example.com'
    http:
      paths:
      - path: '/'
        backend:
          serviceName: <your_service_name> # Replace it with the name of your target Service.
          servicePort: 80
        property:
          ingress.beta.kubernetes.io/url-match-mode: STARTS_WITH

```

### Step 3 Create an ingress.

```
kubectl create -f ingress-test.yaml
```

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

```
ingress/ingress-test created
```

### Step 4 Check the created ingress.

```
kubectl get ingress
```

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

NAME	CLASS	HOSTS	ADDRESS	PORTS	AGE
ingress-test	cce	www.example.com,foo.example.com	121.**.**.*	80	10s

----End

#### 7.4.2.4.4 Configuring HTTP/2 for a LoadBalancer Ingress

Ingresses can use HTTP/2 to expose Services. Connections from the load balancer to your application use HTTP/1.x by default. If your application is capable of receiving HTTP/2 requests, you can refer to [Using the CCE Console](#) and [Using kubectl](#) to enable the use of HTTP/2.

### Prerequisites

- A CCE standard cluster is available, and the cluster version meets the following requirements:
  - v1.23: v1.23.13-r0 or later
  - v1.25: v1.25.8-r0 or later
  - v1.28: v1.28.3-r0 or later
  - Other clusters of later versions
- An available workload has been deployed in the cluster for external access. If no workload is available, deploy a workload by referring to [Creating a Deployment](#), [Creating a StatefulSet](#), or [Creating a DaemonSet](#).
- A Service for external access has been configured for the workload. [Services Supported by LoadBalancer Ingresses](#) lists the Service types supported by LoadBalancer ingresses.
- You can obtain a trusted certificate from a certificate provider.

### Notes and Constraints

- Only an HTTPS-compliant load balancer supports HTTP/2.
- After HTTP/2 is configured, if you delete the advanced configuration for enabling HTTP/2 on the CCE console or delete the target annotation from the YAML file, HTTP/2 will be disabled on the ELB.

## Configuring HTTP/2 for an Ingress

You can configure HTTP/2 for an ingress using either the CCE console or kubectl.

### Using the CCE Console

- Step 1** Log in to the CCE console and click the cluster name to access the cluster console.
- Step 2** Choose **Services & Ingresses** in the navigation pane, click the **Ingresses** tab, and click **Create Ingress** in the upper right corner.
- Step 3** Configure ingress parameters.

#### NOTE

This example explains only key parameters for configuring HTTP/2. You can configure other parameters as required. For details, see [Creating a LoadBalancer Ingress on the Console](#).

**Table 7-63** Key parameters

Parameter	Description	Example
Name	Enter an ingress name.	ingress-test
Load Balancer	Select a load balancer to be associated with the ingress or automatically create a load balancer.	Shared
Listener	<ul style="list-style-type: none"> <li>● <b>External Protocol:</b> Select <b>HTTPS</b>.</li> <li>● <b>External Port:</b> specifies the port of the load balancer listener. The default HTTPS port is 443.</li> <li>● <b>Certificate Source:</b> Select <b>ELB server certificate</b>.</li> <li>● <b>Server Certificate:</b> Use a certificate created on ELB. If no certificate is available, go to the ELB console and create one.</li> <li>● <b>Advanced Options:</b> Add advanced configurations, select <b>HTTP2</b>, and set its status to <b>Enable</b>.</li> </ul>	<ul style="list-style-type: none"> <li>● External Protocol: HTTPS</li> <li>● External Port: 443</li> <li>● Certificate Source: ELB server certificate</li> <li>● Server Certificate: cert-test</li> <li>● Advanced Options: HTTP2 enabled</li> </ul>
Forwarding Policy	<ul style="list-style-type: none"> <li>● <b>Domain Name:</b> Enter an actual domain name to be accessed. If it is left blank, the ingress can be accessed through the IP address. Ensure that the domain name has been registered and licensed. Once a forwarding policy is configured with a domain name specified, you must use the domain name for access.</li> <li>● <b>Path Matching Rule:</b> Select <b>Prefix match</b>, <b>Exact match</b>, or <b>RegEx match</b>.</li> <li>● <b>Path:</b> Enter the path provided by a backend application for external access. The path added must be valid in the backend application, or the forwarding cannot take effect.</li> <li>● <b>Destination Service:</b> Select an existing Service or create a Service. Any Services that do not match the search criteria will be filtered out automatically.</li> <li>● <b>Destination Service Port:</b> Select the access port of the destination Service.</li> </ul>	<ul style="list-style-type: none"> <li>● Domain Name: You do not need to configure this parameter.</li> <li>● Path Matching Rule: Prefix match</li> <li>● Path: /</li> <li>● Destination Service: nginx</li> <li>● Destination Service Port: 80</li> </ul>

**Figure 7-41** Configuring HTTP/2 for an ingress

The screenshot shows the configuration interface for an Ingress Listener. The 'External Protocol' is set to 'HTTPS' and the 'Backend Protocol' is set to 'HTTP'. The 'Advanced Options' section is highlighted with a red box, showing 'HTTP2' selected in the dropdown and 'Enable' checked in the checkbox. Other fields include 'External Port' (443), 'Access Control' (Not configured), 'Certificate Source' (ELB server certificate), 'Server Certificate' (cert-test), 'SNI' (--Select--), and 'Security Policy' (--Select--).

**Step 4** Click **OK**.

----End

## Using kubectl

**Step 1** Use kubectl to access the cluster. For details, see [Connecting to a Cluster Using kubectl](#).

**Step 2** Create a YAML file named **ingress-test.yaml**. The file name can be customized.

```
vi ingress-test.yaml
```

An example YAML file of an ingress associated with an existing load balancer is as follows:

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: ingress-test
annotations:
  kubernetes.io/elb.id: <your_elb_id> # Replace it with the ID of your existing load balancer.
  kubernetes.io/elb.ip: <your_elb_ip> # Replace it with the IP of your existing load balancer.
  kubernetes.io/elb.port: '443'
  kubernetes.io/elb.http2-enable: 'true' # Enable HTTP/2.
spec:
  tls:
  - secretName: ingress-test-secret
  rules:
  - host: ""
    http:
      paths:
      - path: '/'
        backend:
          service:
            name: <your_service_name> # Replace it with the name of your target Service.
            port:
              number: 80 # Replace 80 with the port number of your target Service.
          property:
            ingress.beta.kubernetes.io/url-match-mode: STARTS_WITH
            pathType: ImplementationSpecific
  ingressClassName: cce
```

**Table 7-64** HTTP/2 parameters

Parameter	Mandatory	Type	Description
kubernetes.io/elb.http2-enable	No	String	Whether HTTP/2 is enabled. Request forwarding using HTTP/2 improves the access performance between your application and the load balancer. However, the load balancer still uses HTTP/1.x to forward requests to the backend server. Options: <ul style="list-style-type: none"> <li><b>true</b>: enabled</li> <li><b>false</b>: disabled (default value)</li> </ul> Note: <b>HTTP/2 can be enabled or disabled only when the listener uses HTTPS</b> . This parameter is invalid and defaults to <b>false</b> when the listener protocol is HTTP.

**Step 3** Create an ingress.

```
kubectl create -f ingress-test.yaml
```

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

```
ingress/ingress-test created
```

**Step 4** Check the created ingress.

```
kubectl get ingress
```

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

```
NAME      CLASS  HOSTS      ADDRESS      PORTS  AGE
ingress-test  cce    *          121.**.**.**    80,443  10s
```

----End

### 7.4.2.4.5 Configuring HTTPS Backend Services for a LoadBalancer Ingress

Ingresses can interconnect with backend services of different protocols. By default, the backend proxy channel of an ingress is HTTP-compliant. To create an HTTPS channel, add the following configuration to the **annotations** field:

```
kubernetes.io/elb.pool-protocol: https
```

### Prerequisites

- A CCE standard cluster is available, and the cluster version meets the following requirements:
  - v1.23: v1.23.8-r0 or later
  - v1.25: v1.25.3-r0 or later
  - Other clusters of later versions
- An available workload has been deployed in the cluster for external access. If no workload is available, deploy a workload by referring to [Creating a Deployment](#), [Creating a StatefulSet](#), or [Creating a DaemonSet](#).

- A Service for external access has been configured for the workload. [Services Supported by LoadBalancer Ingresses](#) lists the Service types supported by LoadBalancer ingresses.
- You can obtain a trusted certificate from a certificate provider.

## Notes and Constraints

- Ingresses can interconnect with HTTPS backend services only when dedicated load balancers are used.
- When an ingress interconnects with an HTTPS backend service, the ingress protocol must be HTTPS.

## Configuring an HTTPS Backend Service

You can configure an HTTPS backend Service for an ingress using either the CCE console or kubectl.

## Using the CCE Console

- Step 1** Log in to the CCE console and click the cluster name to access the cluster console.
- Step 2** Choose **Services & Ingresses** in the navigation pane, click the **Ingresses** tab, and click **Create Ingress** in the upper right corner.
- Step 3** Configure ingress parameters.

 **NOTE**

This example explains only key parameters for configuring HTTPS backend Services. You can configure other parameters as required. For details, see [Creating a LoadBalancer Ingress on the Console](#).

**Table 7-65** Key parameters

Parameter	Description	Example
Name	Enter an ingress name.	ingress-test
Load Balancer	Select a load balancer to be associated with the ingress or automatically create a load balancer. In this example, only dedicated load balancers are supported.	Dedicated

Parameter	Description	Example
Listener	<ul style="list-style-type: none"> <li>● <b>External Protocol:</b> Select <b>HTTPS</b> for an HTTPS backend Service for an ingress.</li> <li>● <b>External Port:</b> specifies the port of the load balancer listener. The default HTTPS port is 443.</li> <li>● <b>Certificate Source:</b> Select <b>ELB server certificate</b>.</li> <li>● <b>Server Certificate:</b> Use a certificate created on ELB. If no certificate is available, go to the ELB console and create one.</li> <li>● <b>Backend Protocol:</b> Select <b>HTTPS</b>.</li> </ul>	<ul style="list-style-type: none"> <li>● External Protocol: HTTPS</li> <li>● External Port: 443</li> <li>● Certificate Source: ELB server certificate</li> <li>● Server Certificate: cert-test</li> <li>● Backend Protocol: HTTPS</li> </ul>
Forwarding Policy	<ul style="list-style-type: none"> <li>● <b>Domain Name:</b> Enter an actual domain name to be accessed. If it is left blank, the ingress can be accessed through the IP address. Ensure that the domain name has been registered and licensed. Once a forwarding policy is configured with a domain name specified, you must use the domain name for access.</li> <li>● <b>Path Matching Rule:</b> Select <b>Prefix match</b>, <b>Exact match</b>, or <b>RegEx match</b>.</li> <li>● <b>Path:</b> Enter the path provided by a backend application for external access. The path added must be valid in the backend application, or the forwarding cannot take effect.</li> <li>● <b>Destination Service:</b> Select an existing Service or create a Service. Any Services that do not match the search criteria will be filtered out automatically.</li> <li>● <b>Destination Service Port:</b> Select the access port of the destination Service.</li> </ul>	<ul style="list-style-type: none"> <li>● Domain Name: You do not need to configure this parameter.</li> <li>● Path Matching Rule: Prefix match</li> <li>● Path: /</li> <li>● Destination Service: nginx</li> <li>● Destination Service Port: 80</li> </ul>



**Figure 7-42** Configuring an HTTPS backend Service

The screenshot shows the 'Listener' configuration interface. The 'External Protocol' is set to 'HTTPS' (highlighted with a red box). Below it, 'External Port' is 443, 'Access Control' is 'Not configured', 'Certificate Source' is 'ELB server certificate', 'Server Certificate' is 'cert-test', 'SNI' is '--Select--', and 'Security Policy' is '--Select--'. At the bottom, the 'Backend Protocol' is set to 'HTTPS' (highlighted with a red box). There are also 'Advanced Options' and a link to '+ Add advanced options'.

**Step 4** Click **OK**.

----End

## Using kubectl

**Step 1** Use `kubectl` to access the cluster. For details, see [Connecting to a Cluster Using kubectl](#).

**Step 2** Create a YAML file named **ingress-test.yaml**. The file name can be customized.  
vi ingress-test.yaml

An example YAML file of an ingress associated with an existing load balancer is as follows:

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: ingress-test
  namespace: default
  annotations:
    kubernetes.io/elb.port: '443'
    kubernetes.io/elb.id: <your_elb_id> # In this example, an existing dedicated load balancer is used.
    kubernetes.io/elb.class: performance
    kubernetes.io/elb.pool-protocol: https # Interconnected HTTPS backend service
    kubernetes.io/elb.tls-ciphers-policy: tls-1-2
spec:
  tls:
    - secretName: ingress-test-secret
  rules:
    - host: ""
      http:
        paths:
          - path: '/'
            backend:
              service:
                name: <your_service_name> # Replace it with the name of your target Service.
                port:
                  number: 80
            property:
              ingress.beta.kubernetes.io/url-match-mode: STARTS_WITH
              pathType: ImplementationSpecific
  ingressClassName: cce
```

**Step 3** Create an ingress.

```
kubectl create -f ingress-test.yaml
```

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

```
ingress/ingress-test created
```

**Step 4** Check the created ingress.

```
kubectl get ingress
```

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

```
NAME      CLASS  HOSTS      ADDRESS      PORTS  AGE
ingress-test  cce    *          121.**.**.**    80,443 10s
```

----End

### 7.4.2.4.6 Configuring Timeout for a LoadBalancer Ingress

LoadBalancer ingresses support the following timeout settings:

- Idle timeout setting for client connections: maximum duration for keeping a connection when no client request is received. If no request is received during this period, the load balancer closes the connection and establishes a new one with the client when the next request arrives.
- Timeout for waiting for a request from a client: If the client fails to send a request header to the load balancer during the timeout duration or the interval for sending body data exceeds a specified period, the load balancer will release the connection.
- Timeout setting for waiting for a response from a backend server: If the backend server fails to respond during the timeout duration, the load balancer will stop waiting and return HTTP 504 Gateway Timeout to the client.

 **NOTE**

If you delete the timeout configuration when updating an ingress, the timeout configuration of the existing listener will be reset to the default value.

### Prerequisites

- A CCE standard cluster is available. The table below shows which cluster versions support the timeout configuration.

Timeout Type	Load Balancer Type	Supported Cluster Version
Idle Timeout	Dedicated	<ul style="list-style-type: none"> <li>• v1.19: v1.19.16-r30 or later</li> <li>• v1.21: v1.21.10-r10 or later</li> <li>• v1.23: v1.23.8-r10 or later</li> <li>• v1.25: v1.25.3-r10 or later</li> <li>• Other clusters of later versions</li> </ul>
Request Timeout	Dedicated	
Response Timeout	Dedicated	

- An available workload has been deployed in the cluster for external access. If no workload is available, deploy a workload by referring to [Creating a Deployment](#), [Creating a StatefulSet](#), or [Creating a DaemonSet](#).

- A Service for external access has been configured for the workload. [Services Supported by LoadBalancer Ingresses](#) lists the Service types supported by LoadBalancer ingresses.

## Configuring Timeout

You can configure timeout for an ingress using either the CCE console or kubectl.

## Using the CCE Console

- 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 **Services & Ingresses**. Click the **Ingresses** tab and click **Create Ingress** in the upper right corner.
- Step 3** Configure ingress parameters.

 **NOTE**

This example explains only key parameters for configuring timeout. You can configure other parameters as required. For details, see [Creating a LoadBalancer Ingress on the Console](#).

**Table 7-66** Key parameters

Parameter	Description	Example
Name	Enter an ingress name.	ingress-test
Load Balancer	Select a load balancer to be associated with the ingress or automatically create a load balancer. A load balancer can be dedicated or shared.	Dedicated

Parameter	Description	Example
Listener	<ul style="list-style-type: none"> <li>● <b>External Protocol:</b> HTTP and HTTPS are available.</li> <li>● <b>External Port:</b> specifies the port of the load balancer listener.</li> <li>● <b>Advanced Options</b> <ul style="list-style-type: none"> <li>- <b>Idle Timeout (s):</b> specifies the idle timeout of a client connection. If there are no requests reaching the load balancer during the timeout duration, the load balancer will disconnect the connection from the client and establish a new connection when there is a new request.</li> <li>- <b>Request Timeout (s):</b> specifies the timeout duration for waiting for a client request. Specifically: If the client fails to send a request header to the load balancer during the timeout duration, the request will be interrupted. If the interval between two consecutive request bodies reaching the load balancer is greater than the timeout duration, the connection will be disconnected.</li> <li>- <b>Response Timeout (s):</b> specifies the timeout duration for waiting for a response from a backend server. After a request is forwarded to the backend server, if the backend server does not respond during the timeout duration, the load balancer will stop waiting and return HTTP 504 Gateway Timeout.</li> </ul> </li> </ul>	<ul style="list-style-type: none"> <li>● External Protocol: HTTP</li> <li>● External Port: 80</li> <li>● Advanced Options <ul style="list-style-type: none"> <li>- Idle Timeout (s): 60</li> <li>- Request Timeout (s): 60</li> <li>- Response Timeout (s): 60</li> </ul> </li> </ul>

Parameter	Description	Example
Forwarding Policy	<ul style="list-style-type: none"> <li>• <b>Domain Name:</b> Enter an actual domain name to be accessed. If it is left blank, the ingress can be accessed through the IP address. Ensure that the domain name has been registered and licensed. Once a forwarding policy is configured with a domain name specified, you must use the domain name for access.</li> <li>• <b>Path Matching Rule:</b> Select <b>Prefix match</b>, <b>Exact match</b>, or <b>RegEx match</b>.</li> <li>• <b>Path:</b> Enter the path provided by a backend application for external access. The path added must be valid in the backend application, or the forwarding cannot take effect.</li> <li>• <b>Destination Service:</b> Select an existing Service or create a Service. Any Services that do not match the search criteria will be filtered out automatically.</li> <li>• <b>Destination Service Port:</b> Select the access port of the destination Service.</li> </ul>	<ul style="list-style-type: none"> <li>• Domain Name: You do not need to configure this parameter.</li> <li>• Path Matching Rule: Prefix match</li> <li>• Path: /</li> <li>• Destination Service: nginx</li> <li>• Destination Service Port: 80</li> </ul>

**Step 4** Click **OK**.

----End

## Using kubectl

**Step 1** Use kubectl to access the cluster. For details, see [Connecting to a Cluster Using kubectl](#).

**Step 2** Create a YAML file named **ingress-test.yaml**. The file name can be customized.

```
vi ingress-test.yaml
```

An example YAML file of an ingress associated with an existing load balancer is as follows:

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: test
  namespace: default
  annotations:
    kubernetes.io/elb.port: '80'
    kubernetes.io/elb.id: <your_elb_id> # In this example, an existing dedicated load balancer is used.
    # Replace its ID with the ID of your dedicated load balancer.
    kubernetes.io/elb.class: performance
    kubernetes.io/elb.keepalive_timeout: '300' # Timeout setting for client connections
    kubernetes.io/elb.client_timeout: '60' # Timeout duration for waiting for a request from a client
    kubernetes.io/elb.member_timeout: '60' # Timeout for waiting for a response from a backend
server
spec:
  rules:
```

```
- host: "
  http:
    paths:
      - path: /
        backend:
          service:
            name: test
            port:
              number: 80
          property:
            ingress.beta.kubernetes.io/url-match-mode: STARTS_WITH
            pathType: ImplementationSpecific
          ingressClassName: cce
```

**Table 7-67** Key annotation parameters

Parameter	Mandatory	Type	Description
kubernetes.io/elb.keepalive_timeout	No	String	Timeout for client connections. If there are no requests reaching the load balancer during the timeout duration, the load balancer will disconnect the connection from the client and establish a new connection when there is a new request.  The value ranges from 0 to 4000 seconds. The default value is <b>60</b> .
kubernetes.io/elb.client_timeout	No	String	Timeout for waiting for a request from a client. There are two cases: <ul style="list-style-type: none"> <li>• If the client fails to send a request header to the load balancer during the timeout duration, the request will be interrupted.</li> <li>• If the interval between two consecutive request bodies reaching the load balancer is greater than the timeout duration, the connection will be disconnected.</li> </ul> The value ranges from <b>1</b> to <b>300</b> (in seconds). The default value is <b>60</b> .
kubernetes.io/elb.member_timeout	No	String	Timeout for waiting for a response from a backend server. After a request is forwarded to the backend server, if the backend server does not respond within the duration specified by <b>member_timeout</b> , the load balancer will stop waiting and return HTTP 504 Gateway Timeout.  The value ranges from <b>1</b> to <b>300</b> (in seconds). The default value is <b>60</b> .

**Step 3** Create an ingress.

```
kubectl create -f ingress-test.yaml
```

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

```
ingress/ingress-test created
```

**Step 4** Check the created ingress.

```
kubectl get ingress
```

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

NAME	CLASS	HOSTS	ADDRESS	PORTS	AGE
ingress-test	cce	*	121.**.**.*	80	10s

----End

### 7.4.2.4.7 Configuring a Slow Start for a LoadBalancer Ingress

With slow start configured, a load balancer linearly increases the proportion of requests to backend server pods. When the slow start duration elapses, the load balancer sends full share of requests to backend pods and exits the slow start mode. Slow start gives applications time to warm up and respond to requests with optimal performance.

 **NOTE**

After a slow start is configured, if you delete the target annotation from the YAML file, slow start will not be enabled.

## Prerequisites

- A CCE standard cluster is available, and the cluster version is v1.23 or later.
- An available workload has been deployed in the cluster for external access. If no workload is available, deploy a workload by referring to [Creating a Deployment](#), [Creating a StatefulSet](#), or [Creating a DaemonSet](#).
- A Service for external access has been configured for the workload. [Services Supported by LoadBalancer Ingresses](#) lists the Service types supported by LoadBalancer ingresses.

## Notes and Constraints

- Only dedicated load balancers support slow start for HTTP and HTTPS backend server groups.
- Slow start takes effect only when the weighted round robin algorithm is used.
- Slow start takes effect only for new backend server pods.
- After the slow start duration elapses, backend servers will not enter the slow start mode again.
- Slow start takes effect when health check is enabled and backend server pods are running properly.
- If health check is disabled, slow start takes effect immediately.
- With slow start configured for an ingress, all forwarding policies of the ingress take effect.

## Procedure

**Step 1** Use kubectl to access the cluster. For details, see [Connecting to a Cluster Using kubectl](#).

**Step 2** Create a YAML file named **ingress-test.yaml**. The file name can be customized.

```
vi ingress-test.yaml
```

An example YAML file of an ingress associated with an existing load balancer is as follows:

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: ingress-test
  namespace: default
  annotations:
    kubernetes.io/elb.port: '80'
    kubernetes.io/elb.id: <your_elb_id> # Replace it with the ID of your existing load balancer.
    kubernetes.io/elb.class: performance
    kubernetes.io/elb.slowstart: '30' #Configure a slow start.
spec:
  rules:
  - host: ""
    http:
      paths:
      - path: /
        backend:
          service:
            name: <your_service_name> # Replace it with the name of your target Service.
            port:
              number: 80 # Replace 80 with the port number of your target Service.
          property:
            ingress.beta.kubernetes.io/url-match-mode: STARTS_WITH
            pathType: ImplementationSpecific
  ingressClassName: cce
```

**Table 7-68** Parameters for configuring a slow start

Parameter	Mandatory	Type	Description
kubernetes.io/elb.slowstart	No	String	<p>Specifies whether to enable slow start. After you enable it, the load balancer linearly increases the proportion of requests to backend server pods in this mode. When the slow start duration elapses, the load balancer sends full share of requests to backend server pods and exits the slow start mode.</p> <p><b>Clusters later than v1.23 support this field.</b></p> <p>The slow start duration ranges from <b>30</b> to <b>1200</b>. Duration of slow start, in seconds.</p> <ul style="list-style-type: none"> <li>Grayscale release applies only to dedicated load balancers.</li> <li>This parameter is valid only when the allocation policy of the target Service is weighted round robin (WRR) and sticky session is disabled.</li> </ul>



**Step 3** Create an ingress.

```
kubectl create -f ingress-test.yaml
```

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

```
ingress/ingress-test created
```

**Step 4** Check the created ingress.

```
kubectl get ingress
```

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

NAME	CLASS	HOSTS	ADDRESS	PORTS	AGE
ingress-test	cce	*	121.**.**.**	80	10s

----End

#### 7.4.2.4.8 Configuring a Range of Listening Ports for a LoadBalancer Ingress

Ingress allows you to customize listening ports. You can configure both HTTP and HTTPS listeners for a Service. For example, a Service can make available both HTTP port 80 and HTTPS port 443 for external access.

### Prerequisites

- A CCE standard cluster is available, and the cluster version meets the following requirements:
  - v1.23: v1.23.14-r0 or later
  - v1.25: v1.25.9-r0 or later
  - v1.27: v1.27.6-r0 or later
  - v1.28: v1.28.4-r0 or later
  - Other clusters of later versions
- An available workload has been deployed in the cluster for external access. If no workload is available, deploy a workload by referring to [Creating a Deployment](#), [Creating a StatefulSet](#), or [Creating a DaemonSet](#).
- A Service for external access has been configured for the workload. [Services Supported by LoadBalancer Ingresses](#) lists the Service types supported by LoadBalancer ingresses.

### Using kubectl

**Step 1** Use kubectl to access the cluster. For details, see [Connecting to a Cluster Using kubectl](#).

**Step 2** Create a YAML file named **ingress-test.yaml**. The file name can be customized.

```
vi ingress-test.yaml
```

The following shows an example configuration using an existing load balancer:

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  annotations:
    kubernetes.io/elb.id: 2c623150-17bf-45f1-ae6f-384b036f547e # ID of an existing load balancer
    kubernetes.io/elb.class: performance # Load balancer type
    kubernetes.io/elb.listen-ports: '[{"HTTP": 80}, {"HTTPS": 443}]' # Multi-listener configuration
    kubernetes.io/elb.tls-certificate-ids:
```

```
6cfb43c9de1a41a18478b868e34b0a82,6cfb43c9de1a41a18478b868e34b0a82 # HTTPS certificate
configuration
name: ingress-test
namespace: default
spec:
  ingressClassName: cce
  rules:
  - host: example.com
    http:
      paths:
      - backend:
          service:
            name: test
            port:
              number: 8888
        path: /
        pathType: ImplementationSpecific
    property:
      ingress.beta.kubernetes.io/url-match-mode: STARTS_WITH
```

**Table 7-69** Annotations for custom listening ports

Parameter	Type	Description
kubernetes.io/elb.listen-ports	String	<p>Configure multiple listening ports for an ingress. The port number ranges from 1 to 65535.</p> <p>The following is an example for JSON characters: kubernetes.io/elb.listen-ports: '[{"HTTP":80},{"HTTPS":443}]'</p> <ul style="list-style-type: none"> <li>• Only the listening ports that comply with both HTTP and HTTPS are allowed.</li> <li>• This function is available only for newly created ingresses. Additionally, after you configure multiple listening ports, the annotations cannot be modified or deleted.</li> <li>• If both <b>kubernetes.io/elb.listen-ports</b> and <b>kubernetes.io/elb.port</b> are configured, <b>kubernetes.io/elb.listen-ports</b> takes a higher priority.</li> <li>• Ingress configuration items such as the blacklist, trustlist, and timeout concurrently take effect on multiple listening ports. When HTTP/2 is enabled for an ingress, HTTP/2 takes effect only on the HTTPS port.</li> </ul>

**Step 3** Create an ingress.

```
kubectl create -f ingress-test.yaml
```

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

```
ingress/ingress-test created
```

**Step 4** Check the created ingress.

```
kubectl get ingress
```

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

NAME	CLASS	HOSTS	ADDRESS	PORTS	AGE
ingress-test	cce	example.com	121.**.**	80,443	10s

----End

## 7.4.3 Nginx Ingresses

### 7.4.3.1 Creating an Nginx Ingress on the Console

In Kubernetes, an ingress is a resource object that controls how Services within a cluster can be accessed from outside the cluster. You can use ingresses to configure different forwarding rules to access pods in a cluster. The following uses an [Nginx workload](#) as an example to describe how to create an Nginx ingress on the console.

#### Prerequisites

- An ingress provides network access for backend workloads. Ensure that a workload is available in a cluster. If no workload is available, deploy a workload by referring to [Creating a Deployment](#), [Creating a StatefulSet](#), or [Creating a DaemonSet](#).
- A ClusterIP or NodePort Service has been configured for the workload. For details about how to configure the Service, see [ClusterIP](#) or [NodePort](#).
- To add Nginx Ingress, ensure that the NGINX Ingress Controller add-on has been installed in the cluster. For details, see [Installing the Add-on](#).

#### Notes and Constraints

- **It is not recommended modifying any configuration of a load balancer on the ELB console. Otherwise, the Service will be abnormal.** If you have modified the configuration, uninstall the nginx-ingress add-on and reinstall it.
- The URL registered in an ingress forwarding policy must be the same as the URL used to access the backend Service. Otherwise, a 404 error will be returned.
- The selected or created load balancer must be in the same VPC as the current cluster, and it must match the load balancer type (private or public network).
- The load balancer has at least two listeners, and ports 80 and 443 are not occupied by listeners.

### Creating an Nginx Ingress

This section uses an Nginx workload as an example to describe how to create an Nginx ingress.

- Step 1** Log in to the CCE console and click the cluster name to access the cluster console.
- Step 2** Choose **Services & Ingresses** in the navigation pane, click the **Ingresses** tab, and click **Create Ingress** in the upper right corner.
- Step 3** Configure ingress parameters.
  - **Name:** Customize the name of an ingress, for example, **nginx-ingress-demo**.
  - **Namespace:** Select the namespace to which the ingress is to be added.

- **nginx-ingress:** This option is displayed only after the **NGINX Ingress Controller** add-on is installed in the cluster.
  - **External Protocol:** The options are **HTTP** and **HTTPS**. The default number of the listening port reserved when NGINX Ingress Controller is installed is 80 for HTTP and 443 for HTTPS. To use HTTPS, configure a certificate.
  - **Certificate Source:** source of a certificate for encrypting and authenticating HTTPS data transmission.
    - If you select a TLS key, you must create a key certificate of the IngressTLS or kubernetes.io/tls type beforehand. For details, see [Creating a Secret](#).
    - If you select the default certificate, NGINX Ingress Controller will use its default certificate for encryption and authentication. You can configure the default certificate during **NGINX Ingress Controller** installation. If the default certificate is not configured, the certificate provided by NGINX Ingress Controller will be used.
  - **SNI:** stands for Server Name Indication (SNI), which is an extended protocol of TLS. SNI allows multiple TLS-compliant domain names for external access using the same IP address and port number, and different domain names can use different security certificates. After SNI is enabled, the client is allowed to submit the requested domain name when initiating a TLS handshake request. After receiving the TLS request, the load balancer searches for the certificate based on the domain name in the request. If the certificate corresponding to the domain name is found, the load balancer returns the certificate for authorization. Otherwise, the default certificate (server certificate) is returned for authorization.
- **Forwarding Policy:** When the access address of a request matches the forwarding policy (a forwarding policy consists of a domain name and URL), the request is forwarded to the corresponding target Service for processing. Click **Add Forwarding Policies** to add multiple forwarding policies.
  - **Domain Name:** actual domain name. Ensure that the entered domain name has been registered and archived. After the ingress is created, bind the domain name to the IP address of the automatically created load balancer (IP address of the ingress access address). If a domain name rule is configured, the domain name must always be used for access.
  - **Path Matching Rule:**
    - **Default:** Prefix match is used by default.
    - **Prefix match:** If the URL is set to **/healthz**, the URL that meets the prefix can be accessed, for example, **/healthz/v1** and **/healthz/v2**.
    - **Exact match:** The URL can be accessed only when it is fully matched. For example, if the URL is set to **/healthz**, only **/healthz** can be accessed.
  - **Path:** access path, for example, **/healthz**

 NOTE

- The access path matching rule of Nginx Ingress is based on the path prefix separated by the slash (/) and is case-sensitive. If the subpath separated by a slash (/) matches the prefix, the access is normal. However, if the prefix is only a part of the character string in the subpath, the access is not matched. For example, if the URL is set to /healthz, /healthz/v1 is matched, but /healthzv1 is not matched.
- The access path added here must exist in the backend application. Otherwise, the forwarding fails.  
For example, the default access URL of the Nginx application is **/usr/share/nginx/html**. When adding **/test** to the ingress forwarding policy, ensure the access URL of your Nginx application contains **/usr/share/nginx/html/test**. Otherwise, error 404 will be returned.
- **Destination Service:** Select an existing Service or create a Service. Any Services that do not match the search criteria will be filtered out automatically.
- **Destination Service Port:** Select the access port of the destination Service.
- **Operation:** Click **Delete** to delete the configuration.
- **Annotation:** The value is in the format of key:value. You can use [annotations](#) to query the configurations supported by nginx-ingress.

**Step 4** Click **OK**.

After the ingress is created, it is displayed in the ingress list.

----End

### 7.4.3.2 Creating an Nginx Ingress Using kubectl

This section uses an [Nginx workload](#) as an example to describe how to create an Nginx ingress using kubectl.

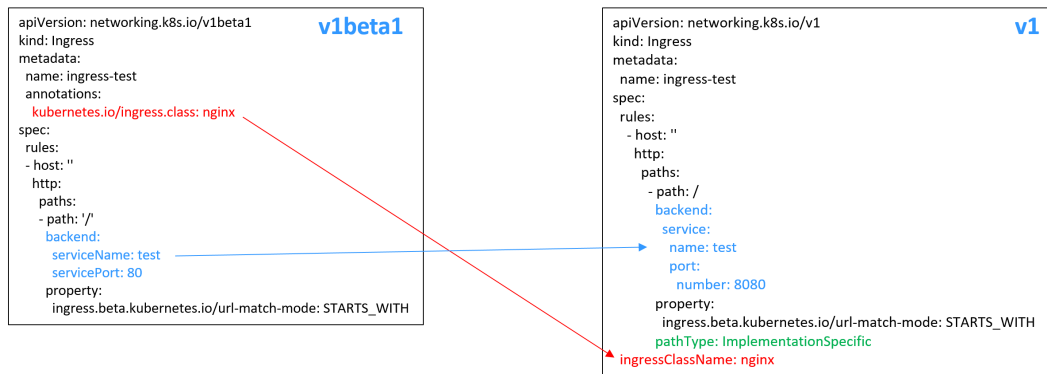
### Ingress API Version Upgrade in CCE Clusters v1.23

In CCE clusters of v1.23 or later, the ingress version is switched to **networking.k8s.io/v1**.

Compared with v1beta1, v1 has the following differences in parameters:

- The ingress type is specified by **spec.ingressClassName** instead of **kubernetes.io/ingress.class** in **annotations**.
- The format of **backend** has changed.
- The **pathType** parameter must be specified for each path. The options are as follows:
  - **ImplementationSpecific:** The matching method depends on Ingress Controller. The matching method defined by **ingress.beta.kubernetes.io/url-match-mode** is used in CCE, which is the same as v1beta1.
  - **Exact:** exact matching of the URL, which is case-sensitive.
  - **Prefix:** matching based on the URL prefix separated by a slash (/). The match is case-sensitive, and elements in the path are matched one by

one. A path element refers to a list of labels in the path separated by a slash (/).



## Prerequisites

- The NGINX Ingress Controller add-on has been installed in a cluster. For details, see [Installing the Add-on](#).
- An ingress provides network access for backend workloads. Ensure that a workload is available in a cluster. If no workload is available, deploy a workload by referring to [Creating a Deployment](#), [Creating a StatefulSet](#), or [Creating a DaemonSet](#).
- A ClusterIP or NodePort Service has been configured for the workload. For details about how to configure the Service, see [ClusterIP](#) or [NodePort](#).

## Creating an Nginx Ingress

**Step 1** Use kubectl to access the cluster. For details, see [Connecting to a Cluster Using kubectl](#).

**Step 2** Create a YAML file named **ingress-test.yaml**. The file name can be customized.

```
vi ingress-test.yaml
```

### NOTE

Starting from cluster v1.23, the ingress version is switched from **networking.k8s.io/v1beta1** to **networking.k8s.io/v1**. For details about the differences between v1 and v1beta1, see [Ingress API Version Upgrade in CCE Clusters v1.23](#).

The following uses HTTP as an example to describe how to configure the YAML file:

### For clusters of v1.23 or later:

```

apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: ingress-test
spec:
  rules:
  - host: ""
    http:
      paths:
      - path: /
        backend:
          service:
            name: <your_service_name> # Replace it with the name of your target Service.
            port:
  
```

```

    number: <your_service_port> # Replace it with the port number of your target Service.
  property:
    ingress.beta.kubernetes.io/url-match-mode: STARTS_WITH
    pathType: ImplementationSpecific
  ingressClassName: nginx # Nginx Ingress is used.

```

**For clusters of v1.21 or earlier:**

```

apiVersion: networking.k8s.io/v1beta1
kind: Ingress
metadata:
  name: ingress-test
  namespace: default
  annotations:
    kubernetes.io/ingress.class: nginx # Nginx Ingress is used.
spec:
  rules:
    - host: ""
      http:
        paths:
          - path: '/'
            backend:
              serviceName: <your_service_name> # Replace it with the name of your target Service.
              servicePort: <your_service_port> # Replace it with the port number of your target Service.

```

**Table 7-70** Key parameters

Parameter	Mandatory	Type	Description
kubernetes.io/ingress.class	Yes (only for clusters of v1.21 or earlier)	String	<b>nginx</b> : indicates that Nginx Ingress is used. This option is available only after the NGINX Ingress Controller add-on is installed.  This parameter is mandatory when an ingress is created by calling the API.
ingressClassName	Yes (only for clusters of v1.23 or later)	String	<b>nginx</b> : indicates that Nginx Ingress is used. This option is available only after the NGINX Ingress Controller add-on is installed.  This parameter is mandatory when an ingress is created by calling the API.
host	No	String	Domain name for accessing the Service. By default, this parameter is left blank, and the domain name needs to be fully matched. Ensure that the domain name has been registered and licensed. Once a forwarding policy is configured with a domain name specified, you must use the domain name for access.

Parameter	Mandatory	Type	Description
path	Yes	String	<p>User-defined route path. All external access requests must match <b>host</b> and <b>path</b>.</p> <p><b>NOTE</b></p> <ul style="list-style-type: none"> <li>The access path matching rule of Nginx Ingress is based on the path prefix separated by the slash (/) and is case-sensitive. If the subpath separated by a slash (/) matches the prefix, the access is normal. However, if the prefix is only a part of the character string in the subpath, the access is not matched. For example, if the URL is set to /healthz, /healthz/v1 is matched, but /healthzv1 is not matched.</li> <li>The access path added here must exist in the backend application. Otherwise, the forwarding fails. For example, the default access URL of the Nginx application is <b>/usr/share/nginx/html</b>. When adding <b>/test</b> to the ingress forwarding policy, ensure the access URL of your Nginx application contains <b>/usr/share/nginx/html/test</b>. Otherwise, error 404 will be returned.</li> </ul>
ingress.beta.kubernetes.io/url-match-mode	No	String	<p>Route matching policy.</p> <p>Default: <b>STARTS_WITH</b> (prefix match)</p> <p>Options:</p> <ul style="list-style-type: none"> <li><b>EQUAL_TO</b>: exact match</li> <li><b>STARTS_WITH</b>: prefix match</li> </ul>



Parameter	Mandatory	Type	Description
pathType	Yes	String	<p>Path type. This field is supported only by clusters of v1.23 or later.</p> <ul style="list-style-type: none"> <li>• <b>ImplementationSpecific:</b> The matching method depends on Ingress Controller. The matching method defined by <b>ingress.beta.kubernetes.io/url-match-mode</b> is used in CCE.</li> <li>• <b>Exact:</b> exact matching of the URL, which is case-sensitive.</li> <li>• <b>Prefix:</b> prefix matching, which is case-sensitive. With this method, the URL path is separated into multiple elements by slashes (/) and the elements are matched one by one. If each element in the URL matches the path, the subpaths of the URL can be routed normally.</li> </ul> <p><b>NOTE</b></p> <ul style="list-style-type: none"> <li>- During prefix matching, each element must be exactly matched. If the last element of the URL is the substring of the last element in the request path, no matching is performed. For example, <b>/foo/bar</b> matches <b>/foo/bar/baz</b> but does not match <b>/foo/barbaz</b>.</li> <li>- When elements are separated by slashes (/), if the URL or request path ends with a slash (/), the slash (/) at the end is ignored. For example, <b>/foo/bar</b> matches <b>/foo/bar/</b>.</li> </ul> <p>See <a href="#">examples</a> of ingress path matching.</p>

**Step 3** Create an ingress.

```
kubectl create -f ingress-test.yaml
```

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

```
ingress/ingress-test created
```

**Step 4** Check the created ingress.

```
kubectl get ingress
```

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

```
NAME          CLASS  HOSTS      ADDRESS      PORTS  AGE
ingress-test  nginx  *          121.**.**.**   80     10s
```

**Step 5** Enter **http://121.\*\*.\*\*.\*\*80** in the address box of the browser to access the workload (for example, [Nginx workload](#)).

**121.\*\*.\*\*.\*\*** indicates the IP address of the unified load balancer.

----End

### 7.4.3.3 Annotations for Configuring Nginx Ingresses

The nginx-ingress add-on in CCE uses the community chart and image. If the default add-on parameters cannot meet your demands, you can add annotations to define what you need, such as the default backend, timeout, and size of a request body.

This section describes common annotations used for creating an ingress of the Nginx type.

#### NOTE

- The key value of an annotation can only be a string. Other types (such as Boolean values or numeric values) must be enclosed in quotation marks (""), for example, "true", "false", and "100".
- Nginx Ingress supports native annotations of the community. For details, see [Annotations](#).
- [Ingress Type](#)
- [Configuring a Redirection Rule](#)
- [Configuring a URL Rewriting Rule](#)
- [Interconnecting with HTTPS Backend Services](#)
- [Creating a Consistent Hashing Rule for Load Balancing](#)
- [Customized Timeout Interval](#)
- [Customizing a Body Size](#)
- [Two-Way HTTPS Authentication](#)
- [Domain Name Regularization](#)
- [Documentation](#)

## Ingress Type

**Table 7-71** Ingress type annotations

Parameter	Type	Description	Supported Cluster Version
kubernetes.io/ingress.class	String	<ul style="list-style-type: none"> <li><b>nginx</b>: Nginx Ingress is used.</li> <li><b>cce</b>: A proprietary LoadBalancer ingress is used.</li> </ul> <p>This parameter is mandatory when an ingress is created by calling the API.</p> <p>For clusters of v1.23 or later, use the parameter <b>ingressClassName</b>. For details, see <a href="#">Creating an Nginx Ingress Using kubectl</a>.</p>	Only clusters of v1.21 or earlier

For details about how to use the preceding annotations, see [Creating an Nginx Ingress Using kubectl](#).

## Configuring a Redirection Rule

**Table 7-72** Redirection rule annotations

Parameter	Type	Description
nginx.ingress.kubernetes.io/permanent-redirect	String	Permanently redirects an access request to a target website (status code 301).
nginx.ingress.kubernetes.io/permanent-redirect-code	String	Changes the returned status code of a permanent redirection rule to a specified value.
nginx.ingress.kubernetes.io/temporal-redirect	String	Temporarily redirects an access request to a target website (status code 302).
nginx.ingress.kubernetes.io/ssl-redirect	String	Specifies whether an HTTP request can be redirected to HTTPS only through SSL. The default value is <b>true</b> when the ingress contains an SSL certificate.
nginx.ingress.kubernetes.io/force-ssl-redirect	String	Indicates whether to forcibly redirect a request to HTTPS even if TLS is not enabled for the ingress. When HTTP is used for access, the request is forcibly redirected (status code 308) to HTTPS.

For details, see [Configuring Redirection Rules for an Nginx Ingress](#).

## Configuring a URL Rewriting Rule

**Table 7-73** URL rewriting rule annotations

Parameter	Type	Description
nginx.ingress.kubernetes.io/rewrite-target	String	Target URI where the traffic must be redirected.

For details, see [Configuring URL Rewriting Rules for an Nginx Ingress](#).

## Interconnecting with HTTPS Backend Services

**Table 7-74** Annotations for interconnecting with HTTPS backend services

Parameter	Type	Description
nginx.ingress.kubernetes.io/backend-protocol	String	If this parameter is set to <b>HTTPS</b> , HTTPS is used to forward requests to the backend service container.

For details, see [Configuring HTTPS Backend Services for an Nginx Ingress](#).

## Creating a Consistent Hashing Rule for Load Balancing

**Table 7-75** Annotation of consistent hashing for load balancing

Parameter	Type	Description
nginx.ingress.kubernetes.io/upstream-hash-by	String	<p>Enable consistent hashing for load balancing for backend servers. The parameter value can be an Nginx parameter, a text value, or any combination. For example:</p> <ul style="list-style-type: none"> <li><b>nginx.ingress.kubernetes.io/upstream-hash-by: "\$request_uri"</b> indicates that requests are hashed based on the request URI.</li> <li><b>nginx.ingress.kubernetes.io/upstream-hash-by: "\$request_uri\$host"</b> indicates that requests are hashed based on the request URI and domain name.</li> <li><b>nginx.ingress.kubernetes.io/upstream-hash-by: "\${request_uri}-text-value"</b> indicates that requests are hashed based on the request URI and text value.</li> </ul>

For details, see [Configuring Consistent Hashing for Load Balancing of an Nginx Ingress](#).

## Customized Timeout Interval

**Table 7-76** Customized timeout interval annotations

Parameter	Type	Description
nginx.ingress.kubernetes.io/proxy-connect-timeout	String	Customized connection timeout interval. You do not need to set the unit when setting the timeout interval. The default unit is second.  Example: nginx.ingress.kubernetes.io/proxy-connect-timeout: '120'

## Customizing a Body Size

**Table 7-77** Annotations of customizing a body size

Parameter	Type	Description
nginx.ingress.kubernetes.io/proxy-body-size	String	When the body size in a request exceeds the upper limit, error 413 will be returned to the client. You can use this parameter to adjust the upper limit of the body size. The basic unit of the parameter value is byte. You can use units such as KB, MB, and GB. The unit conversion is as follows:  1 KB = 1024 bytes, 1 MB = 1024 KB, 1 GB =1024 MB  Example: nginx.ingress.kubernetes.io/proxy-body-size: 8m

## Two-Way HTTPS Authentication

Nginx Ingress supports two-way HTTPS authentication between the server and client to ensure secure connections.

**Step 1** Use kubectl to access the cluster. For details, see [Connecting to a Cluster Using kubectl](#).

**Step 2** Run the following command to create a self-signed CA certificate:

```
openssl req -x509 -sha256 -newkey rsa:4096 -keyout ca.key -out ca.crt -days 356 -nodes -subj '/CN=Ingress Cert Authority'
```

Expected output:

```
Generating a RSA private key
.....++++
.....++++
```

```
writing new private key to 'ca.key'  
-----
```

**Step 3** Create a server certificate.

1. Run the following command to create a request file for generating a server certificate:

```
openssl req -new -newkey rsa:4096 -keyout server.key -out server.csr -nodes -subj '/CN=foo.bar.com'
```

Expected output:

```
Generating a RSA private key  
.....++++  
.....++++  
writing new private key to 'server.key'  
-----
```

2. Run the following command to issue the server request file using the root certificate to generate the server certificate:

```
openssl x509 -req -sha256 -days 365 -in server.csr -CA ca.crt -CAkey ca.key -set_serial 01 -out server.crt
```

Expected output:

```
Signature ok  
subject=CN = foo.bar.com  
Getting CA Private Key
```

**Step 4** Create a client certificate.

1. Run the following command to create a request file for generating a client certificate:

```
openssl req -new -newkey rsa:4096 -keyout client.key -out client.csr -nodes -subj '/CN=Ingress'
```

Expected output:

```
Generating a RSA private key  
.....++++  
.....++++  
writing new private key to 'client.key'  
-----
```

2. Run the following command to issue the client request file using the root certificate to generate the client certificate:

```
openssl x509 -req -sha256 -days 365 -in client.csr -CA ca.crt -CAkey ca.key -set_serial 02 -out client.crt
```

Expected output:

```
Signature ok  
subject=CN = Ingress  
Getting CA Private Key
```

**Step 5** Run the **ls** command to check the created certificates.

Expected output:

```
ca.crt ca.key client.crt client.csr client.key server.crt server.csr server.key
```

**Step 6** Run the following command to create a secret of the CA certificate:

```
kubectl create secret generic ca-secret --from-file=ca.crt=ca.crt
```

Expected output:

```
secret/ca-secret created
```

**Step 7** Run the following command to create a secret of the server certificate:

```
kubectl create secret generic tls-secret --from-file=tls.crt=server.crt --from-file=tls.key=server.key
```

Expected output:

```
secret/tls-secret created
```

**Step 8** Create a YAML file named **ingress-test.yaml**. The file name can be customized.

```
vi ingress-test.yaml
```

- **For clusters of v1.23 or later:**

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  annotations:
    nginx.ingress.kubernetes.io/auth-tls-verify-client: "on"
    nginx.ingress.kubernetes.io/auth-tls-secret: "default/ca-secret" # Replace it with your CA
certificate secret.
    nginx.ingress.kubernetes.io/auth-tls-verify-depth: "1"
    nginx.ingress.kubernetes.io/auth-tls-pass-certificate-to-upstream: "true"
  name: ingress-test
  namespace: default
spec:
  rules:
  - host: foo.bar.com
    http:
      paths:
      - backend:
          service:
            name: nginx-test # Replace it with the name of your target Service.
            port:
              number: 80 # Replace it with the port of your target Service.
          path: /
          pathType: ImplementationSpecific
    tls:
      - hosts:
          - foo.bar.com
        secretName: tls-secret # Replace it with your TLS certificate secret.
      ingressClassName: nginx
```

- **For clusters of v1.21 or earlier:**

```
apiVersion: networking.k8s.io/v1beta1
kind: Ingress
metadata:
  annotations:
    kubernetes.io/ingress.class: nginx
    nginx.ingress.kubernetes.io/auth-tls-verify-client: "on"
    nginx.ingress.kubernetes.io/auth-tls-secret: "default/ca-secret" # Replace it with your CA
certificate secret.
    nginx.ingress.kubernetes.io/auth-tls-verify-depth: "1"
    nginx.ingress.kubernetes.io/auth-tls-pass-certificate-to-upstream: "true"
  name: ingress-test
  namespace: default
spec:
  rules:
  - host: foo.bar.com
    http:
      paths:
      - path: '/'
        backend:
          serviceName: nginx-test # Replace it with the name of your target Service.
          servicePort: 80 # Replace it with the port of your target Service.
    tls:
      - hosts:
          - foo.bar.com
        secretName: tls-secret # Replace it with your TLS key certificate.
```

**Step 9** Run the following command to create an ingress:

```
kubectl create -f ingress-test.yaml
```

Expected output:

```
ingress.networking.k8s.io/ingress-test created
```

**Step 10** Run the following command to obtain the IP address of the ingress:

```
kubectl get ingress
```

Expected output:

```
NAME      CLASS  HOSTS      ADDRESS  PORTS  AGE
nginx-test nginx  foo.bar.com 10.3.xx.xx 80, 443 27m
```

**Step 11** Run the following command to update the IP address of the ingress into the **hosts** file and replace the following IP address with the actual IP address of the ingress:

```
echo "10.3.xx.xx foo.bar.com" | sudo tee -a /etc/hosts
```

Expected output:

```
10.3.xx.xx foo.bar.com
```

**Step 12** Verify the configuration.

- The client does not send the certificate for access.

```
curl --cacert ./ca.crt https://foo.bar.com
```

Expected output:

```
<html>
<head><title>400 No required SSL certificate was sent</title></head>
<body>
<center><h1>400 Bad Request</h1></center>
<center>No required SSL certificate was sent</center>
<hr><center>nginx</center>
</body>
</html>
```

- The client sends the certificate for access.

```
curl --cacert ./ca.crt --cert ./client.crt --key ./client.key https://foo.bar.com
```

Expected output:

```
<!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>
<p>If you see this page, the nginx web server is successfully installed and
working. Further configuration is required.</p>

<p>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>.</p>

<p><em>Thank you for using nginx.</em></p>
</body>
</html>
```

----End

## Domain Name Regularization

Nginx Ingress allows you to configure the **nginx.ingress.kubernetes.io/server-alias** annotation to configure regular expressions for domain names.

**Step 1** Use `kubectl` to access the cluster. For details, see [Connecting to a Cluster Using kubectl](#).



**Step 2** Create a YAML file named **ingress-test.yaml**. The file name can be customized.

```
vi ingress-test.yaml
```

For example, the regular expression **~^www\.\d+\.example\.com**  
**,\$abc.example.com** indicates that you can access the ingress using **www.{One or more digits}.example.com** and **abc.example.com**.

- **For clusters of v1.23 or later:**

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  annotations:
    nginx.ingress.kubernetes.io/server-alias: '~^www\.\d+\.example\.com,$abc.example.com'
  name: ingress-test
  namespace: default
spec:
  rules:
  - host: foo.bar.com
    http:
      paths:
      - backend:
          service:
            name: nginx-93244 # Replace it with the name of your target Service.
            port:
              number: 80 # Replace it with the port of your target Service.
        path: /
        pathType: ImplementationSpecific
    ingressClassName: nginx
```

- **For clusters of v1.21 or earlier:**

```
apiVersion: networking.k8s.io/v1beta1
kind: Ingress
metadata:
  annotations:
    kubernetes.io/ingress.class: nginx
    nginx.ingress.kubernetes.io/server-alias: '~^www\.\d+\.example\.com,$abc.example.com'
  name: ingress-test
  namespace: default
spec:
  rules:
  - host: foo.bar.com
    http:
      paths:
      - path: '/'
        backend:
          serviceName: nginx-test # Replace it with the name of your target Service.
          servicePort: 80 # Replace it with the port of your target Service.
```

**Step 3** Run the following command to create an ingress:

```
kubectl create -f ingress-test.yaml
```

Expected output:

```
ingress.networking.k8s.io/ingress-test created
```

**Step 4** Check the NGINX Ingress Controller configuration.

1. Run the following command to check the NGINX Ingress Controller pods:

```
kubectl get pods -n kube-system | grep nginx-ingress-controller
```

Expected output:

```
cceaddon-nginx-ingress-controller-68d7bcc67-dxxxx 1/1 Running 0 18h
cceaddon-nginx-ingress-controller-68d7bcc67-cxxxx 1/1 Running 0 18h
```

2. Run the following command to check the NGINX Ingress Controller configuration:

```
kubectl exec -n kube-system cceaddon-nginx-ingress-controller-68d7bcc67-dxxxx cat /etc/nginx/nginx.conf | grep -C3 "foo.bar.com"
```

Expected output:

```
## start server foo.bar.com
server {
    server_name foo.bar.com abc.example.com ~^www\.\d+\.example\.com$ ;

    listen 80 ;
    listen [::]:80 ;

}

}
## end server foo.bar.com
```

**Step 5** Run the following command to obtain the IP address of the ingress:

```
kubectl get ingress
```

Expected output:

NAME	CLASS	HOSTS	ADDRESS	PORTS	AGE
nginx-test	nginx	foo.bar.com	10.3.xx.xx	80	14m

**Step 6** Use different rules to test service access.

- Run the following command to access the service through **Host: foo.bar.com:**  

```
curl -H "Host: foo.bar.com" 10.3.xx.xx/
```

It is expected that the web page can be accessed properly.
- Run the following command to access the service through **Host: www.123.example.com:**  

```
curl -H "Host: www.123.example.com" 10.3.xx.xx/
```

It is expected that the web page can be accessed properly.
- Run the following command to access the service through **Host: www.321.example.com:**  

```
curl -H "Host: www.321.example.com" 10.3.xx.xx/
```

It is expected that the web page can be accessed properly.

----End

## Documentation

For details about annotation parameters supported by Nginx Ingress, see [Annotations](#).

### 7.4.3.4 Advanced Setting Examples of Nginx Ingresses

#### 7.4.3.4.1 Configuring an HTTPS Certificate for an Nginx Ingress

HTTPS certificates can be configured for ingresses to provide security services.

**Step 1** Use kubectl to access the cluster. For details, see [Connecting to a Cluster Using kubectl](#).

**Step 2** Ingress supports two TLS secret types: kubernetes.io/tls and IngressTLS. IngressTLS is used as an example. For details, see [Creating a Secret](#). For details about examples of the kubernetes.io/tls secret and its description, see [TLS secrets](#).

Create a YAML file named **ingress-test-secret.yaml**. The file name can be customized.

```
vi ingress-test-secret.yaml
```

**The YAML file is configured as follows:**

```
apiVersion: v1
data:
  tls.crt: LS0*****tLS0tCg==
  tls.key: LS0tL*****0tLS0K
kind: Secret
metadata:
  annotations:
    description: test for ingressTLS secrets
    name: ingress-test-secret
    namespace: default
type: IngressTLS
```

 **NOTE**

In the preceding information, **tls.crt** and **tls.key** are only examples. Replace them with the actual files. The values of **tls.crt** and **tls.key** are Base64-encoded.

**Step 3** Create a secret.

```
kubectl create -f ingress-test-secret.yaml
```

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

```
secret/ingress-test-secret created
```

Check the created secret.

```
kubectl get secrets
```

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

NAME	TYPE	DATA	AGE
ingress-test-secret	IngressTLS	2	13s

**Step 4** Create a YAML file named **ingress-test.yaml**. The file name can be customized.

```
vi ingress-test.yaml
```

**For clusters of v1.23 or later:**

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: ingress-test
  namespace: default
spec:
  tls:
    - hosts:
      - foo.bar.com
      secretName: ingress-test-secret # Replace it with your TLS key certificate.
  rules:
    - host: foo.bar.com
      http:
        paths:
          - path: /
            backend:
              service:
                name: <your_service_name> # Replace it with the name of your target Service.
                port:
                  number: <your_service_port> # Replace it with the port number of your target Service.
            property:
              ingress.beta.kubernetes.io/url-match-mode: STARTS_WITH
            pathType: ImplementationSpecific
          ingressClassName: nginx
```

**For clusters of v1.21 or earlier:**

```
apiVersion: networking.k8s.io/v1beta1
kind: Ingress
metadata:
  name: ingress-test
```

```

annotations:
  kubernetes.io/ingress.class: nginx
spec:
  tls:
  - hosts:
    - foo.bar.com
    secretName: ingress-test-secret # Replace it with your TLS key certificate.
  rules:
  - host: foo.bar.com
    http:
      paths:
      - path: '/'
        backend:
          serviceName: <your_service_name> # Replace it with the name of your target Service.
          servicePort: <your_service_port> # Replace it with the port number of your target Service.
  ingressClassName: nginx

```

**Step 5** Create an ingress.

```
kubectl create -f ingress-test.yaml
```

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

```
ingress/ingress-test created
```

**Step 6** Check the created ingress.

```
kubectl get ingress
```

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

NAME	CLASS	HOSTS	ADDRESS	PORTS	AGE
ingress-test	nginx	*	121.**.**.**	80	10s

**Step 7** Enter **https://121.\*\*.\*\*.\*\*443** in the address box of the browser to access the workload (for example, **Ngix workload**).

**121.\*\*.\*\*.\*\*** indicates the IP address of the unified load balancer.

----End

### 7.4.3.4.2 Configuring Redirection Rules for an Nginx Ingress

#### Configuring a Permanent Redirection Rule

To permanently redirect an access request to a target website (status code 301), use the **nginx.ingress.kubernetes.io/permanent-redirect** annotation. For example, to permanently redirect all access requests to **www.example.com**, run the following command:

```
nginx.ingress.kubernetes.io/permanent-redirect: https://www.example.com
```

The configuration in an Nginx Ingress is as follows:

**Step 1** Use kubectl to access the cluster. For details, see [Connecting to a Cluster Using kubectl](#).

**Step 2** Create a YAML file named **ingress-test.yaml**. The file name can be customized.

```
vi ingress-test.yaml
```

**For clusters of v1.23 or later:**

```

apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: ingress-test
  namespace: default

```

```

annotations:
  nginx.ingress.kubernetes.io/permanent-redirect: https://www.example.com
spec:
  rules:
  - host: ""
    http:
      paths:
      - path: /
        backend:
          service:
            name: <your_service_name> # Replace it with the name of your target Service.
            port:
              number: <your_service_port> # Replace it with the port number of your target Service.
          property:
            ingress.beta.kubernetes.io/url-match-mode: STARTS_WITH
            pathType: ImplementationSpecific
        ingressClassName: nginx

```

**For clusters of v1.21 or earlier:**

```

apiVersion: networking.k8s.io/v1beta1
kind: Ingress
metadata:
  name: ingress-test
  namespace: default
  annotations:
    kubernetes.io/ingress.class: nginx
    nginx.ingress.kubernetes.io/permanent-redirect: https://www.example.com
spec:
  rules:
  - host: ""
    http:
      paths:
      - path: /
        backend:
          serviceName: <your_service_name> # Replace it with the name of your target Service.
          servicePort: <your_service_port> # Replace it with the port number of your target Service.

```

**Step 3** Create an ingress.

```
kubectl create -f ingress-test.yaml
```

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

```
ingress/ingress-test created
```

**Step 4** Check the created ingress.

```
kubectl get ingress
```

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

NAME	CLASS	HOSTS	ADDRESS	PORTS	AGE
ingress-test	nginx	*	121.**.**.**	80	10s

**Step 5** Access the ingress.  $\{ELB\_IP\}$  specifies the IP address of the load balancer associated with the Nginx ingress.

```
curl -I  $\{ELB\_IP\}$ 
```

The access path will be redirected to **www.example.com**.

```

HTTP/1.1 301 Moved Permanently
Date: Sat, 20 Jul 2024 07:55:49 GMT
Content-Type: text/html
Content-Length: 162
Connection: keep-alive
Location: https://www.example.com

```

----End

## Configuring the Returned Status Code for the Permanent Redirection Rule

When configuring a permanent redirection rule, you can use the **nginx.ingress.kubernetes.io/permanent-redirect-code** annotation to modify its returned status code. For example, to set the status code for the permanent redirection to 308, run the following command:

```
nginx.ingress.kubernetes.io/permanent-redirect-code: '308'
```

The configuration in an Nginx Ingress is as follows:

**Step 1** Use `kubectl` to access the cluster. For details, see [Connecting to a Cluster Using kubectl](#).

**Step 2** Create a YAML file named **ingress-test.yaml**. The file name can be customized.

```
vi ingress-test.yaml
```

### For clusters of v1.23 or later:

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: ingress-test
  namespace: default
  annotations:
    nginx.ingress.kubernetes.io/permanent-redirect: https://www.example.com
    nginx.ingress.kubernetes.io/permanent-redirect-code: '308'
spec:
  rules:
  - host: ""
    http:
      paths:
      - path: /
        backend:
          service:
            name: <your_service_name> # Replace it with the name of your target Service.
            port:
              number: <your_service_port> # Replace it with the port number of your target Service.
          property:
            ingress.beta.kubernetes.io/url-match-mode: STARTS_WITH
            pathType: ImplementationSpecific
        ingressClassName: nginx
```

### For clusters of v1.21 or earlier:

```
apiVersion: networking.k8s.io/v1beta1
kind: Ingress
metadata:
  name: ingress-test
  namespace: default
  annotations:
    kubernetes.io/ingress.class: nginx
    nginx.ingress.kubernetes.io/permanent-redirect: https://www.example.com
    nginx.ingress.kubernetes.io/permanent-redirect-code: '308'
spec:
  rules:
  - host: ""
    http:
      paths:
      - path: /
        backend:
          serviceName: <your_service_name> # Replace it with the name of your target Service.
          servicePort: <your_service_port> # Replace it with the port number of your target Service.
```

**Step 3** Create an ingress.

```
kubectl create -f ingress-test.yaml
```

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

```
ingress/ingress-test created
```

**Step 4** Check the created ingress.

```
kubectl get ingress
```

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

NAME	CLASS	HOSTS	ADDRESS	PORTS	AGE
ingress-test	nginx	*	121.**.**.*	80	10s

**Step 5** Access the ingress. `#{ELB_IP}` specifies the IP address of the load balancer associated with the Nginx ingress.

```
curl -I #{ELB_IP}
```

The access path will be redirected to **www.example.com**, and the status code is 308.

```
HTTP/1.1 308 Moved Permanently
Date: Sat, 20 Jul 2024 07:55:49 GMT
Content-Type: text/html
Content-Length: 162
Connection: keep-alive
Location: https://www.example.com
```

----End

## Configuring a Temporary Redirection Rule

To temporarily redirect an access request to a target website (status code 302), use the `nginx.ingress.kubernetes.io/temporal-redirect` annotation. For example, to temporarily redirect all access requests to **www.example.com**, run the following command:

```
nginx.ingress.kubernetes.io/temporal-redirect: https://www.example.com
```

The configuration in an Nginx Ingress is as follows:

**Step 1** Use `kubectl` to access the cluster. For details, see [Connecting to a Cluster Using kubectl](#).

**Step 2** Create a YAML file named `ingress-test.yaml`. The file name can be customized.

```
vi ingress-test.yaml
```

**For clusters of v1.23 or later:**

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: ingress-test
  namespace: default
  annotations:
    nginx.ingress.kubernetes.io/temporal-redirect: https://www.example.com
spec:
  rules:
    - host: ""
      http:
        paths:
          - path: /
            backend:
              service:
                name: <your_service_name> # Replace it with the name of your target Service.
                port:
                  number: <your_service_port> # Replace it with the port number of your target Service.
            property:
              ingress.beta.kubernetes.io/url-match-mode: STARTS_WITH
```

```
pathType: ImplementationSpecific
ingressClassName: nginx
```

**For clusters of v1.21 or earlier:**

```
apiVersion: networking.k8s.io/v1beta1
kind: Ingress
metadata:
  name: ingress-test
  namespace: default
  annotations:
    kubernetes.io/ingress.class: nginx
    nginx.ingress.kubernetes.io/temporal-redirect: https://www.example.com
spec:
  rules:
    - host: "
      http:
        paths:
          - path: /
            backend:
              serviceName: <your_service_name> # Replace it with the name of your target Service.
              servicePort: <your_service_port> # Replace it with the port number of your target Service.
```

**Step 3** Create an ingress.

```
kubectl create -f ingress-test.yaml
```

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

```
ingress/ingress-test created
```

**Step 4** Check the created ingress.

```
kubectl get ingress
```

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

NAME	CLASS	HOSTS	ADDRESS	PORTS	AGE
ingress-test	nginx	*	121.**.**.**	80	10s

**Step 5** Access the ingress. `#{ELB_IP}` specifies the IP address of the load balancer associated with the Nginx ingress.

```
curl -I #{ELB_IP}
```

The access path will be redirected to **www.example.com**, and the status code is 302.

```
HTTP/1.1 302 Moved Temporarily
Date: Sat, 20 Jul 2024 08:01:02 GMT
Content-Type: text/html
Content-Length: 138
Connection: keep-alive
Location: https://www.example.com
```

----End

## Redirecting HTTP to HTTPS

By default, if an ingress uses TLS, requests will be redirected (status code 308) to HTTPS when HTTP is used for access. You can configure the redirection by using the `nginx.ingress.kubernetes.io/ssl-redirect` annotation.

- If the value of this annotation is set to **true**, an HTTP access is redirected to HTTPS (status code 308) when the TLS certificate is used.
- If the value of this annotation is set to **false**, an HTTP access cannot be redirected to HTTPS when the TLS certificate is used.



If you need to forcibly redirect an HTTP access to HTTPS without a TLS, you can configure the redirection by setting the value of `nginx.ingress.kubernetes.io/force-ssl-redirect` to `true`.

**Step 1** Use `kubectl` to access the cluster. For details, see [Connecting to a Cluster Using kubectl](#).

**Step 2** Create a YAML file named `ingress-test.yaml`. The file name can be customized.

```
vi ingress-test.yaml
```

**For clusters of v1.23 or later:**

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: ingress-test
  namespace: default
  annotations:
    nginx.ingress.kubernetes.io/ssl-redirect: 'true'
spec:
  rules:
  - host: ""
    http:
      paths:
      - path: /
        backend:
          service:
            name: <your_service_name> # Replace it with the name of your target Service.
            port:
              number: <your_service_port> # Replace it with the port number of your target Service.
          property:
            ingress.beta.kubernetes.io/url-match-mode: STARTS_WITH
          pathType: ImplementationSpecific
        ingressClassName: nginx
```

**For clusters of v1.21 or earlier:**

```
apiVersion: networking.k8s.io/v1beta1
kind: Ingress
metadata:
  name: ingress-test
  namespace: default
  annotations:
    kubernetes.io/ingress.class: nginx
    nginx.ingress.kubernetes.io/ssl-redirect: 'true'
spec:
  rules:
  - host: ""
    http:
      paths:
      - path: /
        backend:
          serviceName: <your_service_name> # Replace it with the name of your target Service.
          servicePort: <your_service_port> # Replace it with the port number of your target Service.
```

**Step 3** Create an ingress.

```
kubectl create -f ingress-test.yaml
```

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

```
ingress/ingress-test created
```

**Step 4** Check the created ingress.

```
kubectl get ingress
```

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

NAME	CLASS	HOSTS	ADDRESS	PORTS	AGE
ingress-test	nginx	*	121.**.**.**	80	10s

- Step 5** Access the ingress. `#{ELB_IP}` specifies the IP address of the load balancer associated with the Nginx ingress.

```
curl -I #{ELB_IP}
```

The access path will be redirected to HTTPS, and the status code is 308.

```
HTTP/1.1 308 Permanent Redirect
Date: Sat, 20 Jul 2024 08:03:36 GMT
Content-Type: text/html
Content-Length: 164
Connection: keep-alive
Location: https://#{ELB_IP}
```

----End

#### 7.4.3.4.3 Configuring URL Rewriting Rules for an Nginx Ingress

In some application scenarios, the access URL provided by the backend service is different from the path specified in the ingress rule. The ingress directly forwards the access path to the same backend path. If URL rewriting is not configured, 404 is returned for all access requests. For example, if the access path in the ingress rule is set to `/app/demo` and the access path provided by the backend service is `/demo`, access requests are directly forwarded to the `/app/demo` path of the backend service, which does not match the actual access path (`/demo`) provided by the backend service. As a result, 404 is returned.

In this case, you can use the Rewrite method to implement URL rewriting. That is, you can use the `nginx.ingress.kubernetes.io/rewrite-target` annotation to implement rewriting rules for different paths.

### Configuring a Rewriting Rule

- Step 1** Use `kubectl` to access the cluster. For details, see [Connecting to a Cluster Using kubectl](#).
- Step 2** Create a YAML file named `ingress-test.yaml`. The file name can be customized.

```
vi ingress-test.yaml
```

#### For clusters of v1.23 or later:

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: ingress-test
  namespace: default
  annotations:
    nginx.ingress.kubernetes.io/rewrite-target: /$2
spec:
  rules:
    - host: ""
      http:
        paths:
          - path: '/something(/|$)(.*)'
            backend:
              service:
                name: <your_service_name> # Replace it with the name of your target Service.
                port:
                  number: <your_service_port> # Replace it with the port number of your target Service.
            property:
              ingress.beta.kubernetes.io/url-match-mode: STARTS_WITH
            pathType: ImplementationSpecific
          ingressClassName: nginx
```

#### For clusters of v1.21 or earlier:

```
apiVersion: networking.k8s.io/v1beta1
kind: Ingress
metadata:
  name: ingress-test
  namespace: default
  annotations:
    kubernetes.io/ingress.class: nginx
    nginx.ingress.kubernetes.io/rewrite-target: /$2
spec:
  rules:
    - host: ""
      http:
        paths:
          - path: '/something(/|$)(.*)'
            backend:
              serviceName: <your_service_name> # Replace it with the name of your target Service.
              servicePort: <your_service_port> # Replace it with the port number of your target Service.
```

### NOTE

As long as **rewrite-target** is specified for one ingress, all paths under the same host in all ingress definitions are case-sensitive, including the ingresses that do not have **rewrite-target** specified.

In the preceding example, the placeholder **\$2** specifies that all characters matched by the second parenthesis (.\* ) are added to the **nginx.ingress.kubernetes.io/rewrite-target** annotation as the rewritten URL path.

For example, the preceding regular expression matching for ingress causes URL rewriting in multiple cases. These cases include:

- The **/something** path is rewritten to the **/** path.
- The **/something/** path is rewritten to the **/** path.
- The **/something/new** path is rewritten to the **/new** path.

### Step 3 Create an ingress.

```
kubectl create -f ingress-test.yaml
```

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

```
ingress/ingress-test created
```

### Step 4 Check the created ingress.

```
kubectl get ingress
```

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

NAME	CLASS	HOSTS	ADDRESS	PORTS	AGE
ingress-test	nginx	*	121.**.**.**	80	10s

### Step 5 Access the ingress. `/${ELB_IP}` specifies the IP address of the load balancer associated with the Nginx ingress.

```
curl ${ELB_IP}/something
```

The access path will be redirected to **/**.

----End

## Checking the Configurations of an Ingress

In the **nginx-ingress-controller** container, you can check all configurations of an ingress in the **nginx.conf** file in the **/etc/nginx** directory.

**Step 1** Obtain the name of the pod that runs **nginx-ingress-controller**.

```
kubectl get pod -n kube-system | grep nginx-ingress-controller
```

The command output is as follows:

```
cceaddon-nginx-ingress-controller-66855ccb9b-52qcd    1/1    Running    0    37m
```

**Step 2** Log in to the **nginx-ingress-controller** container.

```
kubectl exec -it cceaddon-nginx-ingress-controller-66855ccb9b-52qcd -- /bin/bash
```

**Step 3** Check the **nginx.conf** file in **/etc/nginx**.

```
cat /etc/nginx/nginx.conf
```

The rewriting rule in this example generates a Rewrite command and writes it to the **location** field in the **nginx.conf** file as follows:

```
## start server _
server {
    server_name _;
    ...
    location ~* "^/something(/|$)(.*)" {
        set $namespace    "default";
        set $ingress_name  "ingress-test";
        set $service_name  "<your_service_name>";
        set $service_port  "80";
        ...
        rewrite "(?i)/something(/|$)(.*)" /$2 break;
        ...
    }
}
## end server _
```

The basic syntax of the Rewrite command is as follows:

```
rewrite regex replacement [flag];
```

- **regex**: regular expression for matching URIs. In the preceding example, **(?i)/something(/|\$)(.\*)** is the regular expression for matching URIs, where **(?i)** indicates case-insensitive.
- **replacement**: content to rewrite. In the preceding example, **/\$2** indicates that the path is rewritten to all the characters matched by the second parenthesis **(.\*)**.
- **flag**: rewrite format.
  - **last**: continues to match the next rule after the current rule is matched.
  - **break**: stops matching after the current rule is matched.
  - **redirect**: returns a temporary redirect with the 302 code.
  - **permanent**: returns a permanent redirect with the 301 code.

----End

## Advanced Rewrite Configuration

Some complex, advanced Rewrite requirements can be implemented by modifying the Nginx configuration file **nginx.conf**. However, the **nginx.ingress.kubernetes.io/rewrite-target** annotation function can be customized to meet more complex Rewrite requirements.

- **nginx.ingress.kubernetes.io/server-snippet**: Add custom settings to the **server** field in the **nginx.conf** file.
- **nginx.ingress.kubernetes.io/configuration-snippet**: Add custom settings to the **location** field in the **nginx.conf** file.

 NOTE

Snippet is not enabled by default when the NGINX Ingress Controller version is 2.4.6 or later (corresponding to the community version v1.9.3). For details, see [Changelog](#). If snippet is needed, enable it using [allow-snippet-annotations](#).

You can use the preceding two annotations to insert a Rewrite command into the **server** or **location** field in the **nginx.conf** file to rewrite the URL. The following is an example:

```
annotations:
  kubernetes.io/ingress.class: "nginx"
  nginx.ingress.kubernetes.io/configuration-snippet: |
    rewrite ^/stylesheets/(.*)$ /something/stylesheets/$1 redirect; # Add the /something prefix.
    rewrite ^/images/(.*)$ /something/images/$1 redirect; # Add the /something prefix.
```

In the preceding rules, the **/something** prefix is added to the access URL. Specifically:

- When a user accesses the **/stylesheets/new.css** path, the path is rewritten as the **/something/stylesheets/new.css** path.
- When a user accesses the **/images/new.jpg** path, the path is rewritten as the **/something/images/new.jpg** path.

#### 7.4.3.4.4 Configuring HTTPS Backend Services for an Nginx Ingress

Ingress can function as a proxy for backend services using different protocols. By default, the backend proxy channel of an ingress is an HTTP channel. To create an HTTPS channel, add the following configuration to the **annotations** field:

```
nginx.ingress.kubernetes.io/backend-protocol: "HTTPS"
```

An ingress configuration example is as follows:

**Step 1** Use `kubectl` to access the cluster. For details, see [Connecting to a Cluster Using kubectl](#).

**Step 2** Create a YAML file named **ingress-test.yaml**. The file name can be customized.

```
vi ingress-test.yaml
```

**For clusters of v1.23 or later:**

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: ingress-test
  namespace: default
  annotations:
    nginx.ingress.kubernetes.io/backend-protocol: "HTTPS"
spec:
  tls:
  - secretName: ingress-test-secret # Replace it with your TLS key certificate.
  rules:
  - host: ""
    http:
      paths:
      - path: '/'
        backend:
          service:
            name: <your_service_name> # Replace it with the name of your target Service.
            port:
              number: <your_service_port> # Replace it with the port number of your target Service.
          property:
            ingress.beta.kubernetes.io/url-match-mode: STARTS_WITH
```

```

pathType: ImplementationSpecific
ingressClassName: nginx

For clusters of v1.21 or earlier:
apiVersion: networking.k8s.io/v1beta1
kind: Ingress
metadata:
  name: ingress-test
  namespace: default
  annotations:
    kubernetes.io/ingress.class: nginx
    nginx.ingress.kubernetes.io/backend-protocol: "HTTPS"
spec:
  tls:
    - secretName: ingress-test-secret # Replace it with your TLS key certificate.
  rules:
    - host: ""
      http:
        paths:
          - path: "/"
            backend:
              serviceName: <your_service_name> # Replace it with the name of your target Service.
              servicePort: <your_service_port> # Replace it with the port number of your target Service.

```

**Step 3** Create an ingress.

```
kubectl create -f ingress-test.yaml
```

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

```
ingress/ingress-test created
```

**Step 4** Check the created ingress.

```
kubectl get ingress
```

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

NAME	CLASS	HOSTS	ADDRESS	PORTS	AGE
ingress-test	nginx	*	121.**.**.*	80	10s

----End

### 7.4.3.4.5 Configuring Consistent Hashing for Load Balancing of an Nginx Ingress

The native Nginx supports multiple load balancing rules, including weighted round robin and IP hash. Nginx Ingress supports load balancing by using consistent hashing based on the native Nginx capabilities.

By default, the IP hash method supported by Nginx uses the linear hash space. The backend server is selected based on the hash value of the IP address. However, when this method is used to add or delete a node, all IP addresses need to be hashed again and then routed again. As a result, a large number of sessions are lost or the cache becomes invalid. Therefore, consistent hashing is introduced to Nginx Ingress to solve this problem.

Consistent hashing is a special hash algorithm, which constructs a ring hash space to replace the common linear hash space. When a node is added or deleted, only the target route is migrated clockwise, and other routes do not need to be changed. In this way, rerouting can be reduced as much as possible, resolving the load balancing issue caused by dynamic node addition and deletion.

If a consistent hashing rule is configured, the newly added server will share the load of all other servers. Similarly, when a server is removed, all other servers can share the load of the removed server. This balances the load among nodes in the cluster and prevents the avalanche effect caused by the breakdown of a node.

## Configuring a Consistent Hashing Rule

Nginx Ingress can use the `nginx.ingress.kubernetes.io/upstream-hash-by` annotation to configure consistent hashing rules. The following is an example:

**Step 1** Use `kubectl` to access the cluster. For details, see [Connecting to a Cluster Using kubectl](#).

**Step 2** Create a YAML file named `ingress-test.yaml`. The file name can be customized.

```
vi ingress-test.yaml
```

### For clusters of v1.23 or later:

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: ingress-test
  namespace: default
  annotations:
    nginx.ingress.kubernetes.io/upstream-hash-by: "$request_uri" # Perform hashing based on the
    request URI.
spec:
  rules:
    - host: ""
      http:
        paths:
          - path: "/"
            backend:
              service:
                name: <your_service_name> # Replace it with the name of your target Service.
                port:
                  number: <your_service_port> # Replace it with the port number of your target Service.
            property:
              ingress.beta.kubernetes.io/url-match-mode: STARTS_WITH
            pathType: ImplementationSpecific
          ingressClassName: nginx
```

### For clusters of v1.21 or earlier:

```
apiVersion: networking.k8s.io/v1beta1
kind: Ingress
metadata:
  name: ingress-test
  namespace: default
  annotations:
    kubernetes.io/ingress.class: nginx
    nginx.ingress.kubernetes.io/upstream-hash-by: "$request_uri" # Perform hashing based on the
    request URI.
spec:
  rules:
    - host: ""
      http:
        paths:
          - path: "/"
            backend:
              serviceName: <your_service_name> # Replace it with the name of your target Service.
              servicePort: <your_service_port> # Replace it with the port number of your target Service.
```

The value of `nginx.ingress.kubernetes.io/upstream-hash-by` can be an nginx variable, a text value, or any combination:

- `nginx.ingress.kubernetes.io/upstream-hash-by: "$request_uri"` indicates that requests are hashed based on the request URI.
- `nginx.ingress.kubernetes.io/upstream-hash-by: "$request_uri$host"` indicates that requests are hashed based on the request URI and domain name.

- **nginx.ingress.kubernetes.io/upstream-hash-by: "\${request\_uri}-text-value"** indicates that requests are hashed based on the request URI and text value.

**Step 3** Create an ingress.

```
kubectl create -f ingress-test.yaml
```

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

```
ingress/ingress-test created
```

**Step 4** Check the created ingress.

```
kubectl get ingress
```

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

NAME	CLASS	HOSTS	ADDRESS	PORTS	AGE
ingress-test	nginx	*	121.**.**.**	80	10s

----End

## Documentation

[Custom NGINX upstream hashing](#)

## 7.5 DNS

### 7.5.1 Overview

#### Introduction to CoreDNS

When you create a cluster, the **CoreDNS add-on** is installed to resolve domain names in the cluster.

You can view the pod of the CoreDNS add-on in the kube-system namespace.

```
$ kubectl get po --namespace=kube-system
NAME                READY STATUS RESTARTS AGE
coredns-7689f8bdf-295rk    1/1 Running 0      9m11s
coredns-7689f8bdf-h7n68    1/1 Running 0      11m
```

After CoreDNS is installed, it becomes a DNS. After the Service is created, CoreDNS records the Service name and IP address. In this way, the pod can obtain the Service IP address by querying the Service name from CoreDNS.

**nginx.<namespace>.svc.cluster.local** is used to access the Service. **nginx** is the Service name, **<namespace>** is the namespace, and **svc.cluster.local** is the domain name suffix. In actual use, you can omit **<namespace>.svc.cluster.local** in the same namespace and use the ServiceName.

An advantage of using ServiceName is that you can write ServiceName into the program when developing the application. In this way, you do not need to know the IP address of a specific Service.

After CoreDNS is installed, there is also a Service in the kube-system namespace, as shown below.



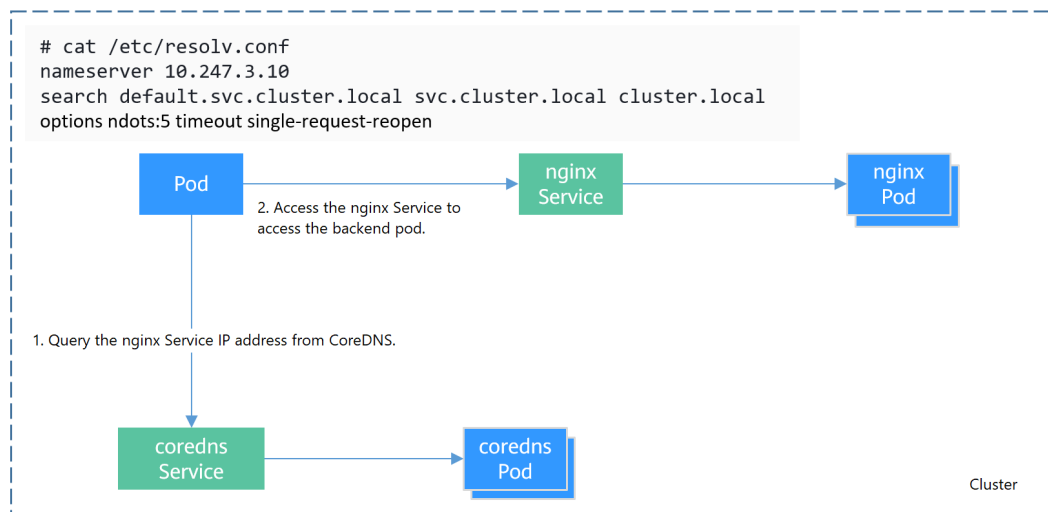
```
$ kubectl get svc -n kube-system
NAME          TYPE        CLUSTER-IP   EXTERNAL-IP   PORT(S)          AGE
coredns      ClusterIP   10.247.3.10  <none>        53/UDP,53/TCP,8080/TCP   13d
```

By default, after other pods are created, the address of the CoreDNS Service is written as the address of the domain name resolution server in the `/etc/resolv.conf` file of the pod. Create a pod and view the `/etc/resolv.conf` file as follows:

```
$ kubectl exec test01-6cbbf97b78-krj6h -it -- /bin/sh
/# cat /etc/resolv.conf
nameserver 10.247.3.10
search default.svc.cluster.local svc.cluster.local cluster.local
options ndots:5 timeout single-request-reopen
```

When a user accesses the `Service name:Port` of the Nginx pod, the IP address of the Nginx Service is resolved from CoreDNS, and then the IP address of the Nginx Service is accessed. In this way, the user can access the backend Nginx pod.

**Figure 7-43** Example of domain name resolution in a cluster



## How Does Domain Name Resolution Work in Kubernetes?

DNS policies can be configured for each pod. Kubernetes supports DNS policies **Default**, **ClusterFirst**, **ClusterFirstWithHostNet**, and **None**. For details, see [DNS for Services and Pods](#). These policies are specified in the `dnsPolicy` field in the pod-specific.

- **Default:** Pods inherit the name resolution configuration from the node that the pods run on. The custom upstream DNS server and the stub domain cannot be used together with this policy.
- **ClusterFirst:** Any DNS query that does not match the configured cluster domain suffix, such as **www.kubernetes.io**, is forwarded to the upstream name server inherited from the node. Cluster administrators may have extra stub domains and upstream DNS servers configured.
- **ClusterFirstWithHostNet:** For pods running with `hostNetwork`, set its DNS policy **ClusterFirstWithHostNet**.
- **None:** It allows a pod to ignore DNS settings from the Kubernetes environment. All DNS settings are supposed to be provided using the `dnsPolicy` field in the pod-specific.

**NOTE**

- Clusters of Kubernetes v1.10 and later support **Default**, **ClusterFirst**, **ClusterFirstWithHostNet**, and **None**. Clusters earlier than Kubernetes v1.10 support only **Default**, **ClusterFirst**, and **ClusterFirstWithHostNet**.
- **Default** is not the default DNS policy. If **dnsPolicy** is not explicitly specified, **ClusterFirst** is used.

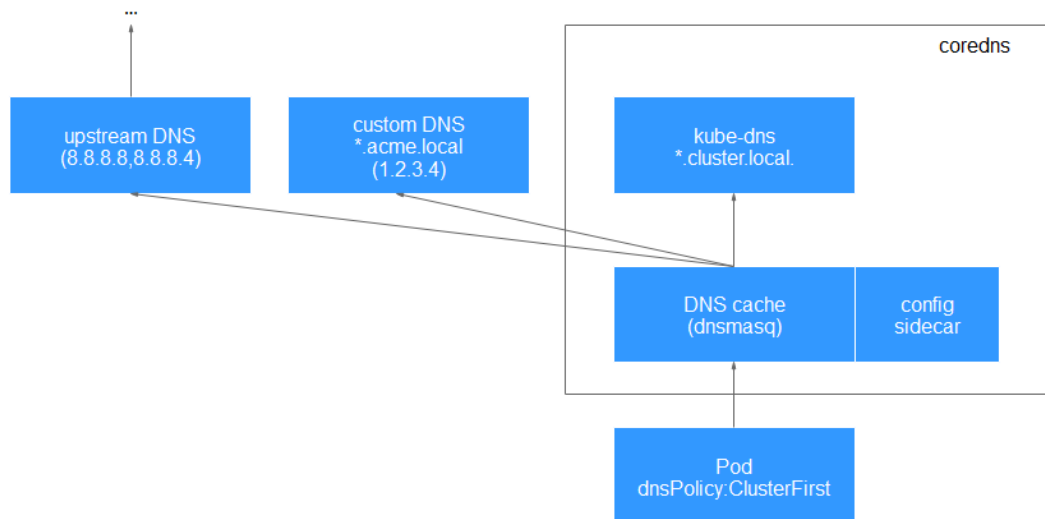
**Routing**

**Without stub domain configurations:** Any query that does not match the configured cluster domain suffix, such as **www.kubernetes.io**, is forwarded to the upstream DNS server inherited from the node.

**With stub domain configurations:** If stub domains and upstream DNS servers are configured, DNS queries are routed according to the following flow:

1. The query is first sent to the DNS caching layer in CoreDNS.
2. From the caching layer, the suffix of the request is examined and then the request is forwarded to the corresponding DNS:
  - Names with the cluster suffix, for example, **.cluster.local**: The request is sent to CoreDNS.
  - Names with the stub domain suffix, for example, **.acme.local**: The request is sent to the configured custom DNS resolver that listens, for example, on 1.2.3.4.
  - Names that do not match the suffix (for example, **widget.com**): The request is forwarded to the upstream DNS.

**Figure 7-44** Routing



**Related Operations**

You can also configure DNS in a workload. For details, see [DNS Configuration](#).

You can also use CoreDNS to implement user-defined domain name resolution. For details, see [Using CoreDNS for Custom Domain Name Resolution](#).

You can also use DNSCache to improve the DNS resolution performance. For details, see [Using NodeLocal DNSCache to Improve DNS Performance](#).

## 7.5.2 DNS Configuration

Every Kubernetes cluster has a built-in DNS add-on (Kube-DNS or CoreDNS) to provide domain name resolution for workloads in the cluster. When handling a high concurrency of DNS queries, Kube-DNS/CoreDNS may encounter a performance bottleneck, that is, it may fail occasionally to fulfill DNS queries. There are cases when Kubernetes workloads initiate unnecessary DNS queries. This makes DNS overloaded if there are many concurrent DNS queries. Tuning DNS configuration for workloads will reduce the risks of DNS query failures to some extent.

For more information about DNS, see [CoreDNS](#).

### DNS Configuration Items

Run the `cat /etc/resolv.conf` command on a Linux node or container to view the DNS resolver configuration file. The following is an example DNS resolver configuration of a container in a Kubernetes cluster:

```
nameserver 10.247.x.x
search default.svc.cluster.local svc.cluster.local cluster.local
options ndots:5
```

#### Configuration Options

- **nameserver:** an IP address list of a name server that the resolver will query. If this parameter is set to 10.247.x.x, the resolver will query the kube-dns/CoreDNS. If this parameter is set to another IP address, the resolver will query a cloud or on-premises DNS server.
- **search:** a search list for host-name lookup. When a domain name cannot be resolved, DNS queries will be attempted combining the domain name with each domain in the search list in turn until a match is found or all domains in the search list are tried. For CCE clusters, the search list is currently limited to three domains per container. When a nonexistent domain name is being resolved, eight DNS queries will be initiated because each domain name (including those in the search list) will be queried twice, one for IPv4 and the other for IPv6.
- **options:** options that allow certain internal resolver variables to be modified. Common options include timeout and ndots.

The value **ndots:5** means that if a domain name has fewer than 5 dots (.), DNS queries will be attempted by combining the domain name with each domain in the search list in turn. If no match is found after all the domains in the search list are tried, the domain name is then used for DNS query. If the domain name has 5 or more than 5 dots, it will be tried first for DNS query. In case that the domain name cannot be resolved, DNS queries will be attempted by combining the domain name with each domain in the search list in turn.

For example, the domain name **www.\*\*\*.com** has only two dots (smaller than the value of **ndots**), and therefore the sequence of DNS queries is as follows: **www.\*\*\*.com.default.svc.cluster.local**, **www.\*\*\*.com.svc.cluster.local**, **www.\*\*\*.com.cluster.local**, and **www.\*\*\*.com**. This means that at least seven DNS queries will be initiated before the domain name is resolved into an IP

address. It is clear that when many unnecessary DNS queries will be initiated to access an external domain name. There is room for improvement in workload's DNS configuration.

#### NOTE

For more information about configuration options in the resolver configuration file used by Linux operating systems, visit <http://man7.org/linux/man-pages/man5/resolv.conf.5.html>.

## Configuring DNS for a Workload Using the Console

Kubernetes provides DNS-related configuration options for applications. The use of application's DNS configuration can effectively reduce unnecessary DNS queries in certain scenarios and improve service concurrency. The following procedure uses an Nginx application as an example to describe how to add DNS configurations for a workload on the console.

- Step 1** Log in to the CCE console, access the cluster console, select **Workloads** in the navigation pane, and click **Create Workload** in the upper right corner.
- Step 2** Configure basic information about the workload. For details, see [Creating a Workload](#).
- Step 3** In the **Advanced Settings** area, click the **DNS** tab and set the following parameters as required:
  - **DNS Policy:** The DNS policies provided on the console correspond to the **dnsPolicy** field in the YAML file. For details, see [Table 7-78](#).
    - **Supplement defaults:** corresponds to **dnsPolicy=ClusterFirst**. Containers can resolve both the cluster-internal domain names registered by a Service and the external domain names exposed to public networks.
    - **Replace defaults:** corresponds to **dnsPolicy=None**. You must configure **IP Address** and **Search Domain**. Containers only use the user-defined IP address and search domain configurations for domain name resolution.
    - **Inherit defaults:** corresponds to **dnsPolicy=Default**. Containers use the domain name resolution configuration from the node that pods run on and cannot resolve the cluster-internal domain names.
  - **Optional Objects:** The options parameters in the **dnsConfig** field. Each object may have a name property (required) and a value property (optional). After setting the properties, click **confirm to add**.
    - **timeout:** Timeout interval, in seconds.
    - **ndots:** Number of dots (.) that must be present in a domain name. If a domain name has dots fewer than this value, the operating system will look up the name in the search domain. If not, the name is a fully qualified domain name (FQDN) and will be tried first as an absolute name.
  - **IP Address of DNS Server:** **nameservers** in **dnsConfig**. You can configure a domain name server for a custom domain name. The value is one or a group of DNS IP addresses.
  - **Search Domain:** **searches** in the **dnsConfig**. 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 search domain names generated from the chosen DNS policy in **dnsPolicy**. Duplicate domain names are removed.

**Step 4 Click Create Workload.**

----End

## Configuring DNS Using the Workload YAML

When creating a workload using a YAML file, you can configure the DNS settings in the YAML. The following is an example for an Nginx application:

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx
  namespace: default
spec:
  replicas: 1
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
        - name: container-1
          image: nginx:latest
          imagePullPolicy: IfNotPresent
      imagePullSecrets:
        - name: default-secret
      dnsPolicy: None
      dnsConfig:
        options:
          - name: ndots
            value: '5'
          - name: timeout
            value: '3'
      nameservers:
        - 10.2.3.4
      searches:
        - my.dns.search.suffix
```

- **dnsPolicy**

The **dnsPolicy** field is used to configure a DNS policy for an application. The default value is **ClusterFirst**. The following table lists **dnsPolicy** configurations.

**Table 7-78** dnsPolicy

Parameter	Description
ClusterFirst (default value)	Custom DNS configuration added to the default DNS configuration. By default, the application connects to CoreDNS (CoreDNS of the CCE cluster connects to the DNS on the cloud by default). The custom dnsConfig will be added to the default DNS parameters. Containers can resolve both the cluster-internal domain names registered by a Service and the external domain names exposed to public networks. The search list ( <b>search</b> option) and <b>ndots: 5</b> are present in the DNS configuration file. Therefore, when accessing an external domain name and a long cluster-internal domain name (for example, kubernetes.default.svc.cluster.local), the search list will usually be traversed first, resulting in at least six invalid DNS queries. The issue of invalid DNS queries disappears only when a short cluster-internal domain name (for example, kubernetes) is being accessed.
ClusterFirstWithHostNet	By default, the applications configured with the <b>host network</b> are interconnected with the DNS configuration of the node where the pod is located. The DNS configuration is specified in the DNS file that the kubelet <b>--resolv-conf</b> parameter points to. In this case, the CCE cluster uses the DNS on the cloud. If workloads need to use Kube-DNS/CoreDNS of the cluster, set <b>dnsPolicy</b> to <b>ClusterFirstWithHostNet</b> and container's DNS configuration file is the same as ClusterFirst, in which invalid DNS queries still exist. ... spec: containers: - image: nginx:latest imagePullPolicy: IfNotPresent name: container-1 restartPolicy: Always <b>hostNetwork: true</b> <b>dnsPolicy: ClusterFirstWithHostNet</b>
Default	The DNS configuration of the node where the pod is located is inherited, and the custom DNS configuration is added to the inherited configuration. Container's DNS configuration file is the DNS configuration file that the kubelet's <b>--resolv-conf</b> flag points to. In this case, a cloud DNS is used for CCE clusters. Both <b>search</b> and <b>options</b> fields are left unspecified. This configuration can only resolve the external domain names registered with the Internet, and not cluster-internal domain names. This configuration is free from the issue of invalid DNS queries.
None	The default DNS configuration is replaced by the custom DNS configuration, and only the custom DNS configuration is used. If <b>dnsPolicy</b> is set to <b>None</b> , the <b>dnsConfig</b> field must be specified because all DNS settings are supposed to be provided using the <b>dnsConfig</b> field.

 NOTE

If the **dnsPolicy** field is not specified, the default value is **ClusterFirst** instead of **Default**.

- **dnsConfig**

The **dnsConfig** field is used to configure DNS parameters for workloads. The configured parameters are merged to the DNS configuration file generated according to **dnsPolicy**. If **dnsPolicy** is set to **None**, the workload's DNS configuration file is specified by the **dnsConfig** field. If **dnsPolicy** is not set to **None**, the DNS parameters configured in **dnsConfig** are added to the DNS configuration file generated according to **dnsPolicy**.

**Table 7-79** dnsConfig

Parameter	Description
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 in <b>dnsPolicy</b> . Duplicate entries are removed.
nameservers	A list of IP addresses that will be used as DNS servers. If workload's <b>dnsPolicy</b> is set to <b>None</b> , the list must contain at least one IP address, otherwise this property is optional. The servers listed will be combined to the nameservers generated from the specified DNS policy in <b>dnsPolicy</b> with duplicate addresses removed.  <b>NOTE</b> A maximum of three DNS addresses can be configured for a nameserver in the container DNS configuration file. <ul style="list-style-type: none"> <li>• If <b>dnsPolicy</b> is set to <b>ClusterFirst</b> and the cluster uses <b>CoreDNS</b>, you can add two custom DNS addresses in addition to the CoreDNS address. Excess DNS addresses are invalid.</li> <li>• If <b>dnsPolicy</b> is set to <b>ClusterFirst</b> and the cluster uses <b>CoreDNS</b> and <b>NodeLocal DNSCache</b>, you can add one custom DNS address in addition to the CoreDNS and NodeLocal DNSCache addresses. Excess DNS addresses are invalid.</li> </ul>
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 search domain names generated from the chosen DNS policy in <b>dnsPolicy</b> . Duplicate domain names are removed. Kubernetes allows for at most 6 search domains.

## Configuration Examples

The following example describes how to configure DNS for workloads.

- **Use Case 1: Using Kube-DNS/CoreDNS Built in Kubernetes Clusters**

**Scenario**

Kubernetes in-cluster Kube-DNS/CoreDNS applies to resolving only cluster-internal domain names or cluster-internal domain names + external domain names. This is the default DNS for workloads.

**Example:**

```
apiVersion: v1
kind: Pod
metadata:
  namespace: default
  name: dns-example
spec:
  containers:
  - name: test
    image: nginx:alpine
  dnsPolicy: ClusterFirst
  imagePullSecrets:
  - name: default-secret
```

Container's DNS configuration file:

```
nameserver 10.247.3.10
search default.svc.cluster.local svc.cluster.local cluster.local
options ndots:5
```

- **Use Case 2: Using a Cloud DNS**

**Scenario**

A DNS cannot resolve cluster-internal domain names and therefore applies to the scenario where workloads access only external domain names registered with the Internet.

**Example:**

```
apiVersion: v1
kind: Pod
metadata:
  namespace: default
  name: dns-example
spec:
  containers:
  - name: test
    image: nginx:alpine
  dnsPolicy: Default # The DNS configuration file that the kubelet --resolv-conf parameter points to
  is used. In this case, the CCE cluster uses the DNS on the cloud.
  imagePullSecrets:
  - name: default-secret
```

Container's DNS configuration file:

```
nameserver 100.125.x.x
```

- **Use Case 3: Using Kube-DNS/CoreDNS for Workloads Running with hostNetwork**

**Scenario**

By default, a DNS is used for workloads running with hostNetwork. If workloads need to use Kube-DNS/CoreDNS, set **dnsPolicy** to **ClusterFirstWithHostNet**.

**Example:**

```
apiVersion: v1
kind: Pod
metadata:
  name: nginx
spec:
  hostNetwork: true
```



**dnsPolicy: ClusterFirstWithHostNet**

```
containers:
- name: nginx
  image: nginx:alpine
  ports:
  - containerPort: 80
imagePullSecrets:
- name: default-secret
```

Container's DNS configuration file:

```
nameserver 10.247.3.10
search default.svc.cluster.local svc.cluster.local cluster.local
options ndots:5
```

- **Use Case 4: Customizing Application's DNS Configuration**

**Scenario**

You can flexibly customize the DNS configuration file for applications. Using **dnsPolicy** and **dnsConfig** together can address almost all scenarios, including the scenarios in which an on-premises DNS will be used, multiple DNSs will be cascaded, and DNS configuration options will be modified.

**Example 1: Using Your On-Premises DNS**

Set **dnsPolicy** to **None** so application's DNS configuration file is generated based on **dnsConfig**.

```
apiVersion: v1
kind: Pod
metadata:
  namespace: default
  name: dns-example
spec:
  containers:
  - name: test
    image: nginx:alpine
  dnsPolicy: "None"
  dnsConfig:
    nameservers:
    - 10.2.3.4 # IP address of your on-premises DNS
    searches:
    - ns1.svc.cluster.local
    - my.dns.search.suffix
    options:
    - name: ndots
      value: "2"
    - name: timeout
      value: "3"
  imagePullSecrets:
  - name: default-secret
```

Container's DNS configuration file:

```
nameserver 10.2.3.4
search ns1.svc.cluster.local my.dns.search.suffix
options timeout:3 ndots:2
```

**Example 2: Modifying the ndots Option in the DNS Configuration File to Reduce Invalid DNS Queries**

Set **dnsPolicy** to a value other than **None** so the DNS parameters configured in **dnsConfig** are added to the DNS configuration file generated based on **dnsPolicy**.

```
apiVersion: v1
kind: Pod
metadata:
  namespace: default
  name: dns-example
spec:
  containers:
```

```
- name: test
  image: nginx:alpine
  dnsPolicy: "ClusterFirst"
  dnsConfig:
    options:
      - name: ndots
        value: "2" # The ndots:5 option in the DNS configuration file generated based on the
ClusterFirst policy is changed to ndots:2.
  imagePullSecrets:
    - name: default-secret
```

Container's DNS configuration file:

```
nameserver 10.247.3.10
search default.svc.cluster.local svc.cluster.local cluster.local
options ndots:2
```

### Example 3: Using Multiple DNSs in Serial Sequence

```
apiVersion: v1
kind: Pod
metadata:
  namespace: default
  name: dns-example
spec:
  containers:
    - name: test
      image: nginx:alpine
      dnsPolicy: ClusterFirst # Added DNS configuration. The cluster connects to CoreDNS by default.
      dnsConfig:
        nameservers:
          - 10.2.3.4 # IP address of your on-premises DNS
  imagePullSecrets:
    - name: default-secret
```

#### NOTE

A maximum of three DNS addresses can be configured for a nameserver in the container DNS configuration file.

- If **dnsPolicy** is set to **ClusterFirst** and the cluster uses **CoreDNS**, you can add two custom DNS addresses in addition to the CoreDNS address. Excess DNS addresses are invalid.
- If **dnsPolicy** is set to **ClusterFirst** and the cluster uses **CoreDNS** and **NodeLocal DNSCache**, you can add one custom DNS address in addition to the CoreDNS and NodeLocal DNSCache addresses. Excess DNS addresses are invalid.

Container's DNS configuration file:

```
nameserver 10.247.3.10 10.2.3.4
search default.svc.cluster.local svc.cluster.local cluster.local
options ndots:5
```

## 7.5.3 Using CoreDNS for Custom Domain Name Resolution

### Challenges

When using CCE, you may need to resolve custom internal domain names in the following scenarios:

- In the legacy code, a fixed domain name is configured for calling other internal services. If the system decides to use Kubernetes Services, the code refactoring workload could be heavy.
- A service is created outside the cluster. Data in the cluster needs to be sent to the service through a fixed domain name.

## Solution

There are several CoreDNS-based solutions for custom domain name resolution:

- **Configuring the Stub Domain for CoreDNS:** You can configure a DNS server for a specific domain name and perform this operation on the console.
- **Using the CoreDNS Hosts plug-in to configure resolution for any domain name:** You can add any record set for a specific domain name, which is similar to adding a record set in the local `/etc/hosts` file.
- **Using the CoreDNS Rewrite plug-in to point a domain name to ClusterIP Services:** Perform CNAME resolution on a domain name in the cluster and redirect it to another domain name in the cluster. This is like giving an alias to the Kubernetes Service, and you do not have to get the IP address of the record set beforehand.
- **Using the CoreDNS Forward plug-in to set the on-premises DNS as the upstream DNS:** Use the on-premises DNS as the external DNS server. You can manage a large number of records on the on-premises DNS, without modifying the CoreDNS configuration when adding or deleting records

## Precautions

Improper modification on CoreDNS configuration may cause domain name resolution failures in the cluster. Perform tests before and after the modification.

## Configuring the Stub Domain for CoreDNS

Cluster administrators can modify the ConfigMap for the CoreDNS Corefile to change how service discovery works.

Assume that a cluster administrator has a Consul DNS server located at 10.150.0.1 and all Consul domain names have the suffix `.consul.local`.

- 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 7-45** Adding a stub domain

Stub Domain	Delete
consul.local	10.150.0.1

+ Add

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 IP 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 listening at 1.2.3.4,6.7.8.9.

### 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 {
  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

## Modifying the CoreDNS Hosts Configuration File

After modifying the hosts file in CoreDNS, you do not need to configure the hosts file in each pod to add resolution records.

**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:

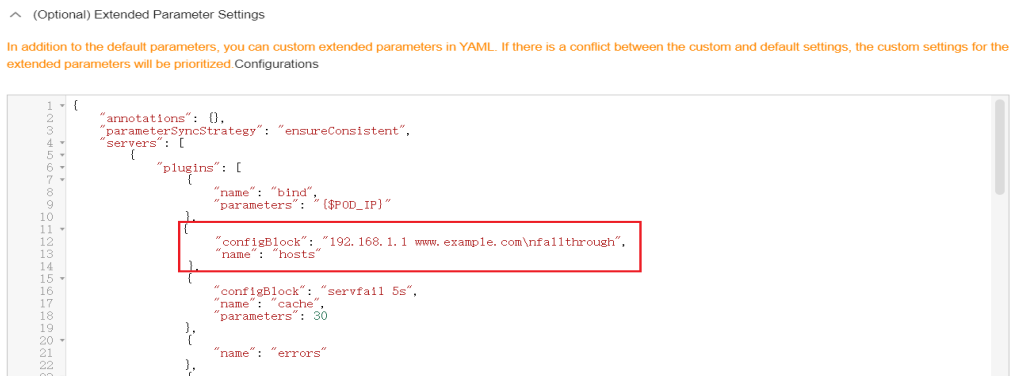
```
{
  "configBlock": "192.168.1.1 www.example.com\nfallthrough",
  "name": "hosts"
}
```

### 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/>.

**Figure 7-46** Modifying the CoreDNS hosts configuration



**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

## Adding the CoreDNS Rewrite Configuration to Point the Domain Name to ClusterIP Services

Use the Rewrite plug-in of CoreDNS to resolve a specified domain name to the domain name of a Service. For example, the request for accessing the example.com domain name is redirected to the example.default.svc.cluster.local domain name, that is, the example service in the default namespace.

**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:

```
{
  "name": "rewrite",
  "parameters": "name example.com example.default.svc.cluster.local"
}
```

**Figure 7-47** Adding the CoreDNS rewrite configuration to point the domain name to ClusterIP Services

^ (Optional) Extended Parameter Settings

In addition to the default parameters, you can custom extended parameters in YAML. If there is a conflict between the custom and default settings, the custom settings for the extended parameters will be prioritized. Configurations

```
1 + {
2   "annotations": {},
3   "parameterSyncStrategy": "ensureConsistent",
4   "servers": [
5     {
6       "plugins": [
7         {
8           "name": "bind",
9           "parameters": "${POD_IP}"
10        },
11        {
12          "name": "rewrite",
13          "parameters": "name example.com example.default.svc.cluster.local"
14        },
15      ],
16      "configBlock": "servfail 5s",
17      "name": "cache",
18      "parameters": 30
19    },
20    {
21      "name": "errors"
22    }
23  ]
}
```

**Corefile:**

```
.:53 {
  bind {${POD_IP}
  rewrite name example.com example.default.svc.cluster.local
  cache 30
  errors
  health {${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

### Using CoreDNS to Cascade On-premises DNS

If the domain name is not in the Kubernetes domain, CoreDNS will use the **/etc/resolv.conf** file of the node for resolution by default. You can also change the resolution address to that of the external 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 modify the following content in the **plugins** field:

```
{
  "configBlock": "policy random",
  "name": "forward",
  "parameters": ". 192.168.1.1"
}
```

**Figure 7-48** Using CoreDNS to cascade on-premises DNS

^ (Optional) Extended Parameter Settings

In addition to the default parameters, you can custom extended parameters in YAML. If there is a conflict between the custom and default settings, the custom settings for the extended parameters will be prioritized. Configurations

```
41 |         "name": "prometheus",
42 |         "parameters": "${POD_IP}:9153"
43 |       },
44 |       {
45 |         "configBlock": "policy random",
46 |         "name": "forward",
47 |         "parameters": ". 192.168.1.1"
48 |       },
49 |       {
50 |         "name": "reload"
51 |       }
52 |     ],
53 |     "port": 5353,
54 |     "zones": [
55 |       {
56 |         "zone": "."
57 |       }
58 |     ]
59 |   },
60 |   "upstream_nameservers": [],
61 |   "extraConfig": {}
62 | }
63 |
```

Corefile:

```
.: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 . 192.168.1.1 {
    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

## 7.5.4 Using NodeLocal DNSCache to Improve DNS Performance

### 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.

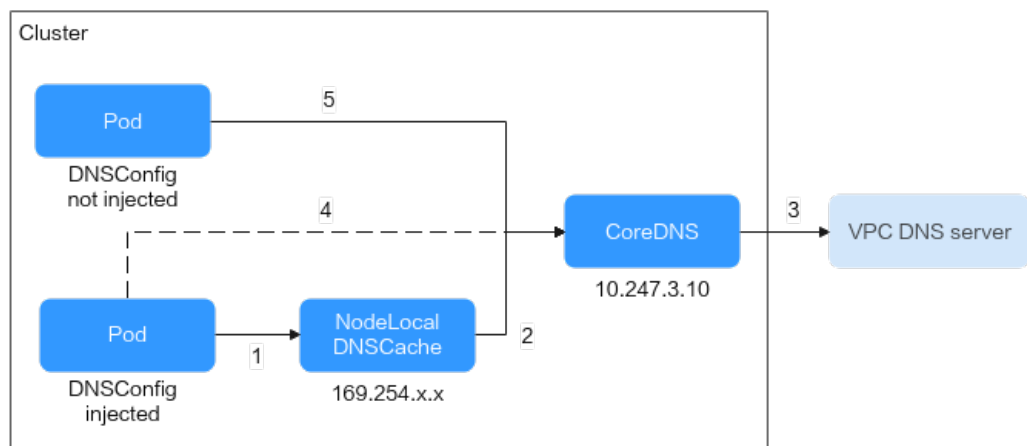
- Increased resource usage: To ensure DNS performance, CoreDNS requires higher specifications.

## Solution

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's load and improves the cluster DNS performance.

After NodeLocal DNSCache is enabled, a DNS query goes through the path as shown below.

**Figure 7-49** NodeLocal DNSCache query path



The resolution rules are as follows:

- 1. By default, the pods with DNSConfig injected use NodeLocal DNSCache to resolve requested domain names.
- 2. If NodeLocal DNSCache cannot resolve domain names, it will ask CoreDNS for resolution.
- 3. CoreDNS uses the DNS server in the VPC to resolve the domain names out of the cluster.
- 4. If a pod with DNSConfig injected cannot access NodeLocal DNSCache, CoreDNS will resolve the domain name.
- 5. By default, CoreDNS resolves domain names for the pods without DNSConfig injected.

## Notes and Constraints

- Only clusters of version 1.19 or later support the **NodeLocal DNSCache** add-on.
- The **node-local-dns-injection** label is the system label used by NodeLocal DNSCache. Use this label only to **prevent an automatic DNSConfig injection**.



## Installing the Add-on

CCE provides add-on **NodeLocal DNSCache** that simplifies the installation of NodeLocal DNSCache.

### NOTE

NodeLocal DNSCache serves as a transparent caching proxy for CoreDNS and does not provide plug-ins such as hosts or rewrite. If you want to enable these plug-ins, modify the CoreDNS configurations.

**Step 1** (Optional) Modify the CoreDNS configuration so that the CoreDNS preferentially uses UDP to communicate with the upstream DNS server.

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. 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.

Perform the following operations. In the forward add-on, specify **prefer\_udp** as the protocol used by requests. After the modification, CoreDNS preferentially uses UDP to communicate with the upstream system.

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
}
```

**Step 2** Log in to the CCE console and click the cluster name to access the cluster console. Choose **Add-ons** in the navigation pane, locate **NodeLocal DNSCache** on the right, and click **Install**.

**Step 3** On the **Install Add-on** page, select the add-on specifications and set related parameters.

- **DNSConfig Automatic Injection:** After this function is enabled, a DNSConfig admission controller will be created. The controller intercepts pod creation requests in the namespace labeled with **node-local-dns-injection=enabled** based on admission webhooks and automatically configures DNSConfig for pods. If this function is disabled or the pod belongs to a non-target namespace, you must manually configure DNSConfig for the pod.

**Step 4** Click **Install**.

----End

## Using NodeLocal DNSCache

By default, application requests are sent through the CoreDNS proxy. To use node-local-dns as the DNS cache proxy, use any of the following methods:

- Auto injection: Automatically configure the **dnsConfig** field of the pod when creating the pod. (This function is not available for pods in system namespaces such as kube-system.)
- Manual configuration: Manually configure the **dnsConfig** field of the pod.

## Auto Injection

To enable auto injection, perform the following operations:

**Step 1** Install the NodeLocal DNSCache add-on and enable auto DNSConfig injection. For details, see [Installing the Add-on](#).

**Step 2** Use kubectl to access the cluster and add the **node-local-dns-injection=enabled** label to the namespace. For example, run the following command to add the label to the **default** namespace:

```
kubectl label namespace default node-local-dns-injection=enabled
```

**Step 3** Create a workload in the namespace with auto injection enabled. For details, see [Creating a Deployment](#).

---

### NOTICE

The pods for which auto injection is enabled must meet the following requirements:

- The new pod does not run in system namespaces such as kube-system and kube-public.
- The **node-local-dns-injection=disabled** label for disabling DNS injection is not added to the new pod.
- The new pod's **DNSPolicy** is **ClusterFirstWithHostNet**. Alternatively, the pod does not use the host network and **DNSPolicy** is **ClusterFirst**.

---

**Step 4** Check the pod configuration.

```
kubectl get pod <pod_name> -oyaml
```

After auto injection is enabled, the following **dnsConfig** settings will be automatically added to the created pod. In addition to the NodeLocal DNSCache IP address 169.254.20.10, the CoreDNS IP address 10.247.3.10 is added to **nameservers**, ensuring high availability of the service DNS server.

```
...
dnsConfig:
  nameservers:
    - 169.254.20.10
    - 10.247.3.10
  searches:
    - default.svc.cluster.local
    - svc.cluster.local
    - cluster.local
  options:
    - name: timeout
      value: ""
    - name: ndots
      value: '5'
    - name: single-request-reopen
...

```

----End

## Manual Configuration

Manually add the **dnsConfig** settings to pods.

- Step 1** Use `kubectl` to access the cluster. For details, see [Connecting to a Cluster Using kubectl](#).
- Step 2** Create a pod and add the NodeLocal DNSCache IP address 169.254.20.10 to **nameservers** in **dnsConfig**.

Create a **nginx.yaml** file. The following shows an example:

```
apiVersion: v1
kind: Pod
metadata:
  name: nginx
spec:
  containers:
    - image: nginx:alpine
      name: container-0
  dnsConfig:
    nameservers:
      - 169.254.20.10
      - 10.247.3.10
    searches:
      - default.svc.cluster.local
      - svc.cluster.local
      - cluster.local
    options:
      - name: ndots
        value: '2'
  imagePullSecrets:
    - name: default-secret

```

- Step 3** Run the following command to create a pod:

```
kubectl create -f nginx.yaml
```

----End

## Common Issues

- How Do I Prevent Auto DNSConfig Injection on a Specific Pod?

### Solution:

Add **node-local-dns-injection: disabled** to the **labels** field in the pod template. Example:

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: test
  namespace: default
spec:
  replicas: 2
  selector:
    matchLabels:
      app: test
  template:
    metadata:
      labels:
        app: test
        node-local-dns-injection: disabled # Prevent auto DNSConfig injection.
    spec:
      containers:
        - name: container-1
          image: nginx:latest
          imagePullPolicy: IfNotPresent
      imagePullSecrets:
        - name: default-secret
```

## 7.6 Configuring Intra-VPC Access

This section describes how to access an intranet from a container (outside the cluster in a VPC), including intra-VPC access and cross-VPC access.

### Intra-VPC Access

The performance of accessing an intranet from a container varies depending on the container network models of a cluster.

- **Container tunnel network**

The container tunnel network encapsulates network data packets through tunnels based on the node network. A container can access other resources in the same VPC as long as the node can access the resources. If the access fails, check whether the security group of the peer resource allows access from the node where the container is located.

- **Cloud Native Network 2.0**

In the Cloud Native Network 2.0 model, a container is assigned an IP address from the CIDR block of a VPC. The container CIDR block is the subnet of the VPC where the node is located. The container can naturally communicate with other addresses in the VPC. If the access fails, check whether the security group of peer resources allows the access from the container CIDR block.

- **VPC network**

The VPC network model uses VPC routes to forward container traffic. The container CIDR block and the node VPC are not in the same CIDR block. When a container accesses other resources in the same VPC, **the security group of the peer resource must allow access of the container CIDR block.**

For example, the CIDR block where the cluster node resides is 192.168.10.0/24, and the container CIDR block is 172.16.0.0/16.

There is an ECS whose IP address is 192.168.10.52 in the VPC (outside the cluster). The security group of the ECS allows access of only the CIDR block of the cluster node.

In this case, if you ping 192.168.10.52 from the container, the ping operation fails.

```
kubectl exec test01-6cbbf97b78-krj6h -it -- /bin/sh
/ # ping 192.168.10.25
PING 192.168.10.25 (192.168.10.25): 56 data bytes
^C
--- 192.168.10.25 ping statistics ---
104 packets transmitted, 0 packets received, 100% packet loss
```

Configure the security group to allow access from the container CIDR block 172.16.0.0/16.

In this case, 192.168.10.52 can be pinged from the container.

```
$ kubectl exec test01-6cbbf97b78-krj6h -it -- /bin/sh
/ # ping 192.168.10.25
PING 192.168.10.25 (192.168.10.25): 56 data bytes
64 bytes from 192.168.10.25: seq=0 ttl=64 time=1.412 ms
64 bytes from 192.168.10.25: seq=1 ttl=64 time=1.400 ms
64 bytes from 192.168.10.25: seq=2 ttl=64 time=1.299 ms
64 bytes from 192.168.10.25: seq=3 ttl=64 time=1.283 ms
^C
--- 192.168.10.25 ping statistics ---
4 packets transmitted, 4 packets received, 0% packet loss
```

## Cross-VPC Access

Cross-VPC access is implemented by establishing a peering connection between VPCs.

- In the container tunnel network model, a container can access the peer VPC only when the communication is enabled between the node network and the peer VPC.
- Cloud Native Network 2.0 is similar to the container tunnel network. You only need to enable the communication between the subnet where the container is located and the peer VPC.
- Each VPC network has an independent container CIDR block. In addition to the VPC CIDR block, the container CIDR block also needs to be connected.

Assume that there are two VPCs.

- vpc-demo: Its CIDR block is 192.168.0.0/16, the cluster is in vpc-demo, and the container CIDR block is 10.0.0.0/16.
- vpc-demo2: Its CIDR block is 10.1.0.0/16.

Create a peering connection named **peering-demo** (the local VPC is vpc-demo and the peer VPC is vpc-demo2). Add the container CIDR block to the route of the peer VPC.

After this configuration, you can access the container CIDR block 10.0.0.0/16 in vpc-demo2. During the access, pay attention to the security group configuration and enable the port configuration.

## Accessing Other Cloud Services

Common services that communicate with CCE through an intranet include RDS, DCS, Kafka, RabbitMQ, and ModelArts.

In addition to the network configurations described in [Intra-VPC Access](#) and [Cross-VPC Access](#), you also need to check **whether these cloud services allow external access**. For example, the DCS Redis instance can be accessed only by the IP addresses in its whitelist. Generally, these cloud services can be accessed by IP addresses in the same VPC. However, the container CIDR block in the VPC network model is different from the CIDR block of the VPC. Therefore, you must add the container CIDR block to the whitelist.

## What If a Container Fails to Access an Intranet?

If an intranet cannot be accessed from a container, perform the following operations:

1. View the security group rule of the peer server to check whether the container is allowed to access the peer server.
  - The container tunnel network model needs to allow the IP address of the node where the container is located.
  - The VPC network model needs to allow the container CIDR block.
  - The Cloud Native Network 2.0 model needs to allow the subnet where the container is located.
2. Check whether a whitelist is configured for the peer server. For example, the DCS Redis instance can be accessed only by the IP addresses in its whitelist. Add the container and node CIDR blocks to the whitelist.
3. Check whether the container engine is installed on the peer server and whether it conflicts with the container CIDR block in CCE. If a network conflict occurs, the access fails.

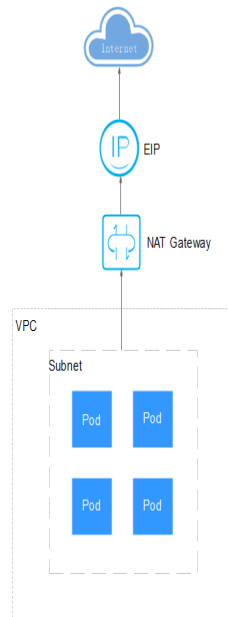
## 7.7 Accessing the Internet from a Container

Containers can access the Internet in either of the following ways:

- Bind an EIP to the node where the container is located.
- Configure SNAT rules through NAT Gateway.



You can use NAT Gateway to enable container pods in a VPC to access the Internet. NAT Gateway provides source network address translation (SNAT), which translates private IP addresses to a public IP address by binding an elastic IP address (EIP) to the gateway, providing secure and efficient access to the Internet. [Figure 7-50](#) shows the SNAT architecture. The SNAT function allows the container pods in a VPC to access the Internet without being bound to an EIP. SNAT supports a large number of concurrent connections, which makes it suitable for applications involving a large number of requests and connections.

Figure 7-50 SNAT



To enable a container to access the Internet, perform the following steps:


**Step 1** Obtain an EIP.

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. Click  in the upper left corner and choose **Networking** > **Elastic IP** in the expanded list.
4. On the **EIPs** page, click **Buy EIP**.
5. Configure parameters as required.

 **NOTE**

Set **Region** to the region where container pods are located.

**Step 2** Create a NAT gateway.

1. Click  in the upper left corner and choose **Networking** > **NAT Gateway** in the expanded list.
2. On the **Public Network Gateways** page, click **Buy Public NAT Gateway** in the upper right corner.
3. Configure parameters as required.

 **NOTE**

Select the same VPC.

**Step 3** Configure an SNAT rule and bind the EIP to the subnet.

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. Set parameters as required.

 **NOTE**

SNAT rules take effect by CIDR block. As different container network models use different communication modes, the subnet needs to be selected according to the following rules:

- Tunnel network and VPC network: Select the subnet where the node is located, that is, the subnet selected during node creation.

If there are multiple CIDR blocks, you can create multiple SNAT rules or customize a CIDR block as long as the CIDR block contains the node subnet.

After the SNAT rule is configured, workloads can access the Internet from the container. The Internet can be pinged from the container.

----**End**



# 8 Storage

## 8.1 Overview

### Container Storage

CCE container storage is implemented based on Kubernetes container storage APIs ([CSI](#)). CCE integrates multiple types of cloud storage and covers different application scenarios. CCE is fully compatible with Kubernetes native storage services, such as `emptyDir`, `hostPath`, `secret`, and `ConfigMap`.

CCE allows workload pods to use multiple types of storage:

- In terms of implementation, storage supports Container Storage Interface (CSI) and Kubernetes native storage.

Type	Description
CSI	An <b>out-of-tree</b> volume add-on, which specifies the standard container storage API and allows storage vendors to use standard custom storage plugins that are mounted using PVCs and PVs without the need to add their plugin source code to the Kubernetes repository for unified build, compilation, and release. CSI is a recommended in Kubernetes 1.13 and later versions.
Kubernetes native storage	An "in-tree" volume add-on that is built, compiled, and released with the Kubernetes repository.

- In terms of storage media, storage can be classified as cloud storage, local storage, and Kubernetes resource objects.

Type	Description	Application Scenario
Cloud storage	The storage media is provided by storage vendors. Storage volumes of this type are mounted using PVCs and PVs.	Data requires high availability or needs to be shared, for example, logs and media resources.  Select a proper cloud storage type based on the application scenario. For details, see <a href="#">Cloud Storage Comparison</a> .
Local storage	The storage media is the local data disk or memory of the node. The local PV is a customized storage type provided by CCE and mounted using PVCs and PVs through the CSI. Other storage types are Kubernetes native storage.	Non-HA data requires high I/O and low latency.  Select a proper local storage type based on the application scenario. For details, see <a href="#">Local Storage Comparison</a> .
Kubernetes resource objects	ConfigMaps and secrets are resources created in clusters. They are special storage types and are provided by tmpfs (RAM-based file system) on the Kubernetes API server.	ConfigMaps are used to inject configuration data to pods.  Secrets are used to transmit sensitive information such as passwords to pods.

## Cloud Storage Comparison

Item	EVS	SFS Turbo	OBS
Definition	EVS offers scalable block storage for cloud servers. With high reliability, high performance, and rich specifications, EVS disks can be used for distributed file systems, dev/test environments, data warehouses, and high-performance computing (HPC) applications.	Expandable to 320 TB, SFS Turbo provides fully hosted shared file storage, which is highly available and stable, to support small files and applications requiring low latency and high IOPS. You can use SFS Turbo in high-traffic websites, log storage, compression/decompression, DevOps, enterprise OA, and containerized applications.	Object Storage Service (OBS) provides massive, secure, and cost-effective data storage for you to store data of any type and size. You can use it in enterprise backup/archiving, video on demand (VoD), video surveillance, and many other scenarios.
Data storage logic	Stores binary data and cannot directly store files. To store files, format the file system first.	Stores files and sorts and displays data in the hierarchy of files and folders.	Stores objects. Files directly stored automatically generate the system metadata, which can also be customized by users.
Access mode	Accessible only after being mounted to ECSs or BMSs and initialized.	Supports the Network File System (NFS) protocol (NFSv3 only). You can seamlessly integrate existing applications and tools with SFS Turbo.	Accessible through the Internet or Direct Connect (DC). Specify the bucket address and use transmission protocols such as HTTP or HTTPS.
Static storage volumes	Supported. For details, see <a href="#">Using an Existing EVS Disk Through a Static PV</a> .	Supported. For details, see <a href="#">Using an Existing SFS Turbo File System Through a Static PV</a> .	Supported. For details, see <a href="#">Using an Existing OBS Bucket Through a Static PV</a> .
Dynamic storage volumes	Supported. For details, see <a href="#">Using an EVS Disk Through a Dynamic PV</a> .	Not supported	Supported. For details, see <a href="#">Using an OBS Bucket Through a Dynamic PV</a> .

Item	EVS	SFS Turbo	OBS
Features	Non-shared storage. Each volume can be mounted to only one node.	Shared storage featuring high performance and bandwidth	Shared, user-mode file system
Application scenarios	HPC, enterprise core cluster applications, enterprise application systems, and dev/test  <b>NOTE</b> HPC apps here require high-speed and high-IOPS storage, such as industrial design and energy exploration.	High-traffic websites, log storage, DevOps, and enterprise OA	Big data analytics, static website hosting, online video on demand (VoD), gene sequencing, intelligent video surveillance, backup and archiving, and enterprise cloud boxes (web disks)
Capacity	TB	General-purpose: TB	EB
Latency	1-2 ms	General-purpose: 1-5 ms	10 ms
Max. IOPS	2200-256000, depending on flavors	General-purpose: up to 100,000	Tens of millions
Bandwidth	MB/s	General-purpose: up to GB/s	TB/s

## Local Storage Comparison

Item	Local PV	Local Ephemeral Volume	emptyDir	hostPath
Definition	Node's local disks form a storage pool (VolumeGroup) through LVM. LVM divides them into logical volumes (LVs) and mounts them to pods.	Kubernetes native emptyDir, where node's local disks form a storage pool (VolumeGroup) through LVM. LVs are created as the storage medium of emptyDir and mounted to pods. LVs deliver better performance than the default storage medium of emptyDir.	Kubernetes native emptyDir. Its lifecycle is the same as that of a pod. Memory can be specified as the storage medium. When the pod is deleted, the emptyDir volume is deleted and its data is lost.	Used to mount a file directory of the host where a pod is located to a specified mount point of the pod.
Features	Low-latency, high-I/O, and non-HA PV. Storage volumes are non-shared storage and bound to nodes through labels. Therefore, storage volumes can be mounted only to a single pod.	Local ephemeral volume. The storage space is from local LVs.	Local ephemeral volume. The storage space comes from the local kubelet root directory or memory.	Used to mount files or directories of the host file system. Host directories can be automatically created. Pods can be migrated (not bound to nodes).
Storage volume mounting	Static storage volumes are not supported. <a href="#">Using a Local PV Through a Dynamic PV</a> is supported.	For details, see <a href="#">Using a Local EV</a> .	For details, see <a href="#">Using a Temporary Path</a> .	For details, see <a href="#">hostPath</a> .

Item	Local PV	Local Ephemeral Volume	emptyDir	hostPath
Application scenarios	High I/O requirements and built-in HA solutions of applications, for example, deploying MySQL in HA mode.	<ul style="list-style-type: none"> <li>Scratch space, such as for a disk-based merge sort</li> <li>Checkpointing a long computation for recovery from crashes</li> <li>Saving the files obtained by the content manager container when web server container data is used</li> </ul>	<ul style="list-style-type: none"> <li>Scratch space, such as for a disk-based merge sort</li> <li>Checkpointing a long computation for recovery from crashes</li> <li>Saving the files obtained by the content manager container when web server container data is used</li> </ul>	<p>Requiring a node file, for example, if Docker is used, you can use hostPath to mount the <b>/var/lib/docker</b> path of the node.</p> <p><b>NOTICE</b> Avoid using hostPath volumes as much as possible, as they are prone to security risks. If hostPath volumes must be used, they can only be applied to files or directories and mounted in read-only mode.</p>

## Enterprise Project Support

### NOTE

To use this function, the Everest add-on must be upgraded to v1.2.33 or later.

- Automatically creating storage:

When creating EVS or OBS PVCs using a StorageClass in CCE, you can specify an enterprise project to assign the created storage resources (EVS disks and OBS) to. **This enterprise project can either be the default one or the same one as the cluster belongs to.**

If no enterprise project is specified, the enterprise project specified in StorageClass will be used by default for creating storage resources.

- For a custom StorageClass, you can specify an enterprise project in StorageClass. For details, see [Scenario 3: Specifying an Enterprise Project for a StorageClass](#). If no enterprise project is specified in StorageClass, the default enterprise project is used.
- For the csi-disk and csi-obs storage classes provided by CCE, the created storage resources belong to the default enterprise project.

- Use existing storage:

When you create a PVC using a PV, ensure that **everest.io/enterprise-project-id** specified in the PVC and PV are the same because an enterprise

project has been specified during storage resource creation. Otherwise, the PVC and PV cannot be bound.

## Documentation

- [Storage Basics](#)
- [Elastic Volume Service](#)
- [Scalable File Service](#)
- [SFS Turbo](#)
- [Object Storage Service](#)

## 8.2 Storage Basics

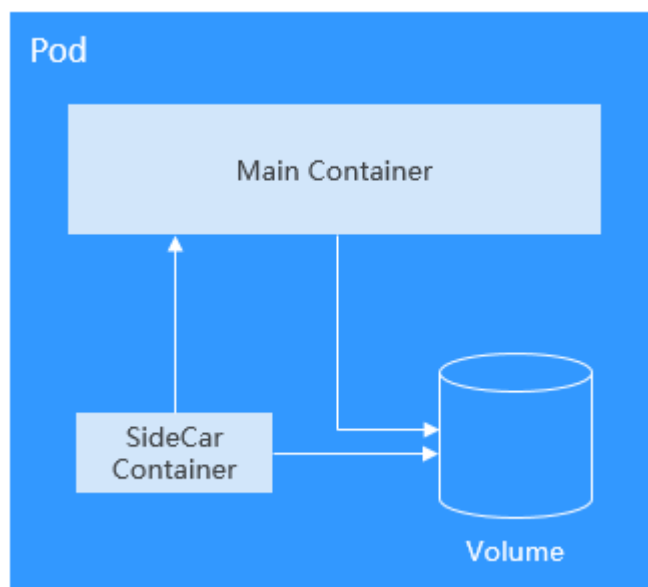
### Volumes

On-disk files in a container are ephemeral, which presents the following problems to important applications running in the container:

1. When a container is rebuilt, files in the container will be lost.
2. When multiple containers run in a pod at the same time, files need to be shared among the containers.

Kubernetes volumes resolve both of these problems. Volumes, as part of a pod, cannot be created independently and can only be defined in pods. All containers in a pod can access its volumes, but the volumes must have been mounted to any directory in a container.

The following figure shows how a storage volume is used between containers in a pod.



The basic principles for using volumes are as follows:

- Multiple volumes can be mounted to a pod. However, do not mount too many volumes to a pod.

- Multiple types of volumes can be mounted to a pod.
- Each volume mounted to a pod can be shared among containers in the pod.
- You are advised to use PVCs and PVs to mount volumes for Kubernetes.

 **NOTE**

The lifecycle of a volume is the same as that of the pod to which the volume is mounted. When the pod is deleted, the volume is also deleted. However, files in the volume may outlive the volume, depending on the volume type.

Kubernetes provides various volume types, which can be classified as in-tree and out-of-tree.

Volume Classification	Description
In-tree	<p>Maintained through the Kubernetes code repository and built, edited, and released with Kubernetes binary files. Kubernetes does not accept this volume type anymore.</p> <p>Kubernetes-native volumes such as hostPath, emptyDir, Secret, and ConfigMap are all the in-tree type.</p> <p>PVCs are a special in-tree volume. Kubernetes uses this type of volume to convert from in-tree to out-of-tree. PVCs allow you to request for PVs created using the underlying storage resources provided by different storage vendors.</p>
Out-of-tree	<p>Out-of-tree volumes include container storage interfaces (CSIs) and FlexVolumes (deprecated). Storage vendors only need to comply with certain specifications to create custom storage add-ons and PVs that can be used by Kubernetes, without adding add-on source code to the Kubernetes code repository. Cloud storage such as SFS and OBS is used by installing storage drivers in a cluster. You need to create PVs in the cluster and mount the PVs to pods using PVCs.</p>

## PV and PVC

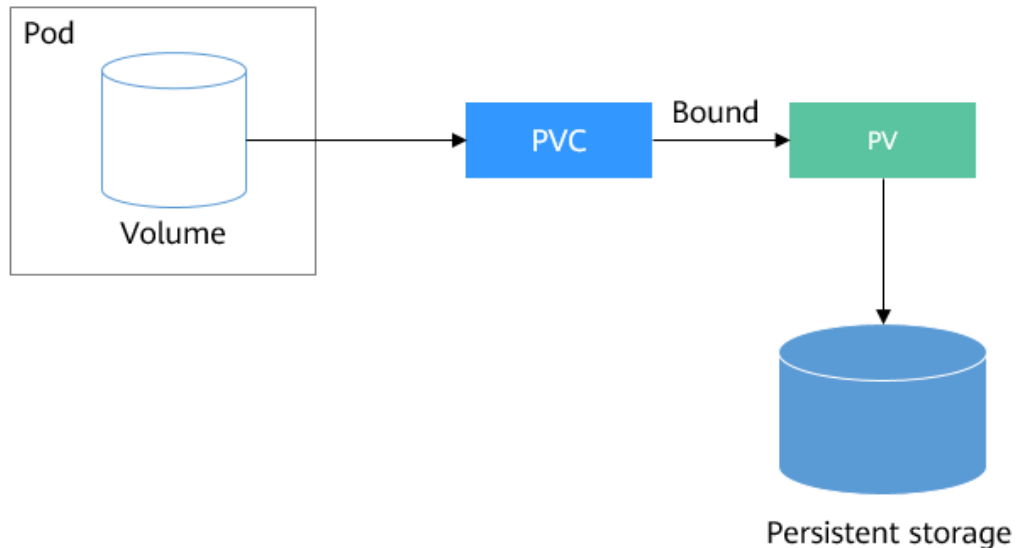
Kubernetes provides PersistentVolumes (PVs) and PersistentVolumeClaims (PVCs) to abstract details of how storage is provided from how it is consumed. You can request specific size of storage when needed, just like pods can request specific levels of resources (CPU and memory).

- PV: describes a persistent storage volume in a cluster. A PV is a cluster-level resource just like a node. It applies to the entire Kubernetes cluster. A PV has a lifecycle independent of any individual Pod that uses the PV.
- PVC: describes a request for storage by a user. When configuring storage for an application, claim a storage request (that is, PVC). Kubernetes selects a PV that best meets the request and binds the PV to the PVC. A PVC to PV binding is a one-to-one mapping. When creating a PVC, describe the attributes of the requested persistent storage, such as the storage size and read/write permission.



You can bind PVCs to PVs in a pod so that the pod can use storage resources. The following figure shows the relationship between PVs and PVCs.

**Figure 8-1** PVC-to-PV binding



## CSI

CSI is a standard for container storage interfaces and a storage plugin implementation solution recommended by the Kubernetes community. [Everest](#) is a storage add-on developed based on CSI. It provides different types of persistent storage for containers.

## Volume Access Modes

Storage volumes can be mounted to the host system only in the mode supported by underlying storage resources. For example, a file storage system can be read and written by multiple nodes, but an EVS disk can be read and written by only one node.

- **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.

**Table 8-1** Access modes supported by storage volumes

Volume Type	ReadWriteOnce	ReadWriteMany
EVS	√	x
SFS	x	√
OBS	x	√
SFS Turbo	x	√

## Mounting a Storage Volume

You can mount volumes in the following ways:

Use PVs to describe existing storage resources, and then create PVCs to use the storage resources in pods. You can also use the dynamic creation mode. That is, specify the **StorageClass** when creating a PVC and use the provisioner in the StorageClass to automatically create a PV and bind the PV to the PVC.

**Table 8-2** Modes of mounting volumes

Mount Mode	Description	Supported Volume Type	Other Constraints
Statically creating storage volume (using existing storage)	Use existing storage (such as EVS disks and SFS file systems) to create PVs and mount the PVs to the workload through PVCs. Kubernetes binds PVCs to the matching PVs so that workloads can access storage services.	All volumes	None
Dynamically creating storage volumes (automatically creating storage)	Specify a <b>StorageClass</b> for a PVC. The storage provisioner creates underlying storage media as required to automatically create PVs and directly bind the PV to the PVC.	EVS, OBS, SFS, and local PV	None
Dynamic mounting (VolumeClaimTemplate)	Achieved by using the <b>volumeClaimTemplates</b> field and depends on the dynamic PV creation capability of StorageClass. In this mode, each pod is associated with a unique PVC and PV. After a pod is rescheduled, the original data can still be mounted to it based on the PVC name.	EVS and local PV	Supported only by StatefulSets

## PV Reclaim Policy

A PV reclaim policy is used to delete or reclaim underlying volumes when a PVC is deleted. The value can be **Delete** or **Retain**.

- **Delete:** Deleting a PVC will remove the PV from Kubernetes, and the associated underlying storage assets will also be removed from the external infrastructure.

### NOTE

Yearly/Monthly resources cannot be deleted using the **Delete** reclaim policy.

- **Retain:** When a PVC is deleted, both the PV and underlying storage resources will be retained. You need to manually delete these resources. After the PVC is deleted, the PV is in the **Released** state and cannot be bound to a PVC again. You can manually delete and reclaim volumes by performing the following operations:
  - a. Delete the PV.
  - b. Clear data on the associated underlying storage resources as required.
  - c. Delete the associated underlying storage resources.

To reuse the underlying storage resources, create a PV.

CCE also allows you to delete a PVC without deleting underlying storage resources. This function can be achieved only by using a YAML file: Set the PV reclaim policy to **Delete** and add **everest.io/reclaim-policy: retain-volume-only** to **annotations**. In this way, when the PVC is deleted, the PV is deleted, but the underlying storage resources are retained.

The following YAML file takes EVS as an example:

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: test
  namespace: default
  annotations:
    volume.beta.kubernetes.io/storage-provisioner: everest-csi-provisioner
    everest.io/disk-volume-type: SAS
  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: 10Gi
  storageClassName: csi-disk
  volumeName: pv-evs-test
---
apiVersion: v1
kind: PersistentVolume
metadata:
  annotations:
    pv.kubernetes.io/provisioned-by: everest-csi-provisioner
    everest.io/reclaim-policy: retain-volume-only
  name: pv-evs-test
  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
  capacity:
    storage: 10Gi
  csi:
    driver: disk.csi.everest.io
    fsType: ext4
    volumeHandle: 2af98016-6082-4ad6-bedc-1a9c673aef20
  volumeAttributes:
    storage.kubernetes.io/csiProvisionerIdentity: everest-csi-provisioner
```

```

everest.io/disk-mode: SCSI
everest.io/disk-volume-type: SAS
persistentVolumeReclaimPolicy: Delete
storageClassName: csi-disk
    
```

## Documentation

- For more information about Kubernetes storage, see [Storage](#).
- For more information about CCE container storage, see [Overview](#).

## 8.3 Elastic Volume Service

### 8.3.1 Overview

To achieve persistent storage, CCE allows you to mount the storage volumes created from Elastic Volume Service (EVS) disks to a path of a container. When the container is migrated within an AZ, the mounted EVS volumes are also migrated. By using EVS volumes, you can mount the remote file directory of a storage system to a container so that data in the data volume is permanently preserved. Even if the container is deleted, the data in the data volume is still stored in the storage system.

### EVS Disk Performance Specifications

EVS performance metrics include:

- IOPS: the number of input/output operations performed by an EVS disk per second
- Throughput: the amount of data read from and written into an EVS disk per second
- I/O latency: the minimum interval between two consecutive I/O operations on an EVS disk

**Table 8-3** EVS disk performance specifications

Parameter	Ultra-high I/O	High I/O
Max. capacity (GiB)	<ul style="list-style-type: none"> <li>• System disk: 1,024</li> <li>• Data disk: 32,768</li> </ul>	<ul style="list-style-type: none"> <li>• System disk: 1,024</li> <li>• Data disk: 32,768</li> </ul>
Max. IOPS	50,000	5000
Max. throughput (MiB/s)	350	150
Burst IOPS limit	16,000	5000
Disk IOPS	Min. (50,000, 1800 + 50 x Capacity)	Min. (5000, 1800 + 8 x Capacity)
Disk throughput (MiB/s)	Min. (350, 120 + 0.5 x Capacity)	Min. (150, 100 + 0.15 x Capacity)

Parameter	Ultra-high I/O	High I/O
Single-queue access latency (ms)	1	1-3
API name	SSD	SAS

For details about EVS disks, see [Disk Types and Performance](#).

## Application Scenarios

EVS disks can be mounted in the following modes based on application scenarios:

- **Using an Existing EVS Disk Through a Static PV:** static creation mode, where you use an existing EVS disk to create a PV and then mount storage to the workload through a PVC. This mode applies if the underlying storage is available or billed on a yearly/monthly basis.
- **Using an EVS Disk Through a Dynamic PV:** dynamic creation mode, in which you do not need to create EVS volumes beforehand. Instead, specify a StorageClass when creating a PVC. Then, an EVS volume and PV will be created automatically. This mode applies to scenarios where no underlying storage is available.
- **Dynamically Mounting an EVS Disk to a StatefulSet:** available only for StatefulSets. In this mode, each pod is associated with a unique PVC and PV. After a pod is rescheduled, the original data can still be mounted to it based on the PVC name. This mode applies to StatefulSets with multiple pods.

## Billing

- To mount storage volumes of the EVS type, the billing mode of EVS disks **automatically created** by specifying the StorageClass is pay-per-use by default and cannot be changed to yearly/monthly. To use a yearly/monthly-billed EVS disk, [use an existing one](#).
- For details about the EVS disk pricing, see [Billing for Disks](#).

### 8.3.2 Using an Existing EVS Disk Through a Static PV

CCE allows you to create a PV using an existing EVS disk. After the PV is created, you can create a PVC and bind it to the PV. This mode applies if the underlying storage is available or billed on a yearly/monthly basis.

## Prerequisites

- You have created a cluster and installed the [CCE Container Storage \(Everest\)](#) add-on in the cluster.
- You have created an EVS disk that meets the following requirements:
  - The EVS disk cannot be a system disk, DSS disk, or shared disk.
  - The EVS disk must be of the **SCSI** type (the default disk type is **VBD** when you purchase an EVS disk).

- The EVS disk must be available and not used by other resources.
- The AZ of the EVS disk must be the same as that of the cluster node. Otherwise, the EVS disk cannot be mounted and the pod cannot start.
- The EVS disk must be in the default enterprise project or the enterprise project to which the cluster belongs.
- EVS disks that have been partitioned are not supported.
- Only ext4 EVS disks are supported.
- To create a cluster using commands, ensure kubectl is used. For details, see [Connecting to a Cluster Using kubectl](#).

## Notes and Constraints

- EVS disks cannot be attached across AZs and cannot be used by multiple workloads, multiple pods of the same workload, or multiple tasks. Data sharing of a shared disk is not supported between nodes in a CCE cluster. If an EVS disk is attached to multiple nodes, I/O conflicts and data cache conflicts may occur. Therefore, select only one pod when creating a Deployment that uses EVS disks.
- For clusters earlier than v1.19.10, if an HPA policy is used to scale out a workload with EVS volumes mounted, the existing pods cannot be read or written when a new pod is scheduled to another node.

For clusters of v1.19.10 and later, if an HPA policy is used to scale out a workload with EVS volumes mounted, a new pod cannot be started because EVS disks cannot be attached.

## Using an Existing EVS Disk on the Console

**Step 1** Log in to the CCE console and click the cluster name to access the cluster console.

**Step 2** Statically create a PVC and PV.

1. Choose **Storage** in the navigation pane. In the right pane, click the **PVCs** tab. Click **Create PVC** in the upper right corner. In the dialog box displayed, configure PVC parameters.

Parameter	Description
PVC Type	In this example, select <b>EVS</b> .
PVC Name	Enter the PVC name, which must be unique in a namespace.
Creation Method	<ul style="list-style-type: none"> <li>- If underlying storage is available, create a PV or use an existing PV to statically create a PVC.</li> <li>- If no underlying storage is available, select <b>Dynamically provision</b>. For details, see <a href="#">Using an EVS Disk Through a Dynamic PV</a>.</li> </ul> <p>In this example, select <b>Create new</b> to create both a PV and PVC on the console.</p>

Parameter	Description
PV <sup>a</sup>	Select an existing PV in the cluster. For details about how to create a PV, see "Creating a storage volume" in <a href="#">Related Operations</a> . You do not need to specify this parameter in this example.
EVS <sup>b</sup>	Click <b>Select EVS</b> . On the displayed page, select the EVS volume that meets your requirements and click <b>OK</b> .
PV Name <sup>b</sup>	Enter the PV name, which must be unique in the same cluster.
Access Mode <sup>b</sup>	EVS volumes support only <b>ReadWriteOnce</b> , indicating that a storage volume can be mounted to one node in read/write mode. For details, see <a href="#">Volume Access Modes</a> .
Reclaim Policy <sup>b</sup>	You can select <b>Delete</b> or <b>Retain</b> to specify the reclaim policy of the underlying storage when the PVC is deleted. For details, see <a href="#">PV Reclaim Policy</a> .

 **NOTE**

- a: The parameter is available when **Creation Method** is set to **Use existing**.
- b: The parameter is available when **Creation Method** is set to **Create new**.

2. Click **Create** to create a PVC and a PV.

You can choose **Storage** in the navigation pane and view the created PVC and PV on the **PVCs** and **PVs** tab pages, respectively.

**Step 3** Create an application.

1. Choose **Workloads** in the navigation pane. In the right pane, click the **StatefulSets** tab.
2. Click **Create Workload** in the upper right corner. On the displayed page, click **Data Storage** in the **Container Settings** area and click **Add Volume** to select **PVC**.

Mount and use storage volumes, as shown in [Table 8-4](#). For details about other parameters, see [Workloads](#).

**Table 8-4** Mounting a storage volume

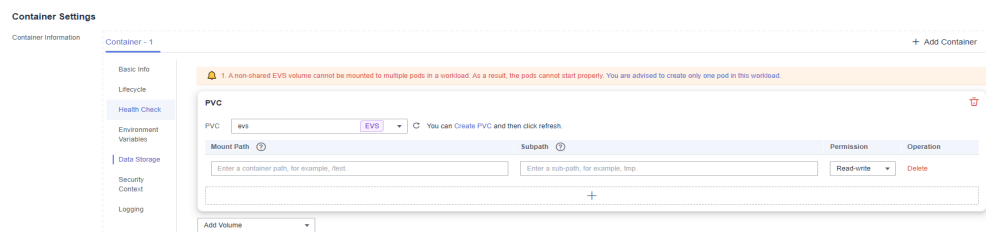
Parameter	Description
PVC	Select an existing EVS volume. An EVS volume can be mounted to only one workload.

Parameter	Description
Mount Path	<p>Enter a mount path, for example, <b>/tmp</b>.</p> <p>This parameter specifies a container path to which a data volume will be mounted. Do not mount the volume to a system directory such as <b>/</b> or <b>/var/run</b>. This may lead to container errors. Mount the volume to an empty directory. If the directory is not empty, ensure that there are no files that affect container startup. Otherwise, the files will be replaced, leading to container startup failures or workload creation failures.</p> <p><b>NOTICE</b> If a volume is mounted to a high-risk directory, use an account with minimum permissions to start the container. Otherwise, high-risk files on the host may be damaged.</p>
Subpath	<p>Enter the subpath of the storage volume and mount a path in the storage volume to the container. In this way, different folders of the same storage volume can be used in a single pod. <b>tmp</b>, for example, indicates that data in the mount path of the container is stored in the <b>tmp</b> folder of the storage volume. If this parameter is left blank, the root path will be used by default.</p>
Permission	<ul style="list-style-type: none"> <li>– <b>Read-only:</b> You can only read the data in the mounted volumes.</li> <li>– <b>Read-write:</b> You can modify the data volumes mounted to the path. Newly written data will not be migrated if the container is migrated, which may cause data loss.</li> </ul>

In this example, the disk is mounted to the **/data** path of the container. The container data generated in this path is stored in the EVS disk.

**NOTE**

A non-shared EVS disk can be attached to only one workload pod. If there are multiple pods, extra pods cannot start properly. Ensure that the number of workload pods is 1 if an EVS disk is attached.



3. After the configuration, click **Create Workload**.

After the workload is created, the data in the container mount directory will be persistently stored. Verify the storage by referring to [Verifying Data Persistence](#).

----End



## Using an Existing EVS Disk Through kubectl

**Step 1** Use kubectl to access the cluster.

**Step 2** Create a PV. If a PV has been created in your cluster, skip this step.

1. Create the **pv-evs.yaml** file.

```
apiVersion: v1
kind: PersistentVolume
metadata:
  annotations:
    pv.kubernetes.io/provisioned-by: everest-csi-provisioner
    everest.io/reclaim-policy: retain-volume-only # (Optional) The underlying volume is retained
    when the PV is deleted.
  name: pv-evs # PV name
  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 # Access mode, which must be ReadWriteOnce for EVS disks
  capacity:
    storage: 10Gi # EVS disk capacity, in the unit of GiB. The value ranges from 1 to 32768.
  csi:
    driver: disk.csi.everest.io # Dependent storage driver for the mounting
    fsType: ext4 # Must be the same as that of the original file system of the disk.
    volumeHandle: <your_volume_id> # EVS volume ID
    volumeAttributes:
      everest.io/disk-mode: SCSI # Device type of the EVS disk. Only SCSI is supported.
      everest.io/disk-volume-type: SAS # EVS disk type
      storage.kubernetes.io/csiProvisionerIdentity: everest-csi-provisioner

      everest.io/enterprise-project-id: <your_project_id> # (Optional) Enterprise project ID. If an
      enterprise project is specified, use the same enterprise project when creating a PVC. Otherwise, the
      PVC cannot be bound to a PV.
    persistentVolumeReclaimPolicy: Delete # Reclaim policy
    storageClassName: csi-disk # StorageClass name. The value must be csi-disk for EVS disks.
```

**Table 8-5** Key parameters

Parameter	Mandatory	Description
everest.io/reclaim-policy: retain-volume-only	No	Optional. Only <b>retain-volume-only</b> is supported. This parameter is valid only when the Everest version is 1.2.9 or later and the reclaim policy is <b>Delete</b> . If the reclaim policy is <b>Delete</b> and the current value is <b>retain-volume-only</b> , the associated PV is deleted while the underlying storage volume is retained, when a PVC is deleted.
failure-domain.beta.kubernetes.io/region	Yes	Region where the cluster is located.

Parameter	Mandatory	Description
failure-domain.beta.kubernetes.io/zone	Yes	AZ where the EVS volume is created. It must be the same as the AZ planned for the workload.
fsType	Yes	File system type, which defaults to <b>ext4</b> .
volumeHandle	Yes	Volume ID of the EVS disk. To obtain a volume ID, log in to the <b>Cloud Server Console</b> . In the navigation pane, choose <b>Elastic Volume Service &gt; Disks</b> . Click the name of the target EVS disk to go to its details page. On the <b>Summary</b> tab page, click the copy button after <b>ID</b> .
everest.io/disk-volume-type	Yes	EVS disk type. All letters are in uppercase. <ul style="list-style-type: none"> <li>- <b>SAS</b>: high I/O</li> <li>- <b>SSD</b>: ultra-high I/O</li> </ul>
everest.io/enterprise-project-id	No	Optional. Enterprise project ID of the EVS disk. If an enterprise project is specified, use the same enterprise project when creating a PVC. Otherwise, the PVC cannot be bound to a PV. To obtain an enterprise project ID, log in to the <b>Cloud Server Console</b> . In the navigation pane, choose <b>Elastic Volume Service &gt; Disks</b> . Click the name of the target EVS disk to go to its details page. On the <b>Summary</b> tab page, click the enterprise project in <b>Management Information</b> to access the enterprise project console. Copy the corresponding ID to obtain the ID of the enterprise project to which the EVS disk belongs.

Parameter	Mandatory	Description
persistentVolumeReclaimPolicy	Yes	<p>A reclaim policy is supported when the cluster version is or later than 1.19.10 and the Everest version is or later than 1.2.9.</p> <p>The <b>Delete</b> and <b>Retain</b> reclaim policies are supported. For details, see <a href="#">PV Reclaim Policy</a>. If high data security is required, select <b>Retain</b> to prevent data from being deleted by mistake.</p> <p><b>Delete:</b></p> <ul style="list-style-type: none"> <li>- If <b>everest.io/reclaim-policy</b> is not specified, both the PV and EVS disk will be deleted when a PVC is deleted.</li> <li>- If <b>everest.io/reclaim-policy</b> is set to <b>retain-volume-only</b>, when a PVC is deleted, the PV will be deleted but the EVS disk will be retained.</li> </ul> <p><b>Retain:</b> When a PVC is deleted, both the PV and underlying storage resources will be retained. You need to manually delete these resources. After the PVC is deleted, the PV is in the <b>Released</b> state and cannot be bound to a PVC again.</p>
storageClassName	Yes	The StorageClass for EVS disks is <b>csi-disk</b> .

2. Run the following command to create a PV:  

```
kubectl apply -f pv-evs.yaml
```

### Step 3 Create a PVC.

1. Create the **pvc-evs.yaml** file.

```

apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: pvc-evs
  namespace: default
  annotations:
    everest.io/disk-volume-type: SAS # EVS disk type

    everest.io/enterprise-project-id: <your_project_id> # (Optional) Enterprise project ID. If an
enterprise project is specified, use the same enterprise project when creating a PVC. Otherwise, the
PVC cannot be bound to a PV.
  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 # The value must be ReadWriteOnce for EVS disks.
  resources:
    requests:
      storage: 10Gi # EVS disk capacity, ranging from 1 to 32768. The value must be the same
as the storage size of the existing PV.

```

```
storageClassName: csi-disk # StorageClass is EVS.
volumeName: pv-evs # PV name
```

**Table 8-6** Key parameters

Parameter	Mandatory	Description
failure-domain.beta.kubernetes.io/region	Yes	Region where the cluster is located.
failure-domain.beta.kubernetes.io/zone	Yes	AZ where the EVS volume is created. It must be the same as the AZ planned for the workload.
storage	Yes	Requested capacity in the PVC, in Gi. The value must be the same as the storage size of the existing PV.
volumeName	Yes	PV name, which must be the same as the PV name in <a href="#">1</a> .
storageClassName	Yes	StorageClass name, which must be the same as the StorageClass of the PV in <a href="#">1</a> . The StorageClass for EVS disks is <b>csi-disk</b> .

2. Run the following command to create a PVC:  

```
kubectl apply -f pvc-evs.yaml
```

**Step 4** Create an application.

1. Create a file named **web-evs.yaml**. In this example, the EVS volume is mounted to the **/data** path.

```
apiVersion: apps/v1
kind: StatefulSet
metadata:
  name: web-evs
  namespace: default
spec:
  replicas: 1 # The number of workload replicas that use the EVS volume must be 1.
  selector:
    matchLabels:
      app: web-evs
  serviceName: web-evs # Headless Service name
  template:
    metadata:
      labels:
        app: web-evs
    spec:
      containers:
        - name: container-1
          image: nginx:latest
          volumeMounts:
            - name: pvc-disk # Volume name, which must be the same as the volume name in the
volumes field.
              mountPath: /data # Location where the storage volume is mounted
          imagePullSecrets:
            - name: default-secret
          volumes:
```

```

- name: pvc-disk # Volume name, which can be customized
  persistentVolumeClaim:
    claimName: pvc-evs # Name of the created PVC
---
apiVersion: v1
kind: Service
metadata:
  name: web-evs # Headless Service name
  namespace: default
  labels:
    app: web-evs
spec:
  selector:
    app: web-evs
  clusterIP: None
  ports:
    - name: web-evs
      targetPort: 80
      nodePort: 0
      port: 80
      protocol: TCP
  type: ClusterIP

```

2. Run the following command to create a workload to which the EVS volume is mounted:

```
kubectl apply -f web-evs.yaml
```

After the workload is created, the data in the container mount directory will be persistently stored. Verify the storage by referring to [Verifying Data Persistence](#).

----End

## Verifying Data Persistence

**Step 1** View the deployed application and EVS volume files.

1. Run the following command to view the created pod:

```
kubectl get pod | grep web-evs
```

Expected output:

```
web-evs-0          1/1    Running    0          38s
```

2. Run the following command to check whether the EVS volume has been mounted to the **/data** path:

```
kubectl exec web-evs-0 -- df | grep data
```

Expected output:

```
/dev/sdc          10255636   36888 10202364   0% /data
```

3. Run the following command to check the files in the **/data** path:

```
kubectl exec web-evs-0 -- ls /data
```

Expected output:

```
lost+found
```

**Step 2** Run the following command to create a file named **static** in the **/data** path:

```
kubectl exec web-evs-0 -- touch /data/static
```

**Step 3** Run the following command to check the files in the **/data** path:

```
kubectl exec web-evs-0 -- ls /data
```

Expected output:

```
lost+found
static
```

**Step 4** Run the following command to delete the pod named **web-eva-0**:

```
kubectl delete pod web-eva-0
```

Expected output:

```
pod "web-eva-0" deleted
```

**Step 5** After the deletion, the StatefulSet controller automatically creates a replica with the same name. Run the following command to check whether the files in the /**data** path have been modified:

```
kubectl exec web-eva-0 -- ls /data
```

Expected output:

```
lost+found  
static
```

The **static** file is retained, indicating that the data in the EVS volume can be stored persistently.

----End

## Related Operations

You can also perform the operations listed in [Table 8-7](#).

**Table 8-7** Related operations

Operation	Description	Procedure
Creating a storage volume (PV)	Create a PV on the CCE console.	<ol style="list-style-type: none"> <li>Choose <b>Storage</b> in the navigation pane. In the right pane, click the <b>PVs</b> tab. Click <b>Create PersistentVolume</b> in the upper right corner. In the dialog box displayed, configure parameters. <ul style="list-style-type: none"> <li><b>Volume Type:</b> Select <b>EVS</b>.</li> <li><b>EVS:</b> Click <b>Select EVS</b>. On the displayed page, select the EVS volume that meets your requirements and click <b>OK</b>.</li> <li><b>PV Name:</b> Enter the PV name, which must be unique in a cluster.</li> <li><b>Access Mode:</b> EVS volumes support only <b>ReadWriteOnce</b>, indicating that a storage volume can be mounted to one node in read/write mode. For details, see <a href="#">Volume Access Modes</a>.</li> <li><b>Reclaim Policy:</b> <b>Delete</b> or <b>Retain</b> is supported. For details, see <a href="#">PV Reclaim Policy</a>.</li> </ul> </li> <li>Click <b>Create</b>.</li> </ol>

Operation	Description	Procedure
Expanding the capacity of an EVS disk	<p>Quickly expand the capacity of an attached EVS disk on the CCE console.</p> <p>Only the capacity of pay-per-use EVS disks can be expanded on the CCE console. To expand the capacity of yearly/monthly EVS disks, click the volume name to go to the EVS console.</p>	<ol style="list-style-type: none"> <li>1. Choose <b>Storage</b> in the navigation pane. In the right pane, click the <b>PVCs</b> tab. Click <b>More</b> in the <b>Operation</b> column of the target PVC and select <b>Scale-out</b>.</li> <li>2. Enter the capacity to be added and click <b>OK</b>.</li> </ol>
Viewing events	View event names, event types, number of occurrences, Kubernetes events, first occurrence time, and last occurrence time of the PVC or PV.	<ol style="list-style-type: none"> <li>1. Choose <b>Storage</b> in the navigation pane. In the right pane, click the <b>PVCs</b> or <b>PVs</b> tab.</li> <li>2. Click <b>View Events</b> in the <b>Operation</b> column of the target PVC or PV to view events generated within one hour (events are retained for one hour).</li> </ol>
Viewing a YAML file	View, copy, or download the YAML file of a PVC or PV.	<ol style="list-style-type: none"> <li>1. Choose <b>Storage</b> in the navigation pane. In the right pane, click the <b>PVCs</b> or <b>PVs</b> tab.</li> <li>2. Click <b>View YAML</b> in the <b>Operation</b> column of the target PVC or PV to view or download the YAML.</li> </ol>

### 8.3.3 Using an EVS Disk Through a Dynamic PV

CCE allows you to specify a StorageClass to automatically create an EVS disk and the corresponding PV. This function is applicable when no underlying storage volume is available.

#### Prerequisites

- You have created a cluster and installed the [CCE Container Storage \(Everest\)](#) add-on in the cluster.
- To create a cluster using commands, ensure kubectl is used. For details, see [Connecting to a Cluster Using kubectl](#).

## Notes and Constraints

- EVS disks cannot be attached across AZs and cannot be used by multiple workloads, multiple pods of the same workload, or multiple tasks. Data sharing of a shared disk is not supported between nodes in a CCE cluster. If an EVS disk is attached to multiple nodes, I/O conflicts and data cache conflicts may occur. Therefore, select only one pod when creating a Deployment that uses EVS disks.
- For clusters earlier than v1.19.10, if an HPA policy is used to scale out a workload with EVS volumes mounted, the existing pods cannot be read or written when a new pod is scheduled to another node.

For clusters of v1.19.10 and later, if an HPA policy is used to scale out a workload with EVS volumes mounted, a new pod cannot be started because EVS disks cannot be attached.

## (Console) Automatically Creating an EVS Disk

**Step 1** Log in to the CCE console and click the cluster name to access the cluster console.

**Step 2** Dynamically create a PVC and PV.

1. Choose **Storage** in the navigation pane. In the right pane, click the **PVCs** tab. Click **Create PVC** in the upper right corner. In the dialog box displayed, configure PVC parameters.

Parameter	Description
PVC Type	In this example, select <b>EVS</b> .
PVC Name	Enter the PVC name, which must be unique in a namespace.
Creation Method	<ul style="list-style-type: none"> <li>– If no underlying storage is available, select <b>Dynamically provision</b> to create a PVC, PV, and underlying storage on the console in cascading mode.</li> <li>– If underlying storage is available, create a PV or use an existing PV to statically create a PVC. For details, see <a href="#">Using an Existing EVS Disk Through a Static PV</a>.</li> </ul> <p>In this example, select <b>Dynamically provision</b>.</p>
Storage Classes	<p>The default StorageClasses for EVS disks are <b>csi-disk</b> and <b>csi-disk-topology</b>.</p> <p><b>NOTE</b></p> <p>If you use the <b>csi-disk</b> (EVS) StorageClass, a PVC and PV will be created immediately. The EVS disk is created with the PV, and then the PVC is bound to the PV.</p> <p>If you use the <b>csi-disk-topology</b> (EVS created with a delay) StorageClass, a PV will not be immediately created when a PVC is created. Instead, the pods that will be associated with the PVC are scheduled first, and then the EVS disk and PV are created, and finally the PV is bound to the PVC.</p>



Parameter	Description
AZ	Select the AZ of the EVS disk. The AZ must be the same as that of the cluster node. <b>NOTE</b> An EVS disk can only be mounted to a node in the same AZ. After an EVS disk is created, its AZ cannot be changed.
Disk Type	Select an EVS disk type. EVS disk types vary depending on regions. Obtain the available EVS types on the console.
Capacity (GiB)	Capacity of the requested storage volume.
Access Mode	EVS volumes support only <b>ReadWriteOnce</b> , indicating that a storage volume can be mounted to one node in read/write mode. For details, see <a href="#">Volume Access Modes</a> .
Enterprise Project	The default enterprise project, the enterprise project to which the cluster belongs, or the enterprise project specified by StorageClass is available.

2. Click **Create**.

You can choose **Storage** in the navigation pane and view the created PVC and PV on the **PVCs** and **PVs** tab pages, respectively.

**Step 3** Create an application.

1. Choose **Workloads** in the navigation pane. In the right pane, click the **StatefulSets** tab.
2. Click **Create Workload** in the upper right corner. On the displayed page, click **Data Storage** in the **Container Settings** area and click **Add Volume** to select **PVC**.

Mount and use storage volumes, as shown in [Table 8-8](#). For details about other parameters, see [Workloads](#).

**Table 8-8** Mounting a storage volume

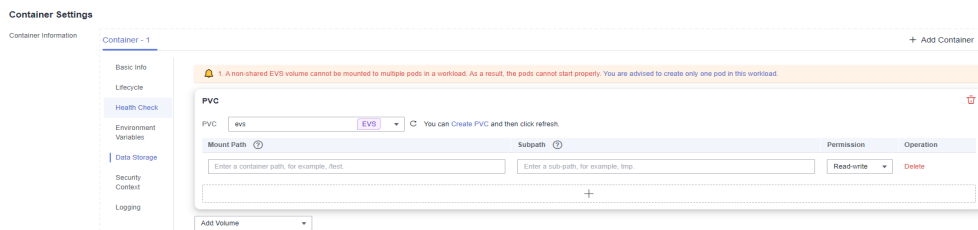
Parameter	Description
PVC	Select an existing EVS volume. An EVS volume can be mounted to only one workload.

Parameter	Description
Mount Path	<p>Enter a mount path, for example, <b>/tmp</b>.</p> <p>This parameter specifies a container path to which a data volume will be mounted. Do not mount the volume to a system directory such as <b>/</b> or <b>/var/run</b>. This may lead to container errors. Mount the volume to an empty directory. If the directory is not empty, ensure that there are no files that affect container startup. Otherwise, the files will be replaced, leading to container startup failures or workload creation failures.</p> <p><b>NOTICE</b> If a volume is mounted to a high-risk directory, use an account with minimum permissions to start the container. Otherwise, high-risk files on the host may be damaged.</p>
Subpath	<p>Enter the subpath of the storage volume and mount a path in the storage volume to the container. In this way, different folders of the same storage volume can be used in a single pod. <b>tmp</b>, for example, indicates that data in the mount path of the container is stored in the <b>tmp</b> folder of the storage volume. If this parameter is left blank, the root path will be used by default.</p>
Permission	<ul style="list-style-type: none"> <li>– <b>Read-only:</b> You can only read the data in the mounted volumes.</li> <li>– <b>Read-write:</b> You can modify the data volumes mounted to the path. Newly written data will not be migrated if the container is migrated, which may cause data loss.</li> </ul>

In this example, the disk is mounted to the **/data** path of the container. The container data generated in this path is stored in the EVS disk.

**NOTE**

A non-shared EVS disk can be attached to only one workload pod. If there are multiple pods, extra pods cannot start properly. Ensure that the number of workload pods is 1 if an EVS disk is attached.



3. After the configuration, click **Create Workload**.

After the workload is created, the data in the container mount directory will be persistently stored. Verify the storage by referring to [Verifying Data Persistence](#).

----End

## Automatically Creating an EVS Volume Through kubectl

**Step 1** Use kubectl to access the cluster.

**Step 2** Use **StorageClass** to dynamically create a PVC and PV.

1. Create the **pvc-evs-auto.yaml** file.

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: pvc-evs-auto
  namespace: default
  annotations:
    everest.io/disk-volume-type: SAS # EVS disk type

    everest.io/enterprise-project-id: <your_project_id> # (Optional) Enterprise project ID. If an
    enterprise project is specified, use the same enterprise project when creating a PVC. Otherwise, the
    PVC cannot be bound to a PV.

  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 # The value must be ReadWriteOnce for EVS disks.
  resources:
    requests:
      storage: 10Gi # EVS disk capacity, ranging from 1 to 32768.
  storageClassName: csi-disk # StorageClass is EVS.
```

**Table 8-9** Key parameters

Parameter	Mandatory	Description
failure-domain.beta.kubernetes.io/region	Yes	Region where the cluster is located.
failure-domain.beta.kubernetes.io/zone	Yes	AZ where the EVS volume is created. It must be the same as the AZ planned for the workload.
everest.io/disk-volume-type	Yes	EVS disk type. All letters are in uppercase. <ul style="list-style-type: none"> <li>- <b>SAS</b>: high I/O</li> <li>- <b>SSD</b>: ultra-high I/O</li> </ul>

Parameter	Mandatory	Description
everest.io/enterprise-project-id	No	Optional. Enterprise project ID of the EVS disk. If an enterprise project is specified, use the same enterprise project when creating a PVC. Otherwise, the PVC cannot be bound to a PV.  To obtain an enterprise project ID, log in to the EPS console, click the name of the target enterprise project, and copy the enterprise project ID.
storage	Yes	Requested PVC capacity, in Gi. The value ranges from <b>1</b> to <b>32768</b> .
storageClassName	Yes	The StorageClass for EVS disks is <b>csi-disk</b> .

2. Run the following command to create a PVC:  

```
kubectl apply -f pvc-evs-auto.yaml
```

### Step 3 Create an application.

1. Create a file named **web-evs-auto.yaml**. In this example, the EVS volume is mounted to the **/data** path.

```
apiVersion: apps/v1
kind: StatefulSet
metadata:
  name: web-evs-auto
  namespace: default
spec:
  replicas: 1
  selector:
    matchLabels:
      app: web-evs-auto
  serviceName: web-evs-auto # Headless Service name
  template:
    metadata:
      labels:
        app: web-evs-auto
    spec:
      containers:
        - name: container-1
          image: nginx:latest
          volumeMounts:
            - name: pvc-disk # Volume name, which must be the same as the volume name in the
volumes field.
              mountPath: /data # Location where the storage volume is mounted
      imagePullSecrets:
        - name: default-secret
      volumes:
        - name: pvc-disk # Volume name, which can be customized
          persistentVolumeClaim:
            claimName: pvc-evs-auto # Name of the created PVC
---
apiVersion: v1
kind: Service
metadata:
  name: web-evs-auto # Headless Service name
  namespace: default
  labels:
```

```
  app: web-eva-auto
spec:
  selector:
    app: web-eva-auto
  clusterIP: None
  ports:
  - name: web-eva-auto
    targetPort: 80
    nodePort: 0
    port: 80
    protocol: TCP
  type: ClusterIP
```

2. Run the following command to create a workload to which the EVS volume is mounted:

```
kubectl apply -f web-eva-auto.yaml
```

After the workload is created, the data in the container mount directory will be persistently stored. Verify the storage by referring to [Verifying Data Persistence](#).

----End

## Verifying Data Persistence

**Step 1** View the deployed application and EVS volume files.

1. Run the following command to view the created pod:

```
kubectl get pod | grep web-eva-auto
```

Expected output:

```
web-eva-auto-0          1/1    Running    0          38s
```

2. Run the following command to check whether the EVS volume has been mounted to the **/data** path:

```
kubectl exec web-eva-auto-0 -- df | grep data
```

Expected output:

```
/dev/sdc                10255636  36888 10202364  0% /data
```

3. Run the following command to check the files in the **/data** path:

```
kubectl exec web-eva-auto-0 -- ls /data
```

Expected output:

```
lost+found
```

**Step 2** Run the following command to create a file named **static** in the **/data** path:

```
kubectl exec web-eva-auto-0 -- touch /data/static
```

**Step 3** Run the following command to check the files in the **/data** path:

```
kubectl exec web-eva-auto-0 -- ls /data
```

Expected output:

```
lost+found
static
```

**Step 4** Run the following command to delete the pod named **web-eva-auto-0**:

```
kubectl delete pod web-eva-auto-0
```

Expected output:

```
pod "web-eva-auto-0" deleted
```

**Step 5** After the deletion, the StatefulSet controller automatically creates a replica with the same name. Run the following command to check whether the files in the **/data** path have been modified:

```
kubectl exec web-evs-auto-0 -- ls /data
```

Expected output:

```
lost+found
static
```

The **static** file is retained, indicating that the data in the EVS volume can be stored persistently.

----End

## Related Operations

You can also perform the operations listed in [Table 8-10](#).

**Table 8-10** Related operations

Operation	Description	Procedure
Expanding the capacity of an EVS disk	Quickly expand the capacity of an attached EVS disk on the CCE console.  Only the capacity of pay-per-use EVS disks can be expanded on the CCE console. To expand the capacity of yearly/monthly EVS disks, click the volume name to go to the EVS console.	<ol style="list-style-type: none"> <li>1. Choose <b>Storage</b> in the navigation pane. In the right pane, click the <b>PVCs</b> tab. Click <b>More</b> in the <b>Operation</b> column of the target PVC and select <b>Scale-out</b>.</li> <li>2. Enter the capacity to be added and click <b>OK</b>.</li> </ol>
Viewing events	View event names, event types, number of occurrences, Kubernetes events, first occurrence time, and last occurrence time of the PVC or PV.	<ol style="list-style-type: none"> <li>1. Choose <b>Storage</b> in the navigation pane. In the right pane, click the <b>PVCs</b> or <b>PVs</b> tab.</li> <li>2. Click <b>View Events</b> in the <b>Operation</b> column of the target PVC or PV to view events generated within one hour (events are retained for one hour).</li> </ol>
Viewing a YAML file	View, copy, or download the YAML file of a PVC or PV.	<ol style="list-style-type: none"> <li>1. Choose <b>Storage</b> in the navigation pane. In the right pane, click the <b>PVCs</b> or <b>PVs</b> tab.</li> <li>2. Click <b>View YAML</b> in the <b>Operation</b> column of the target PVC or PV to view or download the YAML.</li> </ol>

## 8.3.4 Dynamically Mounting an EVS Disk to a StatefulSet

### Application Scenarios

Dynamic mounting is available only for creating a [StatefulSet](#). It is implemented through a volume claim template ([volumeClaimTemplates](#) field) and depends on dynamic creation of PVs through StorageClass. In this mode, each pod in a multi-pod StatefulSet is associated with a unique PVC and PV. After a pod is rescheduled, the original data can still be mounted to it based on the PVC name. In the common mounting mode for a Deployment, if ReadWriteMany is supported, multiple pods of the Deployment will be mounted to the same underlying storage.

#### NOTE

When updating a StatefulSet in Kubernetes, it is not allowed to add or delete the [volumeClaimTemplates](#) field. This field can only be configured during the creation of the StatefulSet.

### Prerequisites

- You have created a cluster and installed the [CCE Container Storage \(Everest\)](#) add-on in the cluster.
- To create a cluster using commands, ensure kubectl is used. For details, see [Connecting to a Cluster Using kubectl](#).

### Dynamically Mounting an EVS Disk on the Console

**Step 1** Log in to the CCE console and click the cluster name to access the cluster console.

**Step 2** Choose **Workloads** in the navigation pane. In the right pane, click the **StatefulSets** tab.

**Step 3** Click **Create Workload** in the upper right corner. On the displayed page, click **Data Storage** in the **Container Settings** area and click **Add Volume** to select **VolumeClaimTemplate**.

**Step 4** Click **Create PVC**. In the dialog box displayed, configure PVC parameters.

Click **Create**.

Parameter	Description
PVC Type	In this example, select <b>EVS</b> .
PVC Name	Enter the name of the PVC. After a PVC is created, a suffix is automatically added based on the number of pods. The format is <i>&lt;Custom PVC name&gt;-&lt;Serial number&gt;</i> , for example, example-0.
Creation Method	You can select <b>Dynamically provision</b> to create a PVC, PV, and underlying storage on the console in cascading mode.
Storage Classes	The default StorageClass for EVS disks is <b>csi-disk</b> .

Parameter	Description
AZ	Select the AZ of the EVS disk. The AZ must be the same as that of the cluster node. <b>NOTE</b> An EVS disk can only be mounted to a node in the same AZ. After an EVS disk is created, its AZ cannot be changed.
Disk Type	Select an EVS disk type. EVS disk types vary depending on regions. Obtain the available EVS types on the console.
Capacity (GiB)	Capacity of the requested storage volume.
Access Mode	EVS volumes support only <b>ReadWriteOnce</b> , indicating that a storage volume can be mounted to one node in read/write mode. For details, see <a href="#">Volume Access Modes</a> .
Enterprise Project	The default enterprise project, the enterprise project to which the cluster belongs, or the enterprise project specified by StorageClass is available.

**Step 5** Enter the path to which the volume is mounted.

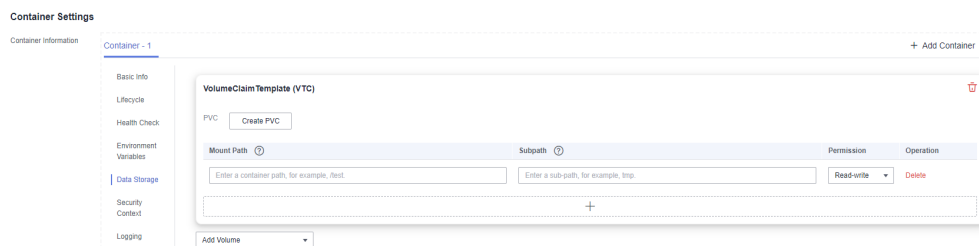
**Table 8-11** Mounting a storage volume

Parameter	Description
Mount Path	Enter a mount path, for example, <b>/tmp</b> . This parameter specifies a container path to which a data volume will be mounted. Do not mount the volume to a system directory such as <b>/</b> or <b>/var/run</b> . This may lead to container errors. Mount the volume to an empty directory. If the directory is not empty, ensure that there are no files that affect container startup. Otherwise, the files will be replaced, leading to container startup failures or workload creation failures. <b>NOTICE</b> If a volume is mounted to a high-risk directory, use an account with minimum permissions to start the container. Otherwise, high-risk files on the host may be damaged.
Subpath	Enter the subpath of the storage volume and mount a path in the storage volume to the container. In this way, different folders of the same storage volume can be used in a single pod. <b>tmp</b> , for example, indicates that data in the mount path of the container is stored in the <b>tmp</b> folder of the storage volume. If this parameter is left blank, the root path will be used by default.



Parameter	Description
Permission	<ul style="list-style-type: none"> <li>● <b>Read-only:</b> You can only read the data in the mounted volumes.</li> <li>● <b>Read-write:</b> You can modify the data volumes mounted to the path. Newly written data will not be migrated if the container is migrated, which may cause data loss.</li> </ul>

In this example, the disk is mounted to the **/data** path of the container. The container data generated in this path is stored in the EVS disk.



**Step 6** Dynamically mount and use storage volumes. For details about other parameters, see [Creating a StatefulSet](#). After the configuration, click **Create Workload**.

After the workload is created, the data in the container mount directory will be persistently stored. Verify the storage by referring to [Verifying Data Persistence](#).

----End

## Dynamically Mounting an EVS Volume Through kubectl

**Step 1** Use kubectl to access the cluster.

**Step 2** Create a file named **statefulset-evs.yaml**. In this example, the EVS volume is mounted to the **/data** path.

```

apiVersion: apps/v1
kind: StatefulSet
metadata:
  name: statefulset-evs
  namespace: default
spec:
  selector:
    matchLabels:
      app: statefulset-evs
  template:
    metadata:
      labels:
        app: statefulset-evs
    spec:
      containers:
        - name: container-1
          image: nginx:latest
          volumeMounts:
            - name: pvc-disk          # The value must be the same as that in the volumeClaimTemplates field.
              mountPath: /data      # Location where the storage volume is mounted
          imagePullSecrets:
            - name: default-secret
      serviceName: statefulset-evs  # Headless Service name
      replicas: 2
      volumeClaimTemplates:

```

```

- apiVersion: v1
  kind: PersistentVolumeClaim
  metadata:
    name: pvc-disk
    namespace: default
    annotations:
      everest.io/disk-volume-type: SAS # EVS disk type

  everest.io/enterprise-project-id: <your_project_id> # (Optional) Enterprise project ID. If an enterprise
  project is specified, use the same enterprise project when creating a PVC. Otherwise, the PVC cannot be
  bound to a PV.

  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 # The value must be ReadWriteOnce for EVS disks.
    resources:
      requests:
        storage: 10Gi # EVS disk capacity, ranging from 1 to 32768
        storageClassName: csi-disk # StorageClass is EVS
---
apiVersion: v1
kind: Service
metadata:
  name: statefulset-eva # Headless Service name
  namespace: default
  labels:
    app: statefulset-eva
spec:
  selector:
    app: statefulset-eva
  clusterIP: None
  ports:
    - name: statefulset-eva
      targetPort: 80
      nodePort: 0
      port: 80
      protocol: TCP
  type: ClusterIP

```

**Table 8-12** Key parameters

Parameter	Mandatory	Description
failure-domain.beta.kubernetes.io/region	Yes	Region where the cluster is located.
failure-domain.beta.kubernetes.io/zone	Yes	AZ where the EVS volume is created. It must be the same as the AZ planned for the workload.
everest.io/disk-volume-type	Yes	EVS disk type. All letters are in uppercase. <ul style="list-style-type: none"> <li>• SAS: high I/O</li> <li>• SSD: ultra-high I/O</li> </ul>

Parameter	Mandatory	Description
everest.io/enterprise-project-id	No	Optional. Enterprise project ID of the EVS disk. If an enterprise project is specified, use the same enterprise project when creating a PVC. Otherwise, the PVC cannot be bound to a PV. To obtain an enterprise project ID, log in to the <b>Cloud Server Console</b> . In the navigation pane, choose <b>Elastic Volume Service &gt; Disks</b> . Click the name of the target EVS disk to go to its details page. On the <b>Summary</b> tab page, click the enterprise project in <b>Management Information</b> to access the enterprise project console. Copy the corresponding ID to obtain the ID of the enterprise project to which the EVS disk belongs.
storage	Yes	Requested PVC capacity, in Gi. The value ranges from <b>1</b> to <b>32768</b> .
storageClassName	Yes	The StorageClass for EVS disks is <b>csi-disk</b> .

**Step 3** Run the following command to create a workload to which the EVS volume is mounted:

```
kubectl apply -f statefulset-evs.yaml
```

After the workload is created, the data in the container mount directory will be persistently stored. Verify the storage by referring to [Verifying Data Persistence](#).

----End

## Verifying Data Persistence

**Step 1** View the deployed application and EVS volume files.

1. Run the following command to view the created pod:

```
kubectl get pod | grep statefulset-evs
```

Expected output:

```
statefulset-evs-0 1/1 Running 0 45s
statefulset-evs-1 1/1 Running 0 28s
```

2. Run the following command to check whether the EVS volume has been mounted to the **/data** path:

```
kubectl exec statefulset-evs-0 -- df | grep data
```

Expected output:

```
/dev/sdd 10255636 36888 10202364 0% /data
```

3. Run the following command to check the files in the **/data** path:

```
kubectl exec statefulset-evs-0 -- ls /data
```

Expected output:

```
lost+found
```

**Step 2** Run the following command to create a file named **static** in the **/data** path:

```
kubectl exec statefulset-evs-0 -- touch /data/static
```

**Step 3** Run the following command to check the files in the **/data** path:

```
kubectl exec statefulset-evs-0 -- ls /data
```

Expected output:

```
lost+found
static
```

**Step 4** Run the following command to delete the pod named **web-evs-auto-0**:

```
kubectl delete pod statefulset-evs-0
```

Expected output:

```
pod "statefulset-evs-0" deleted
```

**Step 5** After the deletion, the StatefulSet controller automatically creates a replica with the same name. Run the following command to check whether the files in the **/data** path have been modified:

```
kubectl exec statefulset-evs-0 -- ls /data
```

Expected output:

```
lost+found
static
```

The **static** file is retained, indicating that the data in the EVS volume can be stored persistently.

----End

## Related Operations

You can also perform the operations listed in [Table 8-13](#).

**Table 8-13** Related operations

Operation	Description	Procedure
Expanding the capacity of an EVS disk	Quickly expand the capacity of an attached EVS disk on the CCE console.  Only the capacity of pay-per-use EVS disks can be expanded on the CCE console. To expand the capacity of yearly/monthly EVS disks, click the volume name to go to the EVS console.	<ol style="list-style-type: none"> <li>1. Choose <b>Storage</b> in the navigation pane. In the right pane, click the <b>PVCs</b> tab. Click <b>More</b> in the <b>Operation</b> column of the target PVC and select <b>Scale-out</b>.</li> <li>2. Enter the capacity to be added and click <b>OK</b>.</li> </ol>

Operation	Description	Procedure
Viewing events	View event names, event types, number of occurrences, Kubernetes events, first occurrence time, and last occurrence time of the PVC or PV.	<ol style="list-style-type: none"> <li>1. Choose <b>Storage</b> in the navigation pane. In the right pane, click the <b>PVCs</b> or <b>PVs</b> tab.</li> <li>2. Click <b>View Events</b> in the <b>Operation</b> column of the target PVC or PV to view events generated within one hour (events are retained for one hour).</li> </ol>
Viewing a YAML file	View, copy, or download the YAML file of a PVC or PV.	<ol style="list-style-type: none"> <li>1. Choose <b>Storage</b> in the navigation pane. In the right pane, click the <b>PVCs</b> or <b>PVs</b> tab.</li> <li>2. Click <b>View YAML</b> in the <b>Operation</b> column of the target PVC or PV to view or download the YAML.</li> </ol>

### 8.3.5 Snapshots and Backups

CCE works with EVS to support snapshots. A snapshot is a complete copy or image of EVS disk data at a certain point of time, which can be used for data DR.

You can create snapshots to rapidly save the disk data at a certain point of time. In addition, you can use snapshots to create disks so that the created disks will contain the snapshot data in the beginning.

#### Precautions

- The snapshot function is available **only for clusters of v1.15 or later** and requires the CSI-based Everest add-on.
- The subtype (common I/O, high I/O, or ultra-high I/O), disk mode (VBD or SCSI), sharing status, and capacity of an EVS disk created from a snapshot must be the same as those of the disk associated with the snapshot. These attributes cannot be modified after being checked or configured.
- Snapshots can be created only for EVS disks that are available or in use, and a maximum of seven snapshots can be created for a single EVS disk.
- Snapshots can be created only for PVCs created using the storage class (whose name starts with csi) provided by the Everest add-on. Snapshots cannot be created for PVCs created using the FlexVolume storage class whose name is `ssd`, `sas`, or `sata`.

#### Application Scenarios

The snapshot feature helps address your following needs:

- **Routine data backup**  
You can create snapshots for EVS disks regularly and use snapshots to recover your data in case that data loss or data inconsistency occurred due to misoperations, viruses, or attacks.

- **Rapid data restoration**

You can create a snapshot or multiple snapshots before an OS change, application software upgrade, or a service data migration. If an exception occurs during the upgrade or migration, service data can be rapidly restored to the time point when the snapshot was created.

For example, a fault occurred on system disk A of ECS A, and therefore ECS A cannot be started. Because system disk A is already faulty, the data on system disk A cannot be restored by rolling back snapshots. In this case, you can use an existing snapshot of system disk A to create EVS disk B and attach it to ECS B that is running properly. Then, ECS B can read data from system disk A using EVS disk B.

 **NOTE**

The snapshot capability provided by CCE is the same as the CSI snapshot function provided by the Kubernetes community. EVS disks can be created only based on snapshots, and snapshots cannot be rolled back to source EVS disks.

- **Rapid deployment of multiple services**

You can use a snapshot to create multiple EVS disks containing the same initial data, and these disks can be used as data resources for various services, for example, data mining, report query, and development and testing. This method protects the initial data and creates disks rapidly, meeting the diversified service data requirements.

## Creating a Snapshot

### Using the CCE console

**Step 1** Log in to the CCE console.

**Step 2** Click the cluster name to go to the cluster console. Choose **Storage** in the navigation pane. In the right pane, click the **Snapshots and Backups** tab.

**Step 3** Click **Create Snapshot** in the upper right corner. In the dialog box displayed, set related parameters.

- **Snapshot Name:** Enter a snapshot name.
- **Storage:** Select an EVS PVC.

**Step 4** Click **Create**.

----End

### Using YAML

```
kind: VolumeSnapshot
apiVersion: snapshot.storage.k8s.io/v1beta1
metadata:
  name: cce-disksnap-test # Snapshot name
  namespace: default
spec:
  source:
    persistentVolumeClaimName: pvc-evs-test # PVC name. Only an EVS PVC can be selected.
    volumeSnapshotClassName: csi-disk-snapclass
```

## Using a Snapshot to Create a PVC

The disk type and disk mode of the created EVS PVC are consistent with those of the snapshot's source EVS disk.

### Using the CCE console

- Step 1** Log in to the CCE console.
- Step 2** Click the cluster name to go to the cluster console. Choose **Storage** in the navigation pane. In the right pane, click the **Snapshots and Backups** tab.
- Step 3** Locate the snapshot that you want to use for creating a PVC, click **Create PVC**, and configure PVC parameters in the displayed dialog box.
  - **PVC Name:** Enter a PVC name.
- Step 4** Click **Create**.

----End

### Using YAML

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: pvc-test
  namespace: default
  annotations:
    everest.io/disk-volume-type: SSD # EVS disk type, which must be the same as that of the snapshot's
    source EVS disk.

  labels:
    failure-domain.beta.kubernetes.io/region: <your_region> # Replace the region with the one where
    the EVS disk is located.
    failure-domain.beta.kubernetes.io/zone: <your_zone> # Replace the AZ with the one where the
    EVS disk is located.
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 10Gi
  storageClassName: csi-disk
  dataSource:
    name: cce-disksnap-test # Snapshot name
    kind: VolumeSnapshot
    apiGroup: snapshot.storage.k8s.io
```

## 8.4 Scalable File Service

### 8.4.1 Overview

#### Introduction

CCE allows you to mount a volume created from a Scalable File Service (SFS) file system to a container to store data persistently. SFS volumes are commonly used in ReadWriteMany scenarios for large-capacity expansion and cost-sensitive services, such as media processing, content management, big data analysis, and

workload process analysis. For services with massive volume of small files, SFS Turbo file systems are recommended.

Expandable to petabytes, SFS provides fully hosted shared file storage, highly available and stable to handle data- and bandwidth-intensive applications

- **Standard file protocols:** You can mount file systems as volumes to servers, the same as using local directories.
- **Data sharing:** The same file system can be mounted to multiple servers, so that data can be shared.
- **Private network:** Users can access data only in private networks of data centers.
- **Capacity and performance:** The capacity of a single file system is high (PB level) and the performance is excellent (ms-level I/O latency).
- **Use cases:** Deployments/StatefulSets in the ReadWriteMany mode and jobs created for high-performance computing (HPC), media processing, content management, web services, big data analysis, and workload process analysis

## Performance

CCE supports SFS Capacity-Oriented and SFS 3.0. For details about file system types, see [File System Type](#).

### NOTE

- SFS Capacity-Oriented file systems are sold out and cannot be used to create PVs on the CCE console. However, you can still create PVs using existing SFS Capacity-Oriented file systems through [kubectl](#).

**Table 8-14** Performance

Parameter	SFS Capacity-Oriented
Maximum bandwidth	2 GB/s
Maximum IOPS	2000
Latency	3–20 ms
Maximum capacity	4 PB

## Application Scenarios

SFS supports the following mounting modes based on application scenarios:

- **Using an Existing SFS File System Through a Static PV:** static creation mode, where you use an existing SFS volume to create a PV and then mount storage to the workload through a PVC. This mode applies if the underlying storage is available or billed on a yearly/monthly basis.
- **Using an SFS File System Through a Dynamic PV:** dynamic creation mode, in which you do not need to create SFS file systems beforehand. Instead, specify a StorageClass when creating a PVC. Then, a file system and PV will be



created automatically. This mode applies to scenarios where no underlying storage is available.

## Billing

- The default billing mode of an SFS volume **automatically created** using StorageClass is **Pay-per-use**, indicating that it is billed based on the used storage capacity and duration. For details about SFS pricing, see **Billing**.
- To use a yearly/monthly-billed SFS volume, **use an existing one**.

## 8.4.2 Using an Existing SFS File System Through a Static PV

SFS is a network-attached storage (NAS) that provides shared, scalable, and high-performance file storage. It applies to large-capacity expansion and cost-sensitive services. This section describes how to use an existing SFS file system to statically create PVs and PVCs for data persistence and sharing in workloads.

### Prerequisites

- You have created a cluster and installed the **CCE Container Storage (Everest)** add-on in the cluster.
- To create a cluster using commands, ensure kubectl is used. For details, see **Connecting to a Cluster Using kubectl**.
- You have created an SFS file system that is in the same VPC as the cluster.

### Notes and Constraints

- Multiple PVs can use the same SFS or SFS Turbo file system with the following restrictions:
  - Do not mount the PVCs/PVs that use the same underlying SFS or SFS Turbo volume to one pod. This will lead to a pod startup failure because not all PVCs can be mounted to the pod due to the same **volumeHandle** value.
  - The **persistentVolumeReclaimPolicy** parameter in the PVs must be set to **Retain**. Otherwise, when a PV is deleted, the associated underlying volume may be deleted. In this case, other PVs associated with the underlying volume malfunction.
  - When the underlying volume is repeatedly used, enable isolation and protection for ReadWriteMany at the application layer to prevent data overwriting and loss.

## Using an Existing SFS File System on the Console

**Step 1** Log in to the CCE console and click the cluster name to access the cluster console.

**Step 2** Statically create a PVC and PV.

1. Choose **Storage** in the navigation pane. In the right pane, click the **PVCs** tab. Click **Create PVC** in the upper right corner. In the dialog box displayed, configure PVC parameters.

Parameter	Description
PVC Type	In this example, select <b>SFS</b> .
PVC Name	Enter the PVC name, which must be unique in a namespace.
Creation Method	<ul style="list-style-type: none"><li>– If underlying storage is available, create a PV or use an existing PV to statically create a PVC.</li><li>– If no underlying storage is available, select <b>Dynamically provision</b>. For details, see <a href="#">Using an SFS File System Through a Dynamic PV</a>.</li></ul> In this example, select <b>Create new</b> to create both a PV and PVC on the console.
PV <sup>a</sup>	Select an existing PV in the cluster. For details about how to create a PV, see "Creating a storage volume" in <a href="#">Related Operations</a> . You do not need to specify this parameter in this example.
SFS <sup>b</sup>	Click <b>Select SFS</b> . On the displayed page, select the SFS file system that meets your requirements and click <b>OK</b> .
PV Name <sup>b</sup>	Enter the PV name, which must be unique in the same cluster.
Access Mode <sup>b</sup>	SFS volumes support only <b>ReadWriteMany</b> , indicating that a storage volume can be mounted to multiple nodes in read/write mode. For details, see <a href="#">Volume Access Modes</a> .
Reclaim Policy <sup>b</sup>	You can select <b>Delete</b> or <b>Retain</b> to specify the reclaim policy of the underlying storage when the PVC is deleted. For details, see <a href="#">PV Reclaim Policy</a> . <b>NOTE</b> If multiple PVs use the same underlying storage volume, use <b>Retain</b> to prevent the underlying volume from being deleted with a PV.
Mount Options <sup>b</sup>	Enter the mounting parameter key-value pairs. For details, see <a href="#">Configuring SFS Volume Mount Options</a> .

 **NOTE**

a: The parameter is available when **Creation Method** is set to **Use existing**.

b: The parameter is available when **Creation Method** is set to **Create new**.

2. Click **Create** to create a PVC and a PV.

You can choose **Storage** in the navigation pane and view the created PVC and PV on the **PVCs** and **PVs** tab pages, respectively.

**Step 3** Create an application.

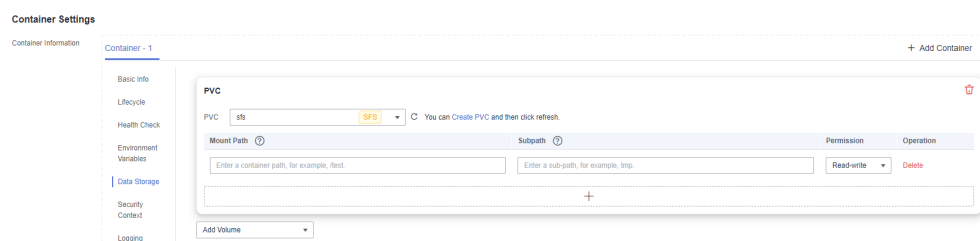
1. Choose **Workloads** in the navigation pane. In the right pane, click the **Deployments** tab.
2. Click **Create Workload** in the upper right corner. On the displayed page, click **Data Storage** in the **Container Settings** area and click **Add Volume** to select **PVC**.

Mount and use storage volumes, as shown in [Table 8-15](#). For details about other parameters, see [Workloads](#).

**Table 8-15** Mounting a storage volume

Parameter	Description
PVC	Select an existing SFS volume.
Mount Path	<p>Enter a mount path, for example, <b>/tmp</b>.</p> <p>This parameter specifies a container path to which a data volume will be mounted. Do not mount the volume to a system directory such as <b>/</b> or <b>/var/run</b>. This may lead to container errors. Mount the volume to an empty directory. If the directory is not empty, ensure that there are no files that affect container startup. Otherwise, the files will be replaced, leading to container startup failures or workload creation failures.</p> <p><b>NOTICE</b></p> <p>If a volume is mounted to a high-risk directory, use an account with minimum permissions to start the container. Otherwise, high-risk files on the host may be damaged.</p>
Subpath	<p>Enter the subpath of the storage volume and mount a path in the storage volume to the container. In this way, different folders of the same storage volume can be used in a single pod. <b>tmp</b>, for example, indicates that data in the mount path of the container is stored in the <b>tmp</b> folder of the storage volume. If this parameter is left blank, the root path will be used by default.</p>
Permission	<ul style="list-style-type: none"> <li>– <b>Read-only:</b> You can only read the data in the mounted volumes.</li> <li>– <b>Read-write:</b> You can modify the data volumes mounted to the path. Newly written data will not be migrated if the container is migrated, which may cause data loss.</li> </ul>

In this example, the disk is mounted to the **/data** path of the container. The container data generated in this path is stored in the SFS file system.



3. After the configuration, click **Create Workload**.

After the workload is created, the data in the container mount directory will be persistently stored. Verify the storage by referring to [Verifying Data Persistence and Sharing](#).

----End

## Using an Existing SFS Capacity-Oriented File System

**Step 1** Use kubectl to access the cluster.

**Step 2** Create a PV.

1. Create the **pv-sfs.yaml** file.

Example:

```
apiVersion: v1
kind: PersistentVolume
metadata:
  annotations:
    pv.kubernetes.io/provisioned-by: everest-csi-provisioner
    everest.io/reclaim-policy: retain-volume-only # (Optional) The underlying volume is retained
when the PV is deleted.
  name: pv-sfs # PV name
spec:
  accessModes:
    - ReadWriteMany # Access mode. The value must be ReadWriteMany for SFS.
  capacity:
    storage: 1Gi # SFS volume capacity
  csi:
    driver: nas.csi.everest.io # Dependent storage driver for the mounting
    fsType: nfs
    volumeHandle: <your_volume_id> # SFS Capacity-Oriented volume ID
    volumeAttributes:
      everest.io/share-export-location: <your_location> # Shared path of the SFS volume
    storage.kubernetes.io/csiProvisionerIdentity: everest-csi-provisioner
    persistentVolumeReclaimPolicy: Retain # Reclaim policy
    storageClassName: csi-nas # StorageClass name. csi-nas indicates that SFS Capacity-
Oriented is used.
  mountOptions: [] # Mount options
```

**Table 8-16** Key parameters

Parameter	Mandatory	Description
everest.io/reclaim-policy	No	Only <b>retain-volume-only</b> is supported. This parameter is valid only when the Everest version is 1.2.9 or later and the reclaim policy is <b>Delete</b> . If the reclaim policy is <b>Delete</b> and the current value is <b>retain-volume-only</b> , the associated PV is deleted while the underlying storage volume is retained, when a PVC is deleted.

Parameter	Mandatory	Description
volumeHandle	Yes	Volume ID if SFS Capacity-Oriented is used. Log in to the CCE console, choose <b>Service List &gt; Storage &gt; Scalable File Service</b> , and select <b>SFS Capacity-Oriented</b> . In the list, click the name of the target SFS file system. On the details page, copy the content following <b>ID</b> .
everest.io/share-export-location	Yes	Shared path of the file system. On the management console, choose <b>Service List &gt; Storage &gt; Scalable File Service</b> . You can obtain the shared path of the file system from the <b>Mount Address</b> column.
mountOptions	Yes	Mount options. If not specified, the following configurations are used by default. For details, see <a href="#">Configuring SFS Volume Mount Options</a> . mountOptions: - vers=3 - timeo=600 - nolock - hard
persistentVolumeReclaimPolicy	Yes	A reclaim policy is supported when the cluster version is or later than 1.19.10 and the Everest version is or later than 1.2.9. The <b>Delete</b> and <b>Retain</b> reclaim policies are supported. For details, see <a href="#">PV Reclaim Policy</a> . If multiple PVs use the same SFS volume, use <b>Retain</b> to prevent the underlying volume from being deleted with a PV. <b>Retain:</b> When a PVC is deleted, both the PV and underlying storage resources will be retained. You need to manually delete these resources. After the PVC is deleted, the PV is in the <b>Released</b> state and cannot be bound to a PVC again. <b>Delete:</b> When a PVC is deleted, its PV will also be deleted.
storage	Yes	Requested capacity in the PVC, in Gi. For SFS, this field is used only for verification (cannot be empty or <b>0</b> ). Its value is fixed at <b>1</b> , and any value you set does not take effect for SFS file systems.

Parameter	Mandatory	Description
storageClassName	Yes	StorageClass name <b>csi-nas</b> , indicating that SFS 1.0 Capacity-Oriented is used for storage.

- Run the following command to create a PV:  

```
kubectl apply -f pv-sfs.yaml
```

### Step 3 Create a PVC.

- Create the **pvc-sfs.yaml** file.

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: pvc-sfs
  namespace: default
  annotations:
    volume.beta.kubernetes.io/storage-provisioner: everest-csi-provisioner
spec:
  accessModes:
    - ReadWriteMany # The value must be ReadWriteMany for SFS.
  resources:
    requests:
      storage: 1Gi # SFS volume capacity
      storageClassName: csi-nas # StorageClass name, which must be the same as that of the PV
      volumeName: pv-sfs # PV name
```

**Table 8-17** Key parameters

Parameter	Mandatory	Description
storage	Yes	Requested capacity in the PVC, in Gi. The value must be the same as the storage size of the existing PV.
storageClassName	Yes	StorageClass name <b>csi-nas</b> , which must be the same as the StorageClass of the PV specified in <b>1</b> . This indicates that SFS 1.0 Capacity-Oriented is used for storage.
volumeName	Yes	PV name, which must be the same as the PV name in <b>1</b> .

- Run the following command to create a PVC:  

```
kubectl apply -f pvc-sfs.yaml
```

### Step 4 Create an application.

- Create a file named **web-demo.yaml**. In this example, the SFS volume is mounted to the **/data** path.

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: web-demo
  namespace: default
spec:
```

```
replicas: 2
selector:
  matchLabels:
    app: web-demo
template:
  metadata:
    labels:
      app: web-demo
  spec:
    containers:
      - name: container-1
        image: nginx:latest
        volumeMounts:
          - name: pvc-sfs-volume # Volume name, which must be the same as the volume name in the
volumes field
            mountPath: /data # Location where the storage volume is mounted
        imagePullSecrets:
          - name: default-secret
        volumes:
          - name: pvc-sfs-volume # Volume name, which can be customized
            persistentVolumeClaim:
              claimName: pvc-sfs # Name of the created PVC
```

2. Run the following command to create a workload to which the SFS volume is mounted:

```
kubectl apply -f web-demo.yaml
```

After the workload is created, the data in the container mount directory will be persistently stored. Verify the storage by referring to [Verifying Data Persistence and Sharing](#).

----End

## Verifying Data Persistence and Sharing

**Step 1** View the deployed application and files.

1. Run the following command to view the created pod:

```
kubectl get pod | grep web-demo
```

Expected output:

```
web-demo-846b489584-mjhm9 1/1 Running 0 46s
web-demo-846b489584-wvv5s 1/1 Running 0 46s
```

2. Run the following commands in sequence to view the files in the **/data** path of the pods:

```
kubectl exec web-demo-846b489584-mjhm9 -- ls /data
kubectl exec web-demo-846b489584-wvv5s -- ls /data
```

If no result is returned for both pods, no file exists in the **/data** path.

**Step 2** Run the following command to create a file named **static** in the **/data** path:

```
kubectl exec web-demo-846b489584-mjhm9 -- touch /data/static
```

**Step 3** Run the following command to check the files in the **/data** path:

```
kubectl exec web-demo-846b489584-mjhm9 -- ls /data
```

Expected output:

```
static
```

**Step 4** Verify data persistence.

1. Run the following command to delete the pod named **web-demo-846b489584-mjhm9**:

```
kubectl delete pod web-demo-846b489584-mjhm9
```

Expected output:

```
pod "web-demo-846b489584-mjhm9" deleted
```

After the deletion, the Deployment controller automatically creates a replica.

2. Run the following command to view the created pod:

```
kubectl get pod | grep web-demo
```

The expected output is as follows, in which **web-demo-846b489584-d4d4j** is the newly created pod:

```
web-demo-846b489584-d4d4j 1/1 Running 0 110s
web-demo-846b489584-wvv5s 1/1 Running 0 7m50s
```

3. Run the following command to check whether the files in the **/data** path of the new pod have been modified:

```
kubectl exec web-demo-846b489584-d4d4j -- ls /data
```

Expected output:

```
static
```

The **static** file is retained, indicating that the data in the file system can be stored persistently.

### Step 5 Verify data sharing.

1. Run the following command to view the created pod:

```
kubectl get pod | grep web-demo
```

Expected output:

```
web-demo-846b489584-d4d4j 1/1 Running 0 7m
web-demo-846b489584-wvv5s 1/1 Running 0 13m
```

2. Run the following command to create a file named **share** in the **/data** path of either pod: In this example, select the pod named **web-demo-846b489584-d4d4j**.

```
kubectl exec web-demo-846b489584-d4d4j -- touch /data/share
```

Check the files in the **/data** path of the pod.

```
kubectl exec web-demo-846b489584-d4d4j -- ls /data
```

Expected output:

```
share
static
```

3. Check whether the **share** file exists in the **/data** path of another pod (**web-demo-846b489584-wvv5s**) as well to verify data sharing.

```
kubectl exec web-demo-846b489584-wvv5s -- ls /data
```

Expected output:

```
share
static
```

After you create a file in the **/data** path of a pod, if the file is also created in the **/data** path of the other pod, the two pods share the same volume.

----End

## Related Operations

You can also perform the operations listed in [Table 8-18](#).



**Table 8-18** Related operations

Operation	Description	Procedure
Creating a storage volume (PV)	Create a PV on the CCE console.	<ol style="list-style-type: none"> <li>Choose <b>Storage</b> in the navigation pane. In the right pane, click the <b>PVs</b> tab. Click <b>Create PersistentVolume</b> in the upper right corner. In the dialog box displayed, configure parameters. <ul style="list-style-type: none"> <li><b>Volume Type:</b> Select <b>SFS</b>.</li> <li><b>SFS:</b> Click <b>Select SFS</b>. On the displayed page, select the SFS file system that meets your requirements and click <b>OK</b>.</li> <li><b>PV Name:</b> Enter the PV name, which must be unique in a cluster.</li> <li><b>Access Mode:</b> SFS volumes support only <b>ReadWriteMany</b>, indicating that a storage volume can be mounted to multiple nodes in read/write mode. For details, see <a href="#">Volume Access Modes</a>.</li> <li><b>Reclaim Policy:</b> <b>Delete</b> or <b>Retain</b> is supported. For details, see <a href="#">PV Reclaim Policy</a>.</li> </ul> <p><b>NOTE</b> If multiple PVs use the same underlying storage volume, use <b>Retain</b> to prevent the underlying volume from being deleted with a PV.</p> <li><b>Mount Options:</b> Enter the mounting parameter key-value pairs. For details, see <a href="#">Configuring SFS Volume Mount Options</a>.</li> </li></ol> <ol style="list-style-type: none"> <li>Click <b>Create</b>.</li> </ol>
Viewing events	View event names, event types, number of occurrences, Kubernetes events, first occurrence time, and last occurrence time of the PVC or PV.	<ol style="list-style-type: none"> <li>Choose <b>Storage</b> in the navigation pane. In the right pane, click the <b>PVCs</b> or <b>PVs</b> tab.</li> <li>Click <b>View Events</b> in the <b>Operation</b> column of the target PVC or PV to view events generated within one hour (events are retained for one hour).</li> </ol>
Viewing a YAML file	View, copy, or download the YAML file of a PVC or PV.	<ol style="list-style-type: none"> <li>Choose <b>Storage</b> in the navigation pane. In the right pane, click the <b>PVCs</b> or <b>PVs</b> tab.</li> <li>Click <b>View YAML</b> in the <b>Operation</b> column of the target PVC or PV to view or download the YAML.</li> </ol>

### 8.4.3 Using an SFS File System Through a Dynamic PV

This section describes how to use storage classes to dynamically create PVs and PVCs for data persistence and sharing in workloads.

#### Prerequisites

- You have created a cluster and installed the [CCE Container Storage \(Everest\)](#) add-on in the cluster.
- To create a cluster using commands, ensure kubectl is used. For details, see [Connecting to a Cluster Using kubectl](#).
- You have created an SFS file system that is in the same VPC as the cluster.

#### Automatically Creating an SFS File System on the Console

**Step 1** Log in to the CCE console and click the cluster name to access the cluster console.

**Step 2** Dynamically create a PVC and PV.

1. Choose **Storage** in the navigation pane. In the right pane, click the **PVCs** tab. Click **Create PVC** in the upper right corner. In the dialog box displayed, configure PVC parameters.

Parameter	Description
PVC Type	In this example, select <b>SFS</b> .
PVC Name	Enter the PVC name, which must be unique in a namespace.
Creation Method	<ul style="list-style-type: none"> <li>– If no underlying storage is available, select <b>Dynamically provision</b> to create a PVC, PV, and underlying storage on the console in cascading mode.</li> <li>– If underlying storage is available, create a PV or use an existing PV to statically create a PVC. For details, see <a href="#">Using an Existing SFS File System Through a Static PV</a>.</li> </ul> <p>In this document, <b>Dynamically provision</b> is selected.</p>
Storage Classes	The default StorageClass for SFS volumes is <b>csi-sfs</b> or <b>csi-nas</b> .
Access Mode	SFS volumes support only <b>ReadWriteMany</b> , indicating that a storage volume can be mounted to multiple nodes in read/write mode. For details, see <a href="#">Volume Access Modes</a> .

2. Click **Create** to create a PVC and a PV.

You can choose **Storage** in the navigation pane and view the created PVC and PV on the **PVCs** and **PVs** tab pages, respectively.

**Step 3** Create an application.

1. Choose **Workloads** in the navigation pane. In the right pane, click the **Deployments** tab.

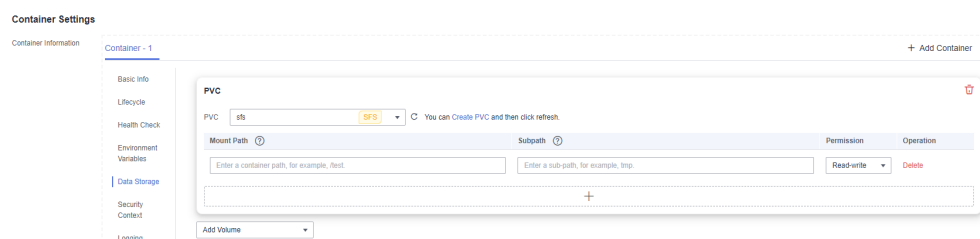
2. Click **Create Workload** in the upper right corner. On the displayed page, click **Data Storage** in the **Container Settings** area and click **Add Volume** to select **PVC**.

Mount and use storage volumes, as shown in [Table 8-19](#). For details about other parameters, see [Workloads](#).

**Table 8-19** Mounting a storage volume

Parameter	Description
PVC	Select an existing SFS volume.
Mount Path	<p>Enter a mount path, for example, <b>/tmp</b>.</p> <p>This parameter specifies a container path to which a data volume will be mounted. Do not mount the volume to a system directory such as <b>/</b> or <b>/var/run</b>. This may lead to container errors. Mount the volume to an empty directory. If the directory is not empty, ensure that there are no files that affect container startup. Otherwise, the files will be replaced, leading to container startup failures or workload creation failures.</p> <p><b>NOTICE</b> If a volume is mounted to a high-risk directory, use an account with minimum permissions to start the container. Otherwise, high-risk files on the host may be damaged.</p>
Subpath	Enter the subpath of the storage volume and mount a path in the storage volume to the container. In this way, different folders of the same storage volume can be used in a single pod. <b>tmp</b> , for example, indicates that data in the mount path of the container is stored in the <b>tmp</b> folder of the storage volume. If this parameter is left blank, the root path will be used by default.
Permission	<ul style="list-style-type: none"> <li>– <b>Read-only:</b> You can only read the data in the mounted volumes.</li> <li>– <b>Read-write:</b> You can modify the data volumes mounted to the path. Newly written data will not be migrated if the container is migrated, which may cause data loss.</li> </ul>

In this example, the disk is mounted to the **/data** path of the container. The container data generated in this path is stored in the SFS file system.



3. After the configuration, click **Create Workload**.  
After the workload is created, the data in the container mount directory will be persistently stored. Verify the storage by referring to [Verifying Data Persistence and Sharing](#).

----End

## Automatically Creating an SFS File System Through kubectl

**Step 1** Use kubectl to access the cluster.

**Step 2** Use **StorageClass** to dynamically create a PVC and PV.

1. Create the **pvc-sfs-auto.yaml** file.

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: pvc-sfs-auto
  namespace: default

spec:
  accessModes:
    - ReadWriteMany          # The value must be ReadWriteMany for SFS.
  resources:
    requests:
      storage: 1Gi          # SFS volume capacity
      storageClassName: csi-nas # StorageClass is SFS.
```

**Table 8-20** Key parameters

Parameter	Mandatory	Description
storage	Yes	Requested capacity in the PVC, in Gi. For SFS, this field is used only for verification (cannot be empty or <b>0</b> ). Its value is fixed at <b>1</b> , and any value you set does not take effect for SFS file systems.

2. Run the following command to create a PVC:  
kubectl apply -f pvc-sfs-auto.yaml

**Step 3** Create an application.

1. Create a file named **web-demo.yaml**. In this example, the SFS volume is mounted to the **/data** path.

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: web-demo
  namespace: default
spec:
  replicas: 2
  selector:
    matchLabels:
      app: web-demo
  template:
```

```
metadata:
  labels:
    app: web-demo
spec:
  containers:
  - name: container-1
    image: nginx:latest
    volumeMounts:
    - name: pvc-sfs-volume # Volume name, which must be the same as the volume name in the
volumes field
      mountPath: /data # Location where the storage volume is mounted
    imagePullSecrets:
    - name: default-secret
  volumes:
  - name: pvc-sfs-volume # Volume name, which can be customized
    persistentVolumeClaim:
      claimName: pvc-sfs-auto # Name of the created PVC
```

2. Run the following command to create a workload to which the SFS volume is mounted:

```
kubectl apply -f web-demo.yaml
```

After the workload is created, the data in the container mount directory will be persistently stored. Verify the storage by referring to [Verifying Data Persistence and Sharing](#).

----End

## Verifying Data Persistence and Sharing

**Step 1** View the deployed application and files.

1. Run the following command to view the created pod:

```
kubectl get pod | grep web-demo
```

Expected output:

```
web-demo-846b489584-mjhm9 1/1 Running 0 46s
web-demo-846b489584-wvv5s 1/1 Running 0 46s
```

2. Run the following commands in sequence to view the files in the **/data** path of the pods:

```
kubectl exec web-demo-846b489584-mjhm9 -- ls /data
kubectl exec web-demo-846b489584-wvv5s -- ls /data
```

If no result is returned for both pods, no file exists in the **/data** path.

**Step 2** Run the following command to create a file named **static** in the **/data** path:

```
kubectl exec web-demo-846b489584-mjhm9 -- touch /data/static
```

**Step 3** Run the following command to check the files in the **/data** path:

```
kubectl exec web-demo-846b489584-mjhm9 -- ls /data
```

Expected output:

```
static
```

**Step 4** Verify data persistence.

1. Run the following command to delete the pod named **web-demo-846b489584-mjhm9**:

```
kubectl delete pod web-demo-846b489584-mjhm9
```

Expected output:

```
pod "web-demo-846b489584-mjhm9" deleted
```

After the deletion, the Deployment controller automatically creates a replica.

2. Run the following command to view the created pod:

```
kubectl get pod | grep web-demo
```

The expected output is as follows, in which **web-demo-846b489584-d4d4j** is the newly created pod:

```
web-demo-846b489584-d4d4j 1/1 Running 0 110s
web-demo-846b489584-wvv5s 1/1 Running 0 7m50s
```

3. Run the following command to check whether the files in the **/data** path of the new pod have been modified:

```
kubectl exec web-demo-846b489584-d4d4j -- ls /data
```

Expected output:

```
static
```

The **static** file is retained, indicating that the data in the file system can be stored persistently.

### Step 5 Verify data sharing.

1. Run the following command to view the created pod:

```
kubectl get pod | grep web-demo
```

Expected output:

```
web-demo-846b489584-d4d4j 1/1 Running 0 7m
web-demo-846b489584-wvv5s 1/1 Running 0 13m
```

2. Run the following command to create a file named **share** in the **/data** path of either pod: In this example, select the pod named **web-demo-846b489584-d4d4j**.

```
kubectl exec web-demo-846b489584-d4d4j -- touch /data/share
```

Check the files in the **/data** path of the pod.

```
kubectl exec web-demo-846b489584-d4d4j -- ls /data
```

Expected output:

```
share
static
```

3. Check whether the **share** file exists in the **/data** path of another pod (**web-demo-846b489584-wvv5s**) as well to verify data sharing.

```
kubectl exec web-demo-846b489584-wvv5s -- ls /data
```

Expected output:

```
share
static
```

After you create a file in the **/data** path of a pod, if the file is also created in the **/data** path of the other pod, the two pods share the same volume.

----End

## Related Operations

You can also perform the operations listed in [Table 8-21](#).

**Table 8-21** Related operations

Operation	Description	Procedure
Viewing events	View event names, event types, number of occurrences, Kubernetes events, first occurrence time, and last occurrence time of the PVC or PV.	<ol style="list-style-type: none"> <li>1. Choose <b>Storage</b> in the navigation pane. In the right pane, click the <b>PVCs</b> or <b>PVs</b> tab.</li> <li>2. Click <b>View Events</b> in the <b>Operation</b> column of the target PVC or PV to view events generated within one hour (events are retained for one hour).</li> </ol>
Viewing a YAML file	View, copy, or download the YAML file of a PVC or PV.	<ol style="list-style-type: none"> <li>1. Choose <b>Storage</b> in the navigation pane. In the right pane, click the <b>PVCs</b> or <b>PVs</b> tab.</li> <li>2. Click <b>View YAML</b> in the <b>Operation</b> column of the target PVC or PV to view or download the YAML.</li> </ol>

## 8.4.4 Configuring SFS Volume Mount Options

This section describes how to configure SFS mount options. You can configure mount options in a PV and bind the PV to a PVC. Alternatively, configure mount options in a StorageClass and use the StorageClass to create a PVC. In this way, PVs can be dynamically created and inherit mount options configured in the StorageClass by default.

### Prerequisites

The **CCE Container Storage (Everest)** version must be **1.2.8 or later**. This add-on identifies the mount options and transfers them to the underlying storage resources. The parameter settings take effect only if the underlying storage resources support the specified options.

### Notes and Constraints

- Due to the restrictions of the NFS protocol, if an SFS volume is mounted to a node for multiple times, link-related mounting parameters (such as **timeo**) take effect only when the SFS volume is mounted for the first time by default. For example, if the same SFS file system is mounted to multiple pods running on a node, the mounting parameter set later does not overwrite the existing parameter value. If you want to configure different mounting parameters in the preceding scenario, additionally configure the **nosharecache** parameter.

### SFS Volume Mount Options

The Everest add-on in CCE presets the options described in [Table 8-22](#) for mounting SFS volumes.

**Table 8-22** SFS volume mount options

Parameter	Value	Description
keep-original-ownership	Blank	Whether to retain the ownership of the file mount point. If this option is used, the Everest add-on must be v1.2.63 or v2.1.2 or later. <ul style="list-style-type: none"> <li>By default, this option is not added, and the mount point ownership is <b>root:root</b> when SFS is mounted.</li> <li>If this option is added, the original ownership of the file system is retained when SFS is mounted.</li> </ul>
vers	3	File system version. Currently, only NFSv3 is supported. Value: <b>3</b>
nolock	Blank	Whether to lock files on the server using the NLM protocol. If <b>nolock</b> is selected, the lock is valid for applications on one host. For applications on another host, the lock is invalid.
timeo	600	Waiting time before the NFS client retransmits a request. The unit is 0.1 seconds. Recommended value: <b>600</b>
hard/soft	Blank	Mount mode. <ul style="list-style-type: none"> <li><b>hard</b>: If the NFS request times out, the client keeps resending the request until the request is successful.</li> <li><b>soft</b>: If the NFS request times out, the client returns an error to the invoking program.</li> </ul> The default value is <b>hard</b> .
sharecache/nosharecache	Blank	How the data cache and attribute cache are shared when one file system is concurrently mounted to different clients. If this parameter is set to <b>sharecache</b> , the caches are shared between the mountings. If this parameter is set to <b>nosharecache</b> , the caches are not shared, and one cache is configured for each client mounting. The default value is <b>sharecache</b> . <p><b>NOTE</b></p> The <b>nosharecache</b> setting will affect the performance. The mounting information must be obtained for each mounting, which increases the communication overhead with the NFS server and the memory consumption of the NFS clients. In addition, the <b>nosharecache</b> setting on the NFS clients may lead to inconsistent caches. Determine whether to use <b>nosharecache</b> based on site requirements.



## Configuring Mount Options in a PV

You can use the **mountOptions** field to configure mount options in a PV. The options you can configure in **mountOptions** are listed in [SFS Volume Mount Options](#).

**Step 1** Use `kubectl` to access the cluster. For details, see [Connecting to a Cluster Using kubectl](#).

**Step 2** Configure mount options in a PV. Example:

```
apiVersion: v1
kind: PersistentVolume
metadata:
  annotations:
    pv.kubernetes.io/provisioned-by: everest-csi-provisioner
    everest.io/reclaim-policy: retain-volume-only # (Optional) The underlying volume is retained when
the PV is deleted.
  name: pv-sfs
spec:
  accessModes:
    - ReadWriteMany # Access mode. The value must be ReadWriteMany for SFS.
  capacity:
    storage: 1Gi # SFS volume capacity
  csi:
    driver: nas.csi.everest.io # Dependent storage driver for the mounting
    fsType: nfs
    volumeHandle: <your_volume_id> # ID of the SFS Capacity-Oriented volume
    volumeAttributes:
      everest.io/share-export-location: <your_location> # Shared path of the SFS volume
      storage.kubernetes.io/csiProvisionerIdentity: everest-csi-provisioner
    persistentVolumeReclaimPolicy: Retain # Reclaim policy
    storageClassName: csi-nas # StorageClass name.
  mountOptions: # Mount options
    - vers=3
    - nolock
    - timeo=600
    - hard
```

**Step 3** After a PV is created, you can create a PVC and bind it to the PV, and then mount the PV to the container in the workload. For details, see [Using an Existing SFS File System Through a Static PV](#).

**Step 4** Check whether the mount options take effect.

In this example, the PVC is mounted to the workload that uses the **nginx:latest** image. You can run the **mount -l** command to check whether the mount options take effect.

1. View the pod to which the SFS volume has been mounted. In this example, the workload name is **web-sfs**.

```
kubectl get pod | grep web-sfs
```

Command output:

```
web-sfs-*** 1/1 Running 0 23m
```

2. Run the following command to check the mount options (**web-sfs-\*\*\*** is an example pod):

```
kubectl exec -it web-sfs-*** -- mount -l | grep nfs
```

If the mounting information in the command output is consistent with the configured mount options, the mount options have been configured.

```
<Your shared path> on /data type nfs
```

```
(rw,relatime,vers=3,rsize=1048576,wsz=1048576,namlen=255,hard,nolock,noresvport,proto=tcp,
```

```
timeo=600,retrans=2,sec=sys,mountaddr=**.**.**,mountvers=3,mountport=2050,mountproto=tcp,local_lock=all,addr=**.**.**)

```

----End

## Configuring Mount Options in a StorageClass

You can use the **mountOptions** field to configure mount options in a StorageClass. The options you can configure in **mountOptions** are listed in [SFS Volume Mount Options](#).

**Step 1** Use `kubectl` to access the cluster. For details, see [Connecting to a Cluster Using kubectl](#).

**Step 2** Create a custom StorageClass. Example:

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: csi-sfs-mount-option
provisioner: 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-to: <your_vpc_id> # VPC ID of the cluster
reclaimPolicy: Delete
volumeBindingMode: Immediate
mountOptions: # Mount options
- vers=3
- nolock
- timeo=600
- hard

```

**Step 3** After the StorageClass is configured, you can use it to create a PVC. By default, the dynamically created PVs inherit the mount options configured in the StorageClass. For details, see [Using an SFS File System Through a Dynamic PV](#).

**Step 4** Check whether the mount options take effect.

In this example, the PVC is mounted to the workload that uses the **nginx:latest** image. You can run the **mount -l** command to check whether the mount options take effect.

1. View the pod to which the SFS volume has been mounted. In this example, the workload name is **web-sfs**.

```
kubectl get pod | grep web-sfs
```

Command output:

```
web-sfs-*** 1/1 Running 0 23m
```

2. Run the following command to check the mount options (**web-sfs-\*\*\*** is an example pod):

```
kubectl exec -it web-sfs-*** -- mount -l | grep nfs
```

If the mounting information in the command output is consistent with the configured mount options, the mount options have been configured.

```
<Your shared path> on /data type nfs
```

```
(rw,relatime,vers=3,rsize=1048576,wsiz=1048576,namlen=255,hard,nolock,noresvport,proto=tcp,timeo=600,retrans=2,sec=sys,mountaddr=**.**.**,mountvers=3,mountport=2050,mountproto=tcp,local_lock=all,addr=**.**.**)

```

----End

## 8.5 SFS Turbo

### 8.5.1 Overview

#### Introduction

CCE allows you to mount storage volumes created by SFS Turbo file systems to a path of a container to meet data persistence requirements. SFS Turbo file systems are fast, on-demand, and scalable, which are suitable for scenarios with a massive number of small files, such as DevOps, containerized microservices, and enterprise office applications.

Expandable to 320 TB, SFS Turbo provides fully hosted shared file storage, which is highly available and stable, to support small files and applications requiring low latency and high IOPS.

- **Standard file protocols:** You can mount file systems as volumes to servers, the same as using local directories.
- **Data sharing:** The same file system can be mounted to multiple servers, so that data can be shared.
- **Private network:** Users can access data only in private networks of data centers.
- **Data isolation:** The on-cloud storage service provides exclusive cloud file storage, which delivers data isolation and ensures IOPS performance.
- **Use cases:** Deployments/StatefulSets in the ReadWriteMany mode, DaemonSets, and jobs created for high-traffic websites, log storage, DevOps, and enterprise OA applications

#### SFS Turbo Performance

For details about the performance parameters of SFS Turbo, see [File System Types](#).

#### Application Scenarios

SFS Turbo supports the following mounting modes:

- **Using an Existing SFS Turbo File System Through a Static PV:** static creation mode, where you use an existing SFS volume to create a PV and then mount storage to the workload through a PVC.
- **Dynamically Creating an SFS Turbo Subdirectory Using StorageClass:** SFS Turbo allows you to dynamically create subdirectories and mount them to containers so that SFS Turbo can be shared and the SFS Turbo storage capacity can be used more economically and properly.

#### Billing

SFS Turbo does not support dynamic creation. Only created SFS Turbo volumes can be mounted. You can select the pay-per-use billing mode or yearly/monthly package as required. For pricing details about SFS Turbo pricing, see [Billing](#).

## 8.5.2 Using an Existing SFS Turbo File System Through a Static PV

SFS Turbo is a shared file system with high availability and durability. It is suitable for applications that contain massive small files and require low latency, and high IOPS. This section describes how to use an existing SFS Turbo file system to statically create PVs and PVCs for data persistence and sharing in workloads.

### Prerequisites

- You have created a cluster and installed the [CCE Container Storage \(Everest\)](#) add-on in the cluster.
- To create a cluster using commands, ensure `kubectl` is used. For details, see [Connecting to a Cluster Using kubectl](#).
- You have created an available SFS Turbo file system, and the SFS Turbo file system and the cluster are in the same VPC.

### Notes and Constraints

- Multiple PVs can use the same SFS or SFS Turbo file system with the following restrictions:
  - Do not mount the PVCs/PVs that use the same underlying SFS or SFS Turbo volume to one pod. This will lead to a pod startup failure because not all PVCs can be mounted to the pod due to the same `volumeHandle` value.
  - The `persistentVolumeReclaimPolicy` parameter in the PVs must be set to **Retain**. Otherwise, when a PV is deleted, the associated underlying volume may be deleted. In this case, other PVs associated with the underlying volume malfunction.
  - When the underlying volume is repeatedly used, enable isolation and protection for `ReadWriteMany` at the application layer to prevent data overwriting and loss.
- For SFS Turbo storage, the yearly/monthly SFS Turbo resources will not be reclaimed when the cluster or PVC is deleted. Reclaim the resources on the SFS Turbo console.

### Using an Existing SFS Turbo File System on the Console

**Step 1** Log in to the CCE console and click the cluster name to access the cluster console.

**Step 2** Statically create a PVC and PV.

1. Choose **Storage** in the navigation pane. In the right pane, click the **PVCs** tab. Click **Create PVC** in the upper right corner. In the dialog box displayed, configure PVC parameters.

Parameter	Description
PVC Type	In this example, select <b>SFS Turbo</b> .
PVC Name	Enter the PVC name, which must be unique in a namespace.

Parameter	Description
Creation Method	You can create a storage volume or use an existing storage volume to statically create a PVC based on whether a PV has been created.  In this example, select <b>Create new</b> to create both a PV and PVC on the console.
PV <sup>a</sup>	Select an existing PV in the cluster. For details about how to create a PV, see "Creating a storage volume" in <a href="#">Related Operations</a> .  You do not need to specify this parameter in this example.
SFS Turbo <sup>b</sup>	Click <b>Select SFS Turbo</b> . On the displayed page, select the SFS Turbo file system that meets your requirements and click <b>OK</b> .
PV Name <sup>b</sup>	Enter the PV name, which must be unique in the same cluster.
Access Mode <sup>b</sup>	SFS Turbo volumes support only <b>ReadWriteMany</b> , indicating that a storage volume can be mounted to multiple nodes in read/write mode. For details, see <a href="#">Volume Access Modes</a> .
Reclaim Policy <sup>b</sup>	Only <b>Retain</b> is available. This indicates that the PV is not deleted when the PVC is deleted. For details, see <a href="#">PV Reclaim Policy</a> .
Mount Options <sup>b</sup>	Enter the mounting parameter key-value pairs. For details, see <a href="#">Configuring SFS Turbo Mount Options</a> .

 **NOTE**

a: The parameter is available when **Creation Method** is set to **Use existing**.

b: The parameter is available when **Creation Method** is set to **Create new**.

2. Click **Create** to create a PVC and a PV.

You can choose **Storage** in the navigation pane and view the created PVC and PV on the **PVCs** and **PVs** tab pages, respectively.

**Step 3** Create an application.

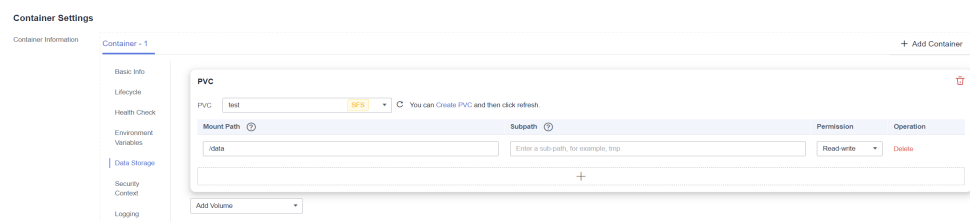
1. Choose **Workloads** in the navigation pane. In the right pane, click the **Deployments** tab.
2. Click **Create Workload** in the upper right corner. On the displayed page, click **Data Storage** in the **Container Settings** area and click **Add Volume** to select **PVC**.

Mount and use storage volumes, as shown in [Table 8-23](#). For details about other parameters, see [Workloads](#).

**Table 8-23** Mounting a storage volume

Parameter	Description
PVC	Select an existing SFS Turbo volume.
Mount Path	<p>Enter a mount path, for example, <b>/tmp</b>.</p> <p>This parameter specifies a container path to which a data volume will be mounted. Do not mount the volume to a system directory such as <b>/</b> or <b>/var/run</b>. This may lead to container errors. Mount the volume to an empty directory. If the directory is not empty, ensure that there are no files that affect container startup. Otherwise, the files will be replaced, leading to container startup failures or workload creation failures.</p> <p><b>NOTICE</b></p> <p>If a volume is mounted to a high-risk directory, use an account with minimum permissions to start the container. Otherwise, high-risk files on the host may be damaged.</p>
Subpath	Enter the subpath of the storage volume and mount a path in the storage volume to the container. In this way, different folders of the same storage volume can be used in a single pod. <b>tmp</b> , for example, indicates that data in the mount path of the container is stored in the <b>tmp</b> folder of the storage volume. If this parameter is left blank, the root path will be used by default.
Permission	<ul style="list-style-type: none"> <li>– <b>Read-only:</b> You can only read the data in the mounted volumes.</li> <li>– <b>Read-write:</b> You can modify the data volumes mounted to the path. Newly written data will not be migrated if the container is migrated, which may cause data loss.</li> </ul>

In this example, the disk is mounted to the **/data** path of the container. The container data generated in this path is stored in the SFS Turbo file system.



3. After the configuration, click **Create Workload**.

After the workload is created, the data in the container mount directory will be persistently stored. Verify the storage by referring to [Verifying Data Persistence and Sharing](#).

----End

## Using an Existing SFS Turbo File System Through kubectl

You can select a mode that meets the requirements of your service.

### Using an Existing SFS Turbo File System

**Step 1** Use kubectl to access the cluster.

**Step 2** Create a PV.

1. Create the **pv-sfsturbo.yaml** file.

Example:

```
apiVersion: v1
kind: PersistentVolume
metadata:
  annotations:
    pv.kubernetes.io/provisioned-by: everest-csi-provisioner
  name: pv-sfsturbo # PV name
spec:
  accessModes:
    - ReadWriteMany # Access mode. The value must be ReadWriteMany for SFS Turbo.
  capacity:
    storage: 500Gi # SFS Turbo volume capacity
  csi:
    driver: sfsturbo.csi.everest.io # Dependent storage driver for the mounting
    fsType: nfs
    volumeHandle: <your_volume_id> # SFS Turbo volume ID
  volumeAttributes:
    everest.io/share-export-location: <your_location> # Shared path of the SFS Turbo volume
    everest.io/enterprise-project-id: <your_project_id> # Project ID of the SFS Turbo volume
    storage.kubernetes.io/csiProvisionerIdentity: everest-csi-provisioner
  persistentVolumeReclaimPolicy: Retain # Reclaim policy
  storageClassName: csi-sfsturbo # StorageClass name of the SFS Turbo file system
  mountOptions: [] # Mount options
```

**Table 8-24** Key parameters

Parameter	Mandatory	Description
volumeHandle	Yes	ID of an SFS Turbo file system for creating a PV.  How to obtain: Log in to the CCE console, choose <b>Service List &gt; Storage &gt; Scalable File Service</b> , and select <b>SFS Turbo</b> . In the list, click the name of the target SFS Turbo file system. On the details page, copy the content following <b>ID</b> .
everest.io/share-export-location	Yes	Shared path of the SFS Turbo volume.  Log in to the CCE console, choose <b>Service List &gt; Storage &gt; Scalable File Service</b> , and select <b>SFS Turbo</b> . You can obtain the shared path of the file system.

Parameter	Mandatory	Description
everest.io/enterprise-project-id	No	Project ID of the SFS Turbo volume. How to obtain: On the SFS console, click <b>SFS Turbo</b> in the left navigation pane. Click the name of the SFS Turbo file system to interconnect. On the <b>Basic Info</b> tab, find and click the enterprise project to go to the console, and copy the ID.
mountOptions	No	Mount options. If not specified, the following configurations are used by default. For details, see <a href="#">Configuring SFS Turbo Mount Options</a> . mountOptions: - vers=3 - timeo=600 - nolock - hard
persistentVolumeReclaimPolicy	Yes	A reclaim policy is supported when the cluster version is or later than 1.19.10 and the Everest version is or later than 1.2.9. For details, see <a href="#">PV Reclaim Policy</a> . <b>Retain:</b> When a PVC is deleted, both the PV and underlying storage resources will be retained. You need to manually delete these resources. After the PVC is deleted, the PV is in the <b>Released</b> state and cannot be bound to a PVC again.
storage	Yes	Requested capacity in the PVC, in Gi.
storageClassName	Yes	The StorageClass name of SFS Turbo volumes is <b>csi-sfsturbo</b> .

- Run the following command to create a PV:  
kubectl apply -f pv-sfsturbo.yaml

### Step 3 Create a PVC.

- Create the **pvc-sfsturbo.yaml** file.

```

apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: pvc-sfsturbo
  namespace: default
  annotations:
    volume.beta.kubernetes.io/storage-provisioner: everest-csi-provisioner
    everest.io/enterprise-project-id: <your_project_id> # Project ID of the SFS Turbo volume
spec:
  accessModes:
    - ReadWriteMany # The value must be ReadWriteMany for SFS Turbo.
  resources:
    requests:
      storage: 500Gi # SFS Turbo volume capacity.
      storageClassName: csi-sfsturbo # StorageClass name of the SFS Turbo file system, which must

```



be the same as that of the PV  
**volumeName: pv-sfsturbo** # PV name

**Table 8-25** Key parameters

Parameter	Mandatory	Description
everest.io/enterprise-project-id	No	Project ID of the SFS Turbo volume. How to obtain: On the SFS console, click <b>SFS Turbo</b> in the left navigation pane. Click the name of the SFS Turbo file system to interconnect. On the <b>Basic Info</b> tab, find and click the enterprise project to go to the console, and copy the ID.
storage	Yes	Requested capacity in the PVC, in Gi. The value must be the same as the storage size of the existing PV.
storageClassName	Yes	StorageClass name, which must be the same as the StorageClass of the PV in <a href="#">1</a> . The StorageClass name of SFS Turbo volumes is <b>csi-sfsturbo</b> .
volumeName	Yes	PV name, which must be the same as the PV name in <a href="#">1</a> .

- Run the following command to create a PVC:  

```
kubectl apply -f pvc-sfsturbo.yaml
```

**Step 4** Create an application.

- Create a file named **web-demo.yaml**. In this example, the SFS Turbo volume is mounted to the **/data** path.

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: web-demo
  namespace: default
spec:
  replicas: 2
  selector:
    matchLabels:
      app: web-demo
  template:
    metadata:
      labels:
        app: web-demo
    spec:
      containers:
        - name: container-1
          image: nginx:latest
          volumeMounts:
            - name: pvc-sfsturbo-volume # Volume name, which must be the same as the volume name
              in the volumes field
              mountPath: /data # Location where the storage volume is mounted
          imagePullSecrets:
            - name: default-secret
      volumes:
```

```
- name: pvc-sfsturbo-volume # Volume name, which can be customized
persistentVolumeClaim:
  claimName: pvc-sfsturbo # Name of the created PVC
```

2. Run the following command to create a workload to which the SFS Turbo volume is mounted:

```
kubectl apply -f web-demo.yaml
```

After the workload is created, you can try [Verifying Data Persistence and Sharing](#).

----End

## Using Subdirectories of an Existing SFS Turbo File System

**Step 1** Use kubectl to access the cluster.

**Step 2** Create a PV.

1. Create the **pv-sfsturbo.yaml** file.

Example:

```
apiVersion: v1
kind: PersistentVolume
metadata:
  annotations:
    pv.kubernetes.io/provisioned-by: everest-csi-provisioner
    everest.io/reclaim-policy: retain-volume-only # When a PVC is deleted, the PV will be deleted but
the subdirectories associated with the PV will be retained. This parameter is available only when
subdirectories are used and the reclaim policy is Delete.
  name: pv-sfsturbo # PV name
spec:
  accessModes:
    - ReadWriteMany # Access mode. The value must be ReadWriteMany for SFS Turbo.
  capacity:
    storage: 500Gi # SFS Turbo volume capacity
  csi:
    driver: sfsturbo.csi.everest.io # Dependent storage driver for the mounting
    fsType: nfs
    volumeHandle: pv-sfsturbo # PV name when subdirectories are used
  volumeAttributes:
    everest.io/share-export-location: <sfsturbo_path>:<absolute_path> # Shared path and
subdirectory of the SFS Turbo file system
    everest.io/enterprise-project-id: <your_project_id> # Project ID of the SFS Turbo volume
storage.kubernetes.io/csiProvisionerIdentity: everest-csi-provisioner
    everest.io/volume-as: absolute-path # (Optional) An SFS Turbo subdirectory is used.
  persistentVolumeReclaimPolicy: Retain # Reclaim policy, which can be set to Delete when
subdirectories are automatically created
  storageClassName: csi-sfsturbo # StorageClass name of the SFS Turbo file system
  mountOptions: [] # Mount options
```

**Table 8-26** Key parameters

Parameter	Man dato ry	Description
volumeHandle	Yes	PV name when an SFS Turbo subdirectory is used to create the PV.

Parameter	Mandatory	Description
everest.io/share-export-location	Yes	<p>Shared path of the SFS Turbo subdirectory.</p> <p>Format: <code>{sfsturbo_path}/{absolute_path}</code></p> <p>How to obtain: Log in to the CCE console, choose <b>Service List &gt; Storage &gt; Scalable File Service</b>, and select <b>SFS Turbo</b>. You can obtain the shared path of the file system.</p>
everest.io/enterprise-project-id	No	<p>Project ID of the SFS Turbo volume.</p> <p>How to obtain: On the SFS console, click <b>SFS Turbo</b> in the left navigation pane. Click the name of the SFS Turbo file system to interconnect. On the <b>Basic Info</b> tab, find and click the enterprise project to go to the console, and copy the ID.</p>
mountOptions	No	<p>Mount options.</p> <p>If not specified, the following configurations are used by default. For details, see <a href="#">Configuring SFS Turbo Mount Options</a>.</p> <p>mountOptions: - vers=3 - timeo=600 - nolock - hard</p>
persistentVolumeReclaimPolicy	Yes	<p>A reclaim policy is supported when the cluster version is or later than 1.19.10 and the Everest version is or later than 1.2.9. For details, see <a href="#">PV Reclaim Policy</a>.</p> <p><b>Retain:</b> When a PVC is deleted, both the PV and underlying storage resources will be retained. You need to manually delete these resources. After the PVC is deleted, the PV is in the <b>Released</b> state and cannot be bound to a PVC again.</p> <p><b>Delete:</b> This parameter can be configured when subdirectories are automatically created, indicating that the PV is deleted when a PVC is deleted.</p>

Parameter	Mandatory	Description
everest.io/reclaim-policy	No	Whether to retain subdirectories when deleting a PVC. This parameter must be used with <b>PV Reclaim Policy</b> . This parameter is available only when the PV reclaim policy is <b>Delete</b> . Options: <ul style="list-style-type: none"> <li><b>retain-volume-only</b>: If a PVC is deleted, the PV will be deleted, but <b>the subdirectories associated with the PV will be retained</b>.</li> <li><b>delete</b>: After a PVC is deleted, <b>the PV and its associated subdirectories will also be deleted</b>.</li> </ul> <b>NOTE</b> When a subdirectory is deleted, only the absolute path of the subdirectory configured in the PVC will be deleted. The upper-layer directory will not be deleted.
everest.io/volume-as	No	The value is fixed at <b>absolute-path</b> , indicating that a dynamically created SFS Turbo subdirectory is used. Ensure Everest of v2.3.23 or later has been installed in the cluster.
storage	Yes	Requested capacity in the PVC, in Gi. If a subdirectory is used, this parameter serves no purpose other than for verification and must have a non-empty, non-zero value.
storageClassName	Yes	The StorageClass name of SFS Turbo volumes is <b>csi-sfsturbo</b> .

- Run the following command to create a PV:  

```
kubectl apply -f pv-sfsturbo.yaml
```

### Step 3 Create a PVC.

- Create the **pvc-sfsturbo.yaml** file.

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: pvc-sfsturbo
  namespace: default
  annotations:
    volume.beta.kubernetes.io/storage-provisioner: everest-csi-provisioner
    everest.io/enterprise-project-id: <your_project_id> # Project ID of the SFS Turbo volume
spec:
  accessModes:
    - ReadWriteMany # The value must be ReadWriteMany for SFS Turbo.
  resources:
    requests:
      storage: 500Gi # SFS Turbo volume capacity.
      storageClassName: csi-sfsturbo # StorageClass name of the SFS Turbo file system, which must
```

be the same as that of the PV  
**volumeName: pv-sfsturbo** # PV name

**Table 8-27** Key parameters

Parameter	Mandatory	Description
everest.io/enterprise-project-id	No	Project ID of the SFS Turbo volume. How to obtain: On the SFS console, click <b>SFS Turbo</b> in the left navigation pane. Click the name of the SFS Turbo file system to interconnect. On the <b>Basic Info</b> tab, find and click the enterprise project to go to the console, and copy the ID.
storage	Yes	Requested capacity in the PVC, in Gi. The value must be the same as the storage size of the existing PV.
storageClassName	Yes	StorageClass name, which must be the same as the StorageClass of the PV in <a href="#">1</a> . The StorageClass name of SFS Turbo volumes is <b>csi-sfsturbo</b> .
volumeName	Yes	PV name, which must be the same as the PV name in <a href="#">1</a> .

- Run the following command to create a PVC:  

```
kubectl apply -f pvc-sfsturbo.yaml
```

**Step 4** Create an application.

- Create a file named **web-demo.yaml**. In this example, the SFS Turbo volume is mounted to the **/data** path.

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: web-demo
  namespace: default
spec:
  replicas: 2
  selector:
    matchLabels:
      app: web-demo
  template:
    metadata:
      labels:
        app: web-demo
    spec:
      containers:
        - name: container-1
          image: nginx:latest
          volumeMounts:
            - name: pvc-sfsturbo-volume # Volume name, which must be the same as the volume name
              in the volumes field
              mountPath: /data # Location where the storage volume is mounted
          imagePullSecrets:
            - name: default-secret
```

```
volumes:
  - name: pvc-sfsturbo-volume # Volume name, which can be customized
    persistentVolumeClaim:
      claimName: pvc-sfsturbo # Name of the created PVC
```

2. Run the following command to create a workload to which the SFS Turbo volume is mounted:

```
kubectl apply -f web-demo.yaml
```

After the workload is created, you can try [Verifying Data Persistence and Sharing](#).

----End

## Verifying Data Persistence and Sharing

### Step 1 View the deployed application and files.

1. Run the following command to view the created pod:

```
kubectl get pod | grep web-demo
```

Expected output:

```
web-demo-846b489584-mjhm9 1/1 Running 0 46s
web-demo-846b489584-wvv5s 1/1 Running 0 46s
```

2. Run the following commands in sequence to view the files in the `/data` path of the pods:

```
kubectl exec web-demo-846b489584-mjhm9 -- ls /data
kubectl exec web-demo-846b489584-wvv5s -- ls /data
```

If no result is returned for both pods, no file exists in the `/data` path.

### Step 2 Run the following command to create a file named `static` in the `/data` path:

```
kubectl exec web-demo-846b489584-mjhm9 -- touch /data/static
```

### Step 3 Run the following command to check the files in the `/data` path:

```
kubectl exec web-demo-846b489584-mjhm9 -- ls /data
```

Expected output:

```
static
```

### Step 4 Verify data persistence.

1. Run the following command to delete the pod named `web-demo-846b489584-mjhm9`:

```
kubectl delete pod web-demo-846b489584-mjhm9
```

Expected output:

```
pod "web-demo-846b489584-mjhm9" deleted
```

After the deletion, the Deployment controller automatically creates a replica.

2. Run the following command to view the created pod:

```
kubectl get pod | grep web-demo
```

The expected output is as follows, in which `web-demo-846b489584-d4d4j` is the newly created pod:

```
web-demo-846b489584-d4d4j 1/1 Running 0 110s
web-demo-846b489584-wvv5s 1/1 Running 0 7m50s
```

3. Run the following command to check whether the files in the `/data` path of the new pod have been modified:

```
kubectl exec web-demo-846b489584-d4d4j -- ls /data
```

Expected output:

```
static
```

The **static** file is retained, indicating that the data in the file system can be stored persistently.

### Step 5 Verify data sharing.

1. Run the following command to view the created pod:

```
kubectl get pod | grep web-demo
```

Expected output:

```
web-demo-846b489584-d4d4j 1/1 Running 0 7m
web-demo-846b489584-wvv5s 1/1 Running 0 13m
```

2. Run the following command to create a file named **share** in the **/data** path of either pod: In this example, select the pod named **web-demo-846b489584-d4d4j**.

```
kubectl exec web-demo-846b489584-d4d4j -- touch /data/share
```

Check the files in the **/data** path of the pod.

```
kubectl exec web-demo-846b489584-d4d4j -- ls /data
```

Expected output:

```
share
static
```

3. Check whether the **share** file exists in the **/data** path of another pod (**web-demo-846b489584-wvv5s**) as well to verify data sharing.

```
kubectl exec web-demo-846b489584-wvv5s -- ls /data
```

Expected output:

```
share
static
```

After you create a file in the **/data** path of a pod, if the file is also created in the **/data** path of the other pod, the two pods share the same volume.

----End

## Related Operations

You can also perform the operations listed in [Table 8-28](#).

**Table 8-28** Related operations

Operation	Description	Procedure
Creating a storage volume (PV)	Create a PV on the CCE console.	<ol style="list-style-type: none"> <li>Choose <b>Storage</b> in the navigation pane. In the right pane, click the <b>PVs</b> tab. Click <b>Create PersistentVolume</b> in the upper right corner. In the dialog box displayed, configure parameters. <ul style="list-style-type: none"> <li><b>Volume Type:</b> Select <b>SFS Turbo</b>.</li> <li><b>SFS Turbo:</b> Click <b>Select SFS Turbo</b>. On the page displayed, select the SFS Turbo file system that meets your requirements and click <b>OK</b>.</li> <li><b>PV Name:</b> Enter the PV name, which must be unique in a cluster.</li> <li><b>Access Mode:</b> SFS volumes support only <b>ReadWriteMany</b>, indicating that a storage volume can be mounted to multiple nodes in read/write mode. For details, see <a href="#">Volume Access Modes</a>.</li> <li><b>Reclaim Policy:</b> Only <b>Retain</b> is supported. For details, see <a href="#">PV Reclaim Policy</a>.</li> <li><b>Mount Options:</b> Enter the mounting parameter key-value pairs. For details, see <a href="#">Configuring SFS Turbo Mount Options</a>.</li> </ul> </li> <li>Click <b>Create</b>.</li> </ol>
Expanding the capacity of an SFS Turbo volume	Quickly expand the capacity of a mounted SFS Turbo volume on the CCE console.	<ol style="list-style-type: none"> <li>Choose <b>Storage</b> in the navigation pane. In the right pane, click the <b>PVCs</b> tab. Click <b>More</b> in the <b>Operation</b> column of the target PVC and select <b>Scale-out</b>.</li> <li>Enter the capacity to be added and click <b>OK</b>.</li> </ol>
Viewing events	View event names, event types, number of occurrences, Kubernetes events, first occurrence time, and last occurrence time of the PVC or PV.	<ol style="list-style-type: none"> <li>Choose <b>Storage</b> in the navigation pane. In the right pane, click the <b>PVCs</b> or <b>PVs</b> tab.</li> <li>Click <b>View Events</b> in the <b>Operation</b> column of the target PVC or PV to view events generated within one hour (events are retained for one hour).</li> </ol>



Operation	Description	Procedure
Viewing a YAML file	View, copy, or download the YAML file of a PVC or PV.	<ol style="list-style-type: none"> <li>1. Choose <b>Storage</b> in the navigation pane. In the right pane, click the <b>PVCs</b> or <b>PVs</b> tab.</li> <li>2. Click <b>View YAML</b> in the <b>Operation</b> column of the target PVC or PV to view or download the YAML.</li> </ol>

### 8.5.3 Configuring SFS Turbo Mount Options

This section describes how to configure SFS Turbo mount options. For SFS Turbo, you can only set mount options in a PV and bind the PV by creating a PVC.

#### Prerequisites

The **CCE Container Storage (Everest)** version must be **1.2.8 or later**. This add-on identifies the mount options and transfers them to the underlying storage resources. The parameter settings take effect only if the underlying storage resources support the specified options.

#### Notes and Constraints

- Due to the restrictions of the NFS protocol, if an SFS volume is mounted to a node for multiple times, link-related mounting parameters (such as **timeo**) take effect only when the SFS volume is mounted for the first time by default. For example, if the same SFS file system is mounted to multiple pods running on a node, the mounting parameter set later does not overwrite the existing parameter value. If you want to configure different mounting parameters in the preceding scenario, additionally configure the **nosharecache** parameter.

#### SFS Turbo Mount Options

The Everest add-on in CCE presets the options described in [Table 8-29](#) for mounting SFS Turbo volumes.

**Table 8-29** SFS Turbo mount options

Parameter	Value	Description
vers	3	File system version. Currently, only NFSv3 is supported. Value: <b>3</b>
noLOCK	Blank	Whether to lock files on the server using the NLM protocol. If <b>noLOCK</b> is selected, the lock is valid for applications on one host. For applications on another host, the lock is invalid.
timeo	600	Waiting time before the NFS client retransmits a request. The unit is 0.1 seconds. Recommended value: <b>600</b>

Parameter	Value	Description
hard/soft	Blank	Mount mode. <ul style="list-style-type: none"> <li><b>hard</b>: If the NFS request times out, the client keeps resending the request until the request is successful.</li> <li><b>soft</b>: If the NFS request times out, the client returns an error to the invoking program.</li> </ul> The default value is <b>hard</b> .
sharecache/nosharecache	Blank	How the data cache and attribute cache are shared when one file system is concurrently mounted to different clients. If this parameter is set to <b>sharecache</b> , the caches are shared between the mountings. If this parameter is set to <b>nosharecache</b> , the caches are not shared, and one cache is configured for each client mounting. The default value is <b>sharecache</b> . <p><b>NOTE</b></p> The <b>nosharecache</b> setting will affect the performance. The mounting information must be obtained for each mounting, which increases the communication overhead with the NFS server and the memory consumption of the NFS clients. In addition, the <b>nosharecache</b> setting on the NFS clients may lead to inconsistent caches. Determine whether to use <b>nosharecache</b> based on site requirements.

## Configuring Mount Options in a PV

You can use the **mountOptions** field to configure mount options in a PV. The options you can configure in **mountOptions** are listed in [SFS Turbo Mount Options](#).

**Step 1** Use `kubectl` to access the cluster. For details, see [Connecting to a Cluster Using kubectl](#).

**Step 2** Configure mount options in a PV. Example:

```
apiVersion: v1
kind: PersistentVolume
metadata:
  annotations:
    pv.kubernetes.io/provisioned-by: everest-csi-provisioner
  name: pv-sfsturbo # PV name
spec:
  accessModes:
    - ReadWriteMany # Access mode. The value must be ReadWriteMany for SFS Turbo.
  capacity:
    storage: 500Gi # SFS Turbo volume capacity
  csi:
    driver: sfsturbo.csi.everest.io # Dependent storage driver for the mounting
    fsType: nfs
    volumeHandle: {your_volume_id} # SFS Turbo volume ID
  volumeAttributes:
    everest.io/share-export-location: {your_location} # Shared path of the SFS Turbo volume
    everest.io/enterprise-project-id: {your_project_id} # Project ID of the SFS Turbo volume
```

```
storage.kubernetes.io/csiProvisionerIdentity: everest-csi-provisioner
persistentVolumeReclaimPolicy: Retain # Reclaim policy
storageClassName: csi-sfsturbo # StorageClass name of the SFS Turbo file system
mountOptions: # Mount options
- vers=3
- nolock
- timeo=600
- hard
```

**Step 3** After a PV is created, you can create a PVC and bind it to the PV, and then mount the PV to the container in the workload. For details, see [Using an Existing SFS Turbo File System Through a Static PV](#).

**Step 4** Check whether the mount options take effect.

In this example, the PVC is mounted to the workload that uses the **nginx:latest** image. You can run the **mount -l** command to check whether the mount options take effect.

1. View the pod to which the SFS Turbo volume has been mounted. In this example, the workload name is **web-sfsturbo**.

```
kubectl get pod | grep web-sfsturbo
```

The command output is as follows:

```
web-sfsturbo-*** 1/1 Running 0 23m
```

2. Run the following command to check the mount options (**web-sfsturbo-\*\*\*** is an example pod):

```
kubectl exec -it web-sfsturbo-*** -- mount -l | grep nfs
```

If the mounting information in the command output is consistent with the configured mount options, the mount options have been configured.

```
<Your mount path> on /data type nfs
```

```
(rw,relatime,vers=3,rsize=1048576,wsiz=1048576,namlen=255,hard,nolock,norevport,proto=tcp,timeo=600,retrans=2,sec=sys,mountaddr=*.*.*.*,mountvers=3,mountport=20048,mountproto=tcp,local_lock=all,addr=*.*.*.*)
```

----End

## 8.5.4 Dynamically Creating an SFS Turbo Subdirectory Using StorageClass

### 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.

### 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
mountOptions:
- 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: 7ca2dba2-1234-1234-1234-626371a8fb3a
  everest.io/share-expand-type: bandwidth
  everest.io/share-export-location: 192.168.1.1/sfsturbo/
  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
provisioner: everest-csi-provisioner
reclaimPolicy: Delete
volumeBindingMode: Immediate
```

In this example:

- **name**: indicates the name of the StorageClass.
- **mountOptions**: indicates the mount options. This field is optional.
  - 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 Volume Mount Options](#). **Do not set noLock to true. Otherwise, the mount operation will fail.**

```
mountOptions:
- vers=3
- timeo=600
- 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/share-expand-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
  namespace: default
spec:
  accessModes:
    - ReadWriteMany
  resources:
    requests:
      storage: 50Gi
  storageClassName: sfsturbo-subpath-sc
  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
  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
          name: container-0
          volumeMounts:
            - mountPath: /tmp
              name: pvc-sfs-turbo-example
      restartPolicy: Always
      imagePullSecrets:
        - name: default-secret
      volumes:
        - name: pvc-sfs-turbo-example
          persistentVolumeClaim:
            claimName: sfs-turbo-test
```

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
  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'
          resources: {}
          volumeMounts:
            - name: sfs-turbo-160024548582479676
              mountPath: /tmp
          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:
          - ReadWriteOnce
        resources:
          requests:
            storage: 10Gi
        storageClassName: sfsturbo-subpath-sc
      serviceName: www
      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
```

## 8.6 Object Storage Service

### 8.6.1 Overview

#### Introduction

Object Storage Service (OBS) provides massive, secure, and cost-effective data storage for you to store data of any type and size. You can use it in enterprise backup/archiving, video on demand (VoD), video surveillance, and many other scenarios.

- **Standard APIs**: With HTTP RESTful APIs, OBS allows you to use client tools or third-party tools to access object storage.
- **Data sharing**: Servers, embedded devices, and IoT devices can use the same path to access shared object data in OBS.
- **Public/Private networks**: OBS allows data to be accessed from public networks to meet Internet application requirements.
- **Capacity and performance**: No capacity limit; high performance (I/O latency within 10 ms).
- **Use cases**: Deployments/StatefulSets in the **ReadOnlyMany** mode and jobs created for big data analysis, static website hosting, online VOD, gene sequencing, intelligent video surveillance, backup and archiving, and enterprise cloud boxes (web disks). You can create object storage by using the OBS console, tools, and SDKs.

#### OBS Specifications

OBS provides multiple storage classes to meet customers' requirements on storage performance and costs.

- Object buckets provide reliable, high-performance, secure, and budget-friendly storage for data. They have no restrictions on the quantity of files or storage capacity.
  - **Standard**: features low latency and high throughput. It is therefore good for storing frequently (multiple times per month) accessed files or small files (less than 1 MB). Its application scenarios include big data analytics, mobile apps, hot videos, and social apps.



- OBS Infrequent Access: applicable to storing semi-frequently accessed (less than 12 times a year) data requiring quick response. Its application scenarios include file synchronization or sharing, and enterprise-level backup. This storage class has the same durability, low latency, and high throughput as the Standard storage class, with a lower cost, but its availability is slightly lower than the Standard storage class.
- Parallel file systems are a sub-product of OBS designed to provide high-performance file semantics for big data scenarios.

For details about OBS storage classes, see [Storage Classes](#).

## Performance

Every time an OBS volume is mounted to a container workload, a resident process is created in the backend. When a workload uses too many OBS volumes or reads and writes a large number of object storage files, resident processes will consume a significant amount of memory. The amount of memory required in these scenarios is listed [Table 8-30](#). To ensure stable running of the workload, make sure that the number of OBS volumes used does not exceed the requested memory. For example, if the workload requests for 4 GiB of memory, the number of OBS volumes should be **no more than 4**.

**Table 8-30** Memory required by a single object storage resident process

Test Item	Memory Usage
Long-term stable running	About 50 MiB
Concurrent write to a 10 MB file from two processes	About 110 MiB
Concurrent write to a 10 MB file from four processes	About 220 MiB
Write to a 100 GB file from a single process	About 300 MiB

## Application Scenarios

OBS supports the following mounting modes based on application scenarios:

- **Using an Existing OBS Bucket Through a Static PV:** static creation mode, where you use an existing OBS volume to create a PV and then mount storage to the workload through a PVC. This mode applies if the underlying storage is available or billed on a yearly/monthly basis.
- **Using an OBS Bucket Through a Dynamic PV:** dynamic creation mode, in which you do not need to create OBS volumes beforehand. Instead, specify a StorageClass when creating a PVC. Then, an OBS volume and PV will be created automatically. This mode applies to scenarios where no underlying storage is available.

## Billing

- The default billing mode of an OBS volume **automatically created** using StorageClass is **pay-per-use**. For details about OBS pricing, see [OBS Pricing Details](#).
- To use a yearly/monthly-billed OBS volume, [use an existing one](#).

## 8.6.2 Using an Existing OBS Bucket Through a Static PV

This section describes how to use an existing Object Storage Service (OBS) bucket to statically create PVs and PVCs for data persistence and sharing in workloads.

### Prerequisites

- You have created a cluster and installed the [CCE Container Storage \(Everest\)](#) add-on in the cluster.
- To create a cluster using commands, ensure kubectl is used. For details, see [Connecting to a Cluster Using kubectl](#).

### Notes and Constraints

- If OBS volumes are used, the owner group and permission of the mount point cannot be modified.
- Every time an OBS volume is mounted to a workload through a PVC, a resident process is created in the backend. When a workload uses too many OBS volumes or reads and writes a large number of object storage files, resident processes will consume a significant amount of memory. To ensure stable running of the workload, make sure that the number of OBS volumes used does not exceed the requested memory. For example, if the workload requests for 4 GiB of memory, the number of OBS volumes should be **no more than 4**.
- Kata containers do not support OBS volumes.
- Hard links are not supported when common buckets are mounted.
- Multiple PVs can use the same OBS storage volume with the following restrictions:
  - Do not mount the PVCs/PVs that use the same underlying OBS volume to one pod. This will lead to a pod startup failure because not all PVCs can be mounted to the pod due to the same **volumeHandle** value.
  - The **persistentVolumeReclaimPolicy** parameter in the PVs must be set to **Retain**. Otherwise, when a PV is deleted, the associated underlying volume may be deleted. In this case, other PVs associated with the underlying volume malfunction.
  - If underlying storage is repeatedly used, you are required to maintain data consistency. Enable isolation and protection for ReadWriteMany at the application layer and prevent multiple clients from writing the same file to prevent data overwriting and loss.

## Using an Existing OBS Bucket on the Console

**Step 1** Log in to the CCE console and click the cluster name to access the cluster console.

**Step 2** Statically create a PVC and PV.

1. Choose **Storage** in the navigation pane. In the right pane, click the **PVCs** tab. Click **Create PVC** in the upper right corner. In the dialog box displayed, configure PVC parameters.

Parameter	Description
PVC Type	In this example, select <b>OBS</b> .
PVC Name	Enter the PVC name, which must be unique in a namespace.
Creation Method	<ul style="list-style-type: none"> <li>- If underlying storage is available, create a PV or use an existing PV to statically create a PVC.</li> <li>- If no underlying storage is available, select <b>Dynamically provision</b>. For details, see <a href="#">Using an OBS Bucket Through a Dynamic PV</a>.</li> </ul> <p>In this example, select <b>Create new</b> to create both a PV and PVC on the console.</p>
PV <sup>a</sup>	<p>Select an existing PV in the cluster. For details about how to create a PV, see "Creating a storage volume" in <a href="#">Related Operations</a>.</p> <p>You do not need to specify this parameter in this example.</p>
OBS <sup>b</sup>	Click <b>Select OBS</b> . On the displayed page, select the OBS volume that meets your requirements and click <b>OK</b> .
PV Name <sup>b</sup>	Enter the PV name, which must be unique in the same cluster.
Access Mode <sup>b</sup>	OBS volumes support only <b>ReadWriteMany</b> , indicating that a storage volume can be mounted to multiple nodes in read/write mode. For details, see <a href="#">Volume Access Modes</a> .
Reclaim Policy <sup>b</sup>	<p>You can select <b>Delete</b> or <b>Retain</b> to specify the reclaim policy of the underlying storage when the PVC is deleted. For details, see <a href="#">PV Reclaim Policy</a>.</p> <p><b>NOTE</b></p> <p>If multiple PVs use the same OBS volume, use <b>Retain</b> to prevent the underlying volume from being deleted with a PV.</p>
Access Key (AK/SK) <sup>b</sup>	<p><b>Custom</b> (Recommended): Customize a secret if you want to assign different user permissions to different OBS storage devices. For details, see <a href="#">Using a Custom Access Key (AK/SK) to Mount an OBS Volume</a>.</p> <p>Only secrets with the <b>secret.kubernetes.io/used-by = csi</b> label can be selected. The secret type is <b>cfe/secure-opaque</b>. If no secret is available, click <b>Create Secret</b> to create one.</p> <ul style="list-style-type: none"> <li>- <b>Name</b>: Enter a secret name.</li> <li>- <b>Namespace</b>: Select the namespace where the secret is.</li> <li>- <b>Access Key (AK/SK)</b>: Upload a key file in .csv format. For details, see <a href="#">Obtaining an Access Key</a>.</li> </ul>

Parameter	Description
Mount Options <sup>b</sup>	Enter the mounting parameter key-value pairs. For details, see <a href="#">Configuring OBS Mount Options</a> .

 NOTE

- a: The parameter is available when **Creation Method** is set to **Use existing**.
- b: The parameter is available when **Creation Method** is set to **Create new**.

2. Click **Create** to create a PVC and a PV.

You can choose **Storage** in the navigation pane and view the created PVC and PV on the **PVCs** and **PVs** tab pages, respectively.

**Step 3** Create an application.

1. Choose **Workloads** in the navigation pane. In the right pane, click the **Deployments** tab.
2. Click **Create Workload** in the upper right corner. On the displayed page, click **Data Storage** in the **Container Settings** area and click **Add Volume** to select **PVC**.

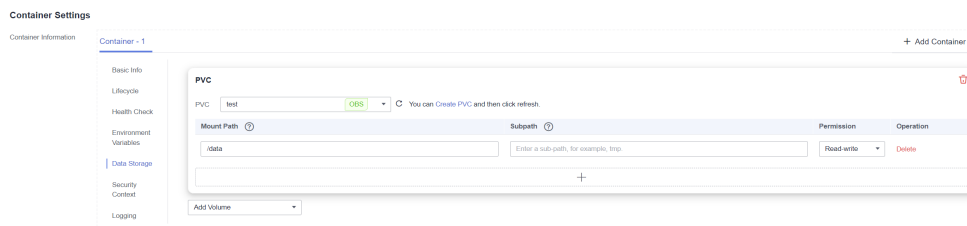
Mount and use storage volumes, as shown in [Table 8-31](#). For details about other parameters, see [Workloads](#).

**Table 8-31** Mounting a storage volume

Parameter	Description
PVC	Select an existing OBS volume.
Mount Path	<p>Enter a mount path, for example, <b>/tmp</b>.</p> <p>This parameter specifies a container path to which a data volume will be mounted. Do not mount the volume to a system directory such as <b>/</b> or <b>/var/run</b>. This may lead to container errors. Mount the volume to an empty directory. If the directory is not empty, ensure that there are no files that affect container startup. Otherwise, the files will be replaced, leading to container startup failures or workload creation failures.</p> <p><b>NOTICE</b></p> <p>If a volume is mounted to a high-risk directory, use an account with minimum permissions to start the container. Otherwise, high-risk files on the host may be damaged.</p>
Subpath	<p>Enter the subpath of the storage volume and mount a path in the storage volume to the container. In this way, different folders of the same storage volume can be used in a single pod. <b>tmp</b>, for example, indicates that data in the mount path of the container is stored in the <b>tmp</b> folder of the storage volume. If this parameter is left blank, the root path will be used by default.</p>

Parameter	Description
Permission	<ul style="list-style-type: none"> <li>– <b>Read-only:</b> You can only read the data in the mounted volumes.</li> <li>– <b>Read-write:</b> You can modify the data volumes mounted to the path. Newly written data will not be migrated if the container is migrated, which may cause data loss.</li> </ul>

In this example, the disk is mounted to the `/data` path of the container. The container data generated in this path is stored in the OBS volume.



3. After the configuration, click **Create Workload**.

After the workload is created, the data in the container mount directory will be persistently stored. Verify the storage by referring to [Verifying Data Persistence and Sharing](#).

----End

## Using an Existing OBS Bucket Through kubectl

**Step 1** Use kubectl to access the cluster.

**Step 2** Create a PV.

1. Create the `pv-obs.yaml` file.

```

apiVersion: v1
kind: PersistentVolume
metadata:
  annotations:
    pv.kubernetes.io/provisioned-by: everest-csi-provisioner
    everest.io/reclaim-policy: retain-volume-only # (Optional) The underlying volume is retained
    when the PV is deleted.
  name: pv-obs # PV name
spec:
  accessModes:
    - ReadWriteMany # Access mode. The value must be ReadWriteMany for OBS.
  capacity:
    storage: 1Gi # OBS volume capacity
  csi:
    driver: obs.csi.everest.io # Dependent storage driver for the mounting
    fsType: obsfs # Instance type
    volumeHandle: <your_volume_id> # Name of the OBS volume
  volumeAttributes:
    storage.kubernetes.io/csiProvisionerIdentity: everest-csi-provisioner
    everest.io/obs-volume-type: STANDARD
    everest.io/region: <your_region> # Region where the OBS volume is
    everest.io/enterprise-project-id: <your_project_id> # (Optional) Enterprise project ID. If an
    enterprise project is specified, use the same enterprise project when creating a PVC. Otherwise, the
    PVC cannot be bound to a PV.
  nodePublishSecretRef:
    name: <your_secret_name> # Custom secret of the OBS volume
    name: <your_secret_name> # Custom secret name

```

```
namespace: <your_namespace> # Namespace of the custom secret
persistentVolumeReclaimPolicy: Retain # Reclaim policy
storageClassName: csi-obs # StorageClass name
mountOptions: [] # Mount options
```

**Table 8-32** Key parameters

Parameter	Mandatory	Description
everest.io/reclaim-policy: retain-volume-only	No	Optional. Only <b>retain-volume-only</b> is supported. This parameter is valid only when the Everest version is 1.2.9 or later and the reclaim policy is <b>Delete</b> . If the reclaim policy is <b>Delete</b> and the current value is <b>retain-volume-only</b> , the associated PV is deleted while the underlying storage volume is retained, when a PVC is deleted.
fsType	Yes	Instance type. The value can be <b>obsfs</b> or <b>s3fs</b> . <ul style="list-style-type: none"> <li>- <b>obsfs</b>: a parallel file system</li> <li>- <b>s3fs</b>: object bucket</li> </ul>
volumeHandle	Yes	OBS volume name.
everest.io/obs-volume-type	Yes	OBS storage class. <ul style="list-style-type: none"> <li>- If <b>fsType</b> is set to <b>s3fs</b>, <b>STANDARD</b> (standard bucket) and <b>WARM</b> (infrequent access bucket) are supported.</li> <li>- This parameter is invalid when <b>fsType</b> is set to <b>obsfs</b>.</li> </ul>
everest.io/region	Yes	Region where the OBS bucket is deployed.
everest.io/enterprise-project-id	No	Optional. Enterprise project ID of OBS. If an enterprise project is specified, use the same enterprise project when creating a PVC. Otherwise, the PVC cannot be bound to a PV.  <b>How to obtain:</b> On the OBS console, choose <b>Buckets</b> or <b>Parallel File Systems</b> in the navigation pane. Click the name of the OBS bucket to access its details page. In the <b>Basic Information</b> area, locate the enterprise project and click it to access the enterprise project console. Copy the corresponding ID to obtain the ID of the enterprise project to which the object storage belongs.

Parameter	Mandatory	Description
nodePublishSecretRef	No	<p>Access key (AK/SK) used for mounting the object storage volume. You can use the AK/SK to create a secret and mount it to the PV. For details, see <a href="#">Using a Custom Access Key (AK/SK) to Mount an OBS Volume</a>.</p> <p>An example is as follows:</p> <pre>nodePublishSecretRef:   name: secret-demo   namespace: default</pre>
mountOptions	No	Mount options. For details, see <a href="#">Configuring OBS Mount Options</a> .
persistentVolumeReclaimPolicy	Yes	<p>A reclaim policy is supported when the cluster version is or later than 1.19.10 and the Everest version is or later than 1.2.9.</p> <p>The <b>Delete</b> and <b>Retain</b> reclaim policies are supported. For details, see <a href="#">PV Reclaim Policy</a>. If multiple PVs use the same OBS volume, use <b>Retain</b> to prevent the underlying volume from being deleted with a PV.</p> <p><b>Delete:</b></p> <ul style="list-style-type: none"> <li>- If <b>everest.io/reclaim-policy</b> is not specified, both the PV and storage resources will be deleted when a PVC is deleted.</li> <li>- If <b>everest.io/reclaim-policy</b> is set to <b>retain-volume-only</b>, when a PVC is deleted, the PV will be deleted but the storage resources will be retained.</li> </ul> <p><b>Retain:</b> When a PVC is deleted, both the PV and underlying storage resources will be retained. You need to manually delete these resources. After the PVC is deleted, the PV is in the <b>Released</b> state and cannot be bound to a PVC again.</p>
storage	Yes	<p>Storage capacity, in Gi.</p> <p>For OBS, this field is used only for verification (cannot be empty or 0). Its value is fixed at <b>1</b>, and any value you set does not take effect for OBS.</p>
storageClassName	Yes	StorageClass name, which is <b>csi-obs</b> for an OBS volume.

2. Run the following command to create a PV:

```
kubectl apply -f pv-obs.yaml
```

### Step 3 Create a PVC.

1. Create the **pvc-obs.yaml** file.

```

apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: pvc-obs
  namespace: default
  annotations:
    volume.beta.kubernetes.io/storage-provisioner: everest-csi-provisioner
    everest.io/obs-volume-type: STANDARD
    csi.storage.k8s.io/fstype: obsfs
    csi.storage.k8s.io/node-publish-secret-name: <your_secret_name> # Custom secret name.
    csi.storage.k8s.io/node-publish-secret-namespace: <your_namespace> # Namespace of the
    custom secret.
    everest.io/enterprise-project-id: <your_project_id> # (Optional) Enterprise project ID. If an
    enterprise project is specified, use the same enterprise project when creating a PVC. Otherwise, the
    PVC cannot be bound to a PV.
spec:
  accessModes:
    - ReadWriteMany # The value must be ReadWriteMany for OBS.
  resources:
    requests:
      storage: 1Gi
  storageClassName: csi-obs # StorageClass name, which must be the same as that of the PV
  volumeName: pv-obs # PV name
  
```

**Table 8-33** Key parameters

Parameter	Mandatory	Description
csi.storage.k8s.io/node-publish-secret-name	No	Name of the custom secret specified in the PV.
csi.storage.k8s.io/node-publish-secret-namespace	No	Namespace of the custom secret specified in the PV.
everest.io/enterprise-project-id	No	Project ID of OBS. <b>How to obtain:</b> On the OBS console, choose <b>Buckets</b> or <b>Parallel File Systems</b> in the navigation pane. Click the name of the OBS bucket to access its details page. In the <b>Basic Information</b> area, locate the enterprise project and click it to access the enterprise project console. Copy the corresponding ID to obtain the ID of the enterprise project to which the object storage belongs.
storage	Yes	Requested capacity in the PVC, in Gi. For OBS, this field is used only for verification (cannot be empty or 0). Its value is fixed at <b>1</b> , and any value you set does not take effect for OBS.



Parameter	Mandatory	Description
storageClassName	Yes	StorageClass name, which must be the same as the StorageClass of the PV in <a href="#">1</a> . StorageClass name, which is <b>csi-obs</b> for an OBS volume.
volumeName	Yes	PV name, which must be the same as the PV name in <a href="#">1</a> .

- Run the following command to create a PVC:  

```
kubectl apply -f pvc-obs.yaml
```

#### Step 4 Create an application.

- Create a file named **web-demo.yaml**. In this example, the OBS volume is mounted to the **/data** path.

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: web-demo
  namespace: default
spec:
  replicas: 2
  selector:
    matchLabels:
      app: web-demo
  template:
    metadata:
      labels:
        app: web-demo
    spec:
      containers:
        - name: container-1
          image: nginx:latest
          volumeMounts:
            - name: pvc-obs-volume # Volume name, which must be the same as the volume name in the
volumes field
              mountPath: /data # Location where the storage volume is mounted
          imagePullSecrets:
            - name: default-secret
          volumes:
            - name: pvc-obs-volume # Volume name, which can be customized
              persistentVolumeClaim:
                claimName: pvc-obs # Name of the created PVC
```

- Run the following command to create a workload to which the OBS volume is mounted:

```
kubectl apply -f web-demo.yaml
```

After the workload is created, you can try [Verifying Data Persistence and Sharing](#).

----End

## Verifying Data Persistence and Sharing

### Step 1 View the deployed application and files.

- Run the following command to view the created pod:  

```
kubectl get pod | grep web-demo
```

Expected output:

```
web-demo-846b489584-mjhm9 1/1 Running 0 46s
web-demo-846b489584-wvv5s 1/1 Running 0 46s
```

2. Run the following commands in sequence to view the files in the **/data** path of the pods:

```
kubectl exec web-demo-846b489584-mjhm9 -- ls /data
kubectl exec web-demo-846b489584-wvv5s -- ls /data
```

If no result is returned for both pods, no file exists in the **/data** path.

**Step 2** Run the following command to create a file named **static** in the **/data** path:

```
kubectl exec web-demo-846b489584-mjhm9 -- touch /data/static
```

**Step 3** Run the following command to check the files in the **/data** path:

```
kubectl exec web-demo-846b489584-mjhm9 -- ls /data
```

Expected output:

```
static
```

**Step 4 Verify data persistence.**

1. Run the following command to delete the pod named **web-demo-846b489584-mjhm9**:

```
kubectl delete pod web-demo-846b489584-mjhm9
```

Expected output:

```
pod "web-demo-846b489584-mjhm9" deleted
```

After the deletion, the Deployment controller automatically creates a replica.

2. Run the following command to view the created pod:

```
kubectl get pod | grep web-demo
```

The expected output is as follows, in which **web-demo-846b489584-d4d4j** is the newly created pod:

```
web-demo-846b489584-d4d4j 1/1 Running 0 110s
web-demo-846b489584-wvv5s 1/1 Running 0 7m50s
```

3. Run the following command to check whether the files in the **/data** path of the new pod have been modified:

```
kubectl exec web-demo-846b489584-d4d4j -- ls /data
```

Expected output:

```
static
```

The **static** file is retained, indicating that the data in the file system can be stored persistently.

**Step 5 Verify data sharing.**

1. Run the following command to view the created pod:

```
kubectl get pod | grep web-demo
```

Expected output:

```
web-demo-846b489584-d4d4j 1/1 Running 0 7m
web-demo-846b489584-wvv5s 1/1 Running 0 13m
```

2. Run the following command to create a file named **share** in the **/data** path of either pod: In this example, select the pod named **web-demo-846b489584-d4d4j**.

```
kubectl exec web-demo-846b489584-d4d4j -- touch /data/share
```

Check the files in the **/data** path of the pod.

```
kubectl exec web-demo-846b489584-d4d4j -- ls /data
```

Expected output:

```
share
static
```

3. Check whether the **share** file exists in the **/data** path of another pod (**web-demo-846b489584-wvv5s**) as well to verify data sharing.

```
kubectl exec web-demo-846b489584-wvv5s -- ls /data
```

Expected output:

```
share
static
```

After you create a file in the **/data** path of a pod, if the file is also created in the **/data** path of the other pod, the two pods share the same volume.

----End

## Related Operations

You can also perform the operations listed in [Table 8-34](#).

**Table 8-34** Related operations

Operation	Description	Procedure
Creating a storage volume (PV)	Create a PV on the CCE console.	<p>1. Choose <b>Storage</b> in the navigation pane. In the right pane, click the <b>PVs</b> tab. Click <b>Create PersistentVolume</b> in the upper right corner. In the dialog box displayed, configure parameters.</p> <ul style="list-style-type: none"> <li>• <b>Volume Type:</b> Select <b>OBS</b>.</li> <li>• <b>OBS:</b> Click <b>Select OBS</b>. On the displayed page, select the OBS volume that meets your requirements and click <b>OK</b>.</li> <li>• <b>PV Name:</b> Enter the PV name, which must be unique in a cluster.</li> <li>• <b>Access Mode:</b> SFS volumes support only <b>ReadWriteMany</b>, indicating that a storage volume can be mounted to multiple nodes in read/write mode. For details, see <a href="#">Volume Access Modes</a>.</li> <li>• <b>Reclaim Policy:</b> <b>Delete</b> or <b>Retain</b> is supported. For details, see <a href="#">PV Reclaim Policy</a>.</li> </ul> <p><b>NOTE</b> If multiple PVs use the same underlying storage volume, use <b>Retain</b> to prevent the underlying volume from being deleted with a PV.</p> <ul style="list-style-type: none"> <li>• <b>Access Key (AK/SK):</b> (Recommended) Customize a secret if you want to assign different user permissions to different OBS storage devices. For details, see <a href="#">Using a Custom Access Key (AK/SK) to Mount an OBS Volume</a>. Only secrets with the <b>secret.kubernetes.io/used-by = csi</b> label can be selected. The secret type is <b>cfe/secure-opaque</b>. If no secret is available, click <b>Create Secret</b> to create one.</li> <li>• <b>Mount Options:</b> Enter the mounting parameter key-value pairs. For details, see <a href="#">Configuring OBS Mount Options</a>.</li> </ul> <p>2. Click <b>Create</b>.</p>

Operation	Description	Procedure
Updating an access key	Update the access key of object storage on the CCE console.	<ol style="list-style-type: none"> <li>1. Choose <b>Storage</b> in the navigation pane. In the right pane, click the <b>PVCs</b> tab. Click <b>More</b> in the <b>Operation</b> column of the target PVC and select <b>Update Access Key</b>.</li> <li>2. Upload a key file in .csv format. For details, see <a href="#">Obtaining an Access Key</a>. Click <b>OK</b>.</li> </ol> <p><b>NOTE</b> After a global access key is updated, all pods mounted with the object storage that uses this access key can be accessed only after being restarted.</p>
Viewing events	View event names, event types, number of occurrences, Kubernetes events, first occurrence time, and last occurrence time of the PVC or PV.	<ol style="list-style-type: none"> <li>1. Choose <b>Storage</b> in the navigation pane. In the right pane, click the <b>PVCs</b> or <b>PVs</b> tab.</li> <li>2. Click <b>View Events</b> in the <b>Operation</b> column of the target PVC or PV to view events generated within one hour (events are retained for one hour).</li> </ol>
Viewing a YAML file	View, copy, or download the YAML file of a PVC or PV.	<ol style="list-style-type: none"> <li>1. Choose <b>Storage</b> in the navigation pane. In the right pane, click the <b>PVCs</b> or <b>PVs</b> tab.</li> <li>2. Click <b>View YAML</b> in the <b>Operation</b> column of the target PVC or PV to view or download the YAML.</li> </ol>

### 8.6.3 Using an OBS Bucket Through a Dynamic PV

This section describes how to automatically create an OBS bucket. It is applicable when no underlying storage volume is available.

#### Notes and Constraints

- If OBS volumes are used, the owner group and permission of the mount point cannot be modified.
- Every time an OBS volume is mounted to a workload through a PVC, a resident process is created in the backend. When a workload uses too many OBS volumes or reads and writes a large number of object storage files, resident processes will consume a significant amount of memory. To ensure stable running of the workload, make sure that the number of OBS volumes used does not exceed the requested memory. For example, if the workload requests for 4 GiB of memory, the number of OBS volumes should be **no more than 4**.
- Kata containers do not support OBS volumes.

- Hard links are not supported when common buckets are mounted.
- OBS allows a single user to create a maximum of 100 buckets. If a large number of dynamic PVCs are created, the number of buckets may exceed the upper limit, and no more OBS buckets can be created. In this case, use OBS by calling its API or SDK and do not mount OBS buckets to workloads.

## Automatically Creating an OBS Volume on the Console

**Step 1** Log in to the CCE console and click the cluster name to access the cluster console.

**Step 2** Dynamically create a PVC and PV.

1. Choose **Storage** in the navigation pane. In the right pane, click the **PVCs** tab. Click **Create PVC** in the upper right corner. In the dialog box displayed, configure PVC parameters.

Parameter	Description
PVC Type	In this example, select <b>OBS</b> .
PVC Name	Enter the PVC name, which must be unique in a namespace.
Creation Method	<ul style="list-style-type: none"> <li>– If no underlying storage is available, select <b>Dynamically provision</b> to create a PVC, PV, and underlying storage on the console in cascading mode.</li> <li>– If underlying storage is available, create a PV or use an existing PV to statically create a PVC. For details, see <a href="#">Using an Existing OBS Bucket Through a Static PV</a>.</li> </ul> In this example, select <b>Dynamically provision</b> .
Storage Classes	The default StorageClass for OBS volumes is <b>csi-obs</b> .
Instance Type	<ul style="list-style-type: none"> <li>– <b>Parallel file system</b>: a high-performance file system provided by OBS. It provides millisecond-level access latency, TB/s-level bandwidth, and million-level IOPS.</li> <li>– <b>Object bucket</b>: provides reliable, high-performance, secure, and budget-friendly storage for data. They have no restrictions on the quantity of files or storage capacity.</li> </ul>
OBS Class	You can select the following object bucket types: <ul style="list-style-type: none"> <li>– <b>Standard</b>: Applicable when a large number of hotspot files or small-sized files need to be accessed frequently (multiple times per month on average) and require fast access response.</li> <li>– <b>Infrequent access</b>: Applicable when data is not frequently accessed (fewer than 12 times per year on average) but requires fast access response.</li> </ul>
Access Mode	OBS volumes support only <b>ReadWriteMany</b> , indicating that a storage volume can be mounted to multiple nodes in read/write mode. For details, see <a href="#">Volume Access Modes</a> .

Parameter	Description
Access Key (AK/SK)	<p><b>Custom</b> (Recommended): Customize a secret if you want to assign different user permissions to different OBS storage devices. For details, see <a href="#">Using a Custom Access Key (AK/SK) to Mount an OBS Volume</a>.</p> <p>Only secrets with the <code>secret.kubernetes.io/used-by = csi</code> label can be selected. The secret type is <code>cfe/secure-opaque</code>. If no secret is available, click <b>Create Secret</b> to create one.</p> <ul style="list-style-type: none"> <li>- <b>Name</b>: Enter a secret name.</li> <li>- <b>Namespace</b>: Select the namespace where the secret is.</li> <li>- <b>Access Key (AK/SK)</b>: Upload a key file in .csv format. For details, see <a href="#">Obtaining an Access Key</a>.</li> </ul>
Enterprise Project	The default enterprise project, the enterprise project to which the cluster belongs, or the enterprise project specified by StorageClass is available.

2. Click **Create** to create a PVC and a PV.  
You can choose **Storage** in the navigation pane and view the created PVC and PV on the **PVCs** and **PVs** tab pages, respectively.

**Step 3** Create an application.

1. Choose **Workloads** in the navigation pane. In the right pane, click the **Deployments** tab.
2. Click **Create Workload** in the upper right corner. On the displayed page, click **Data Storage** in the **Container Settings** area and click **Add Volume** to select **PVC**.

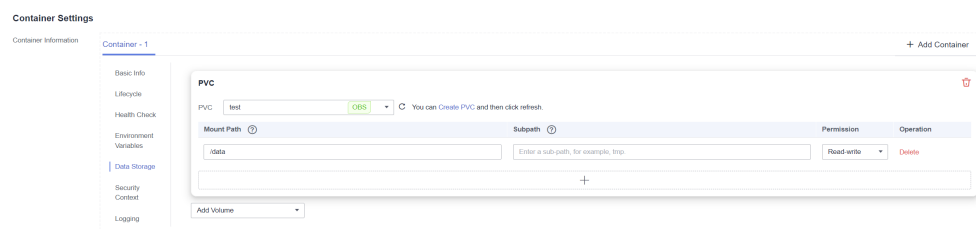
Mount and use storage volumes, as shown in [Table 8-35](#). For details about other parameters, see [Workloads](#).

**Table 8-35** Mounting a storage volume

Parameter	Description
PVC	Select an existing OBS volume.

Parameter	Description
Mount Path	<p>Enter a mount path, for example, <b>/tmp</b>.</p> <p>This parameter specifies a container path to which a data volume will be mounted. Do not mount the volume to a system directory such as <b>/</b> or <b>/var/run</b>. This may lead to container errors. Mount the volume to an empty directory. If the directory is not empty, ensure that there are no files that affect container startup. Otherwise, the files will be replaced, leading to container startup failures or workload creation failures.</p> <p><b>NOTICE</b></p> <p>If a volume is mounted to a high-risk directory, use an account with minimum permissions to start the container. Otherwise, high-risk files on the host may be damaged.</p>
Subpath	<p>Enter the subpath of the storage volume and mount a path in the storage volume to the container. In this way, different folders of the same storage volume can be used in a single pod. <b>tmp</b>, for example, indicates that data in the mount path of the container is stored in the <b>tmp</b> folder of the storage volume. If this parameter is left blank, the root path will be used by default.</p>
Permission	<ul style="list-style-type: none"> <li>– <b>Read-only:</b> You can only read the data in the mounted volumes.</li> <li>– <b>Read-write:</b> You can modify the data volumes mounted to the path. Newly written data will not be migrated if the container is migrated, which may cause data loss.</li> </ul>

In this example, the disk is mounted to the **/data** path of the container. The container data generated in this path is stored in the OBS volume.



3. After the configuration, click **Create Workload**.

After the workload is created, the data in the container mount directory will be persistently stored. Verify the storage by referring to [Verifying Data Persistence and Sharing](#).

----End

## Automatically Creating an OBS Volume Through kubectl

- Step 1** Use kubectl to access the cluster.



**Step 2** Use **StorageClass** to dynamically create a PVC and PV.

1. Create the **pvc-obs-auto.yaml** file.

```

apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: pvc-obs-auto
  namespace: default
  annotations:
    everest.io/obs-volume-type: STANDARD # OBS
    csi.storage.k8s.io/fstype: obsfs # Instance type
    csi.storage.k8s.io/node-publish-secret-name: <your_secret_name> # Custom secret name
    csi.storage.k8s.io/node-publish-secret-namespace: <your_namespace> # Namespace of the
    custom secret
    everest.io/enterprise-project-id: <your_project_id> # (Optional) Enterprise project ID. If an
    enterprise project is specified, use the same enterprise project when creating a PVC. Otherwise, the
    PVC cannot be bound to a PV.
spec:
  accessModes:
    - ReadWriteMany # The value must be ReadWriteMany for OBS.
  resources:
    requests:
      storage: 1Gi # OBS volume capacity
      storageClassName: csi-obs # StorageClass is OBS.
  
```

**Table 8-36** Key parameters

Parameter	Mandatory	Description
everest.io/obs-volume-type	Yes	OBS StorageClass. <ul style="list-style-type: none"> <li>- If <b>fsType</b> is set to <b>s3fs</b>, <b>STANDARD</b> (standard bucket) and <b>WARM</b> (infrequent access bucket) are supported.</li> <li>- This parameter is invalid when <b>fsType</b> is set to <b>obsfs</b>.</li> </ul>
csi.storage.k8s.io/fstype	Yes	Instance type. The value can be <b>obsfs</b> or <b>s3fs</b> . <ul style="list-style-type: none"> <li>- <b>obsfs</b>: a parallel file system</li> <li>- <b>s3fs</b>: object bucket</li> </ul>
csi.storage.k8s.io/node-publish-secret-name	No	Custom secret name. (Recommended) Select this option if you want to assign different user permissions to different OBS storage devices. For details, see <a href="#">Using a Custom Access Key (AK/SK) to Mount an OBS Volume</a> .
csi.storage.k8s.io/node-publish-secret-namespace	No	Namespace of a custom secret.

Parameter	Mandatory	Description
everest.io/enterprise-project-id	No	Project ID of OBS. To obtain an enterprise project ID, log in to the EPS console, click the name of the target enterprise project, and copy the enterprise project ID.
storage	Yes	Requested capacity in the PVC, in Gi. For OBS, this field is used only for verification (cannot be empty or 0). Its value is fixed at <b>1</b> , and any value you set does not take effect for OBS.
storageClassName	Yes	StorageClass name, which is <b>csi-obs</b> for an OBS volume.

- Run the following command to create a PVC:  

```
kubectl apply -f pvc-obs-auto.yaml
```

### Step 3 Create an application.

- Create a file named **web-demo.yaml**. In this example, the OBS volume is mounted to the **/data** path.

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: web-demo
  namespace: default
spec:
  replicas: 2
  selector:
    matchLabels:
      app: web-demo
  template:
    metadata:
      labels:
        app: web-demo
    spec:
      containers:
        - name: container-1
          image: nginx:latest
          volumeMounts:
            - name: pvc-obs-volume # Volume name, which must be the same as the volume name in the
volumes field
              mountPath: /data # Location where the storage volume is mounted
      imagePullSecrets:
        - name: default-secret
      volumes:
        - name: pvc-obs-volume # Volume name, which can be customized
          persistentVolumeClaim:
            claimName: pvc-obs-auto # Name of the created PVC
```

- Run the following command to create a workload to which the OBS volume is mounted:

```
kubectl apply -f web-demo.yaml
```

After the workload is created, you can try [Verifying Data Persistence and Sharing](#).

----End

## Verifying Data Persistence and Sharing

### Step 1 View the deployed application and files.

1. Run the following command to view the created pod:

```
kubectl get pod | grep web-demo
```

Expected output:

```
web-demo-846b489584-mjhm9 1/1 Running 0 46s
web-demo-846b489584-wvv5s 1/1 Running 0 46s
```

2. Run the following commands in sequence to view the files in the `/data` path of the pods:

```
kubectl exec web-demo-846b489584-mjhm9 -- ls /data
kubectl exec web-demo-846b489584-wvv5s -- ls /data
```

If no result is returned for both pods, no file exists in the `/data` path.

### Step 2 Run the following command to create a file named `static` in the `/data` path:

```
kubectl exec web-demo-846b489584-mjhm9 -- touch /data/static
```

### Step 3 Run the following command to check the files in the `/data` path:

```
kubectl exec web-demo-846b489584-mjhm9 -- ls /data
```

Expected output:

```
static
```

### Step 4 Verify data persistence.

1. Run the following command to delete the pod named **web-demo-846b489584-mjhm9**:

```
kubectl delete pod web-demo-846b489584-mjhm9
```

Expected output:

```
pod "web-demo-846b489584-mjhm9" deleted
```

After the deletion, the Deployment controller automatically creates a replica.

2. Run the following command to view the created pod:

```
kubectl get pod | grep web-demo
```

The expected output is as follows, in which **web-demo-846b489584-d4d4j** is the newly created pod:

```
web-demo-846b489584-d4d4j 1/1 Running 0 110s
web-demo-846b489584-wvv5s 1/1 Running 0 7m50s
```

3. Run the following command to check whether the files in the `/data` path of the new pod have been modified:

```
kubectl exec web-demo-846b489584-d4d4j -- ls /data
```

Expected output:

```
static
```

The **static** file is retained, indicating that the data in the file system can be stored persistently.

### Step 5 Verify data sharing.

1. Run the following command to view the created pod:

```
kubectl get pod | grep web-demo
```

Expected output:

```
web-demo-846b489584-d4d4j 1/1 Running 0 7m
web-demo-846b489584-wvv5s 1/1 Running 0 13m
```

2. Run the following command to create a file named **share** in the `/data` path of either pod: In this example, select the pod named **web-demo-846b489584-d4d4j**.

```
kubectl exec web-demo-846b489584-d4d4j -- touch /data/share
```

Check the files in the **/data** path of the pod.

```
kubectl exec web-demo-846b489584-d4d4j -- ls /data
```

Expected output:

```
share
static
```

3. Check whether the **share** file exists in the **/data** path of another pod (**web-demo-846b489584-wvv5s**) as well to verify data sharing.

```
kubectl exec web-demo-846b489584-wvv5s -- ls /data
```

Expected output:

```
share
static
```

After you create a file in the **/data** path of a pod, if the file is also created in the **/data** path of the other pod, the two pods share the same volume.

----End

## Related Operations

You can also perform the operations listed in [Table 8-37](#).

**Table 8-37** Related operations

Operation	Description	Procedure
Updating an access key	Update the access key of object storage on the CCE console.	<ol style="list-style-type: none"> <li>1. Choose <b>Storage</b> in the navigation pane. In the right pane, click the <b>PVCs</b> tab. Click <b>More</b> in the <b>Operation</b> column of the target PVC and select <b>Update Access Key</b>.</li> <li>2. Upload a key file in .csv format. For details, see <a href="#">Obtaining an Access Key</a>. Click <b>OK</b>.</li> </ol> <p><b>NOTE</b> After a global access key is updated, all pods mounted with the object storage that uses this access key can be accessed only after being restarted.</p>
Viewing events	View event names, event types, number of occurrences, Kubernetes events, first occurrence time, and last occurrence time of the PVC or PV.	<ol style="list-style-type: none"> <li>1. Choose <b>Storage</b> in the navigation pane. In the right pane, click the <b>PVCs</b> or <b>PVs</b> tab.</li> <li>2. Click <b>View Events</b> in the <b>Operation</b> column of the target PVC or PV to view events generated within one hour (events are retained for one hour).</li> </ol>
Viewing a YAML file	View, copy, or download the YAML file of a PVC or PV.	<ol style="list-style-type: none"> <li>1. Choose <b>Storage</b> in the navigation pane. In the right pane, click the <b>PVCs</b> or <b>PVs</b> tab.</li> <li>2. Click <b>View YAML</b> in the <b>Operation</b> column of the target PVC or PV to view or download the YAML.</li> </ol>

## 8.6.4 Configuring OBS Mount Options

This section describes how to configure OBS volume mount options. You can configure mount options in a PV and bind the PV to a PVC. Alternatively, configure mount options in a StorageClass and use the StorageClass to create a PVC. In this way, PVs can be dynamically created and inherit mount options configured in the StorageClass by default.

### Prerequisites

The **CCE Container Storage (Everest)** version must be **1.2.8 or later**. This add-on identifies the mount options and transfers them to the underlying storage resources. The parameter settings take effect only if the underlying storage resources support the specified options.

### OBS Mount Options

When mounting an OBS volume, the Everest add-on presets the options described in [Table 8-38](#) and [Table 8-39](#) by default. The options in [Table 8-38](#) are mandatory.

**Table 8-38** Mandatory mount options configured by default

Parameter	Value	Description
use_ino	Blank	If enabled, obsfs allocates the <b>inode</b> number. Enabled by default in read/write mode.
big_writes	Blank	If configured, the maximum size of the cache can be modified.
nonempty	Blank	Allows non-empty mount paths.
allow_other	Blank	Allows other users to access the parallel file system.
no_check_certificate	Blank	Disables server certificate verification.
enable_noobj_cache	Blank	Enables cache entries for objects that do not exist, which can improve performance. Enabled by default in object bucket read/write mode. <b>This option is no longer configured by default since Everest 1.2.40.</b>
sigv2	Blank	Specifies the signature version. Used by default in object buckets.
public_bucket	1	If this parameter is set to <b>1</b> , public buckets are mounted anonymously. Enabled by default in object bucket read-only mode.

**Table 8-39** Optional mount options configured by default

Parameter	Value	Description
max_write	131072	This parameter is valid only when <b>big_writes</b> is configured. The recommended value is <b>128 KB</b> .
ssl_verify_hostname	0	Disables SSL certificate verification based on the host name.
max_background	100	Allows setting the maximum number of waiting requests in the background. Used by default in parallel file systems.
umask	0	Mask of the configuration file permission. For example, if the umask value is <b>022</b> , the directory permission (the maximum permission is <b>777</b> ) is <b>755</b> ( $777 - 022 = 755$ , $rw\!-\!xr\!-\!x$ ).

## Configuring Mount Options in a PV

You can use the **mountOptions** field to configure mount options in a PV. The options you can configure in **mountOptions** are listed in [OBS Mount Options](#).

**Step 1** Use kubectl to access the cluster. For details, see [Connecting to a Cluster Using kubectl](#).

**Step 2** Configure mount options in a PV. Example:

```
apiVersion: v1
kind: PersistentVolume
metadata:
  annotations:
    pv.kubernetes.io/provisioned-by: everest-csi-provisioner
    everest.io/reclaim-policy: retain-volume-only # (Optional) The underlying volume is retained when
the PV is deleted.
  name: pv-obs # PV name
spec:
  accessModes:
    - ReadWriteMany # Access mode. The value must be ReadWriteMany for OBS.
  capacity:
    storage: 1Gi # OBS volume capacity
  csi:
    driver: obs.csi.everest.io # Dependent storage driver for the mounting
    fsType: obsfs # Instance type
    volumeHandle: <your_volume_id> # Name of the OBS volume
  volumeAttributes:
    storage.kubernetes.io/csiProvisionerIdentity: everest-csi-provisioner
    everest.io/obs-volume-type: STANDARD
    everest.io/region: <your_region> # Region where the OBS volume is
    everest.io/enterprise-project-id: <your_project_id> # (Optional) Enterprise project ID. If an enterprise
project is specified, use the same enterprise project when creating a PVC. Otherwise, the PVC cannot be
bound to a PV.
    nodePublishSecretRef: # Custom secret of the OBS volume
      name: <your_secret_name> # Custom secret name
      namespace: <your_namespace> # Namespace of the custom secret
    persistentVolumeReclaimPolicy: Retain # Reclaim policy
    storageClassName: csi-obs # StorageClass name
  mountOptions:
    - umask=027
```

**Step 3** After a PV is created, you can create a PVC and bind it to the PV, and then mount the PV to the container in the workload. For details, see [Using an Existing OBS Bucket Through a Static PV](#).

**Step 4** Check whether the mount options take effect.

In this example, the PVC is mounted to the workload that uses the **nginx:latest** image. You can log in to the node where the pod to which the OBS volume is mounted resides and view the progress details.

Run the following command:

- Object bucket: **ps -ef | grep s3fs**

```
root 22142 1 0 Jun03 ? 00:00:00 /usr/bin/s3fs {your_obs_name} /mnt/paas/kubernetes/kubelet/pods/{pod_uid}/volumes/kubernetes.io~csi/{your_pv_name}/mount -o url=https://{endpoint}:443 -o endpoint={region} -o passwd_file=/opt/everest-host-connector/obstmpcred/{your_obs_name} -o nonempty -o big_writes -o sigv2 -o allow_other -o no_check_certificate -o ssl_verify_hostname=0 -o umask=027 -o max_write=131072 -o multipart_size=20
```

- Parallel file system: **ps -ef | grep obsfs**

```
root 1355 1 0 Jun03 ? 00:03:16 /usr/bin/obsfs {your_obs_name} /mnt/paas/kubernetes/kubelet/pods/{pod_uid}/volumes/kubernetes.io~csi/{your_pv_name}/mount -o url=https://{endpoint}:443 -o endpoint={region} -o passwd_file=/opt/everest-host-connector/obstmpcred/{your_obs_name} -o allow_other -o nonempty -o big_writes -o use_ino -o no_check_certificate -o ssl_verify_hostname=0 -o max_background=100 -o umask=027 -o max_write=131072
```

----End

## Configuring Mount Options in a StorageClass

You can use the **mountOptions** field to configure mount options in a StorageClass. The options you can configure in **mountOptions** are listed in [OBS Mount Options](#).

**Step 1** Use kubectl to access the cluster. For details, see [Connecting to a Cluster Using kubectl](#).

**Step 2** Create a custom StorageClass. Example:

```
kind: StorageClass
apiVersion: storage.k8s.io/v1
metadata:
  name: csi-obs-mount-option
provisioner: everest-csi-provisioner
parameters:
  csi.storage.k8s.io/csi-driver-name: obs.csi.everest.io
  csi.storage.k8s.io/fstype: s3fs
  everest.io/obs-volume-type: STANDARD
reclaimPolicy: Delete
volumeBindingMode: Immediate
mountOptions: # Mount options
- umask=027
```

**Step 3** After the StorageClass is configured, you can use it to create a PVC. By default, the dynamically created PVs inherit the mount options configured in the StorageClass. For details, see [Using an OBS Bucket Through a Dynamic PV](#).

**Step 4** Check whether the mount options take effect.

In this example, the PVC is mounted to the workload that uses the **nginx:latest** image. You can log in to the node where the pod to which the OBS volume is mounted resides and view the progress details.

Run the following command:

- Object bucket: **ps -ef | grep s3fs**  

```
root 22142 1 0 Jun03 ? 00:00:00 /usr/bin/s3fs {your_obs_name} /mnt/paas/kubernetes/kubelet/pods/{pod_uid}/volumes/kubernetes.io~csi/{your_pv_name}/mount -o url=https://{endpoint}:443 -o endpoint={region} -o passwd_file=/opt/everest-host-connector/obstmpcred/{your_obs_name} -o nonempty -o big_writes -o sigv2 -o allow_other -o no_check_certificate -o ssl_verify_hostname=0 -o umask=027 -o max_write=131072 -o multipart_size=20
```
  - Parallel file system: **ps -ef | grep obsfs**  

```
root 1355 1 0 Jun03 ? 00:03:16 /usr/bin/obsfs {your_obs_name} /mnt/paas/kubernetes/kubelet/pods/{pod_uid}/volumes/kubernetes.io~csi/{your_pv_name}/mount -o url=https://{endpoint}:443 -o endpoint={region} -o passwd_file=/opt/everest-host-connector/obstmpcred/{your_obs_name} -o allow_other -o nonempty -o big_writes -o use_ino -o no_check_certificate -o ssl_verify_hostname=0 -o max_background=100 -o umask=027 -o max_write=131072
```
- End

## 8.6.5 Using a Custom Access Key (AK/SK) to Mount an OBS Volume

### Scenario

**CCE Container Storage (Everest)** of v1.2.8 or later supports custom access keys. In this way, IAM users can use their own custom access keys to mount an OBS volume.

### Prerequisites

- The **CCE Container Storage (Everest)** version must be 1.2.8 or later.
- The cluster version must be 1.15.11 or later.

### Notes and Constraints

- When an OBS volume is mounted using a custom access key (AK/SK), the access key cannot be deleted or disabled. Otherwise, the service container cannot access the mounted OBS volume.

### Disabling Auto Key Mounting

On the earlier version's console, you need to upload the AK/SK, which is then used by default for mounting an OBS volume. As a result, all IAM users within your account will use the same key to mount OBS buckets, and they will have identical permissions on the buckets. However, this setting does not allow you to set different permissions for individual IAM users.

If you have uploaded the AK/SK, disable the automatic mounting of access keys by enabling the **disable\_auto\_mount\_secret** parameter in the Everest add-on to prevent IAM users from performing unauthorized operations. In this way, the access keys uploaded on the console will not be used when creating OBS volumes.

#### NOTE

- When enabling **disable-auto-mount-secret**, ensure that no OBS volume exists in the cluster. A workload mounted with an OBS volume, when scaled or restarted, will fail to remount the OBS volume because it needs to specify the access key but is prohibited by **disable-auto-mount-secret**.
- If **disable-auto-mount-secret** is set to **true**, an access key must be specified when a PV or PVC is created. Otherwise, the OBS volume fails to be mounted.



### kubectl edit ds everest-csi-driver -nkube-system

Search for **disable-auto-mount-secret** and set it to **true**.

```

- /bin/sh
- c
- /var/paas/everest-csi-driver/everest-csi-driver --call-mode=kubelet --drivers=*,local.csi.everest.io
--aksk-secret-name=paas.aks --iam-endpoint=https://iam. :443 --evs-endpoint=https://evs. :443
--ecs-endpoint=https://ecs. :443 --sfs-endpoint=https://sfs. :443
--obs-endpoint=https://obs. :443 --sfsturbo-endpoint=https://sfs-turbo. :443
--bms-endpoint=https://bms. :443 --ims-endpoint=https://ims. :443
--feature-gates=supportHcs=false --project-id=b6315dd3d0ff4be5b31a963256794989
--cluster-id=827dced9-c2ad-11e6-bfce-0255ac1036e0 --default-vpc-id=0f090290-2b77-48ae-a601-0e746f350265
--disable-auto-mount-secret=true --cluster-version=v1.19.10-r0 --v=2 1>/var/paas/sys/log/everest-csi-driver/everest-csi-driver-standalone.log
2>31
env:

```

Run **:wq** to save the settings and exit. Wait until the pod is restarted.

## Obtaining an Access Key

- Step 1** Log in to the console.
- Step 2** Hover the cursor over the username in the upper right corner and choose **My Credentials** from the drop-down list.
- Step 3** In the navigation pane, choose **Access Keys**.
- Step 4** Click **Create Access Key**. The **Create Access Key** dialog box is displayed.
- Step 5** Click **OK** to download the access key.

----End

## Creating a Secret Using an Access Key

- Step 1** Obtain an access key.
- Step 2** Encode the keys using Base64. (Assume that the AK is xxx and the SK is yyy.)

```
echo -n xxx|base64
```

```
echo -n yyy|base64
```

Record the encoded AK and SK.

- Step 3** Create a YAML file for the secret, for example, **test-user.yaml**.

```

apiVersion: v1
data:
  access.key: WE5WWVhVNU*****
  secret.key: Nnk4emJyZ0*****
kind: Secret
metadata:
  name: test-user
  namespace: default
  labels:
    secret.kubernetes.io/used-by: csi
type: cfe/secure-opaque

```

Specifically:

Parameter	Description
access.key	Base64-encoded AK.
secret.key	Base64-encoded SK.

Parameter	Description
name	Secret name.
namespace	Namespace of the secret.
secret.kubernetes.io/used-by: csi	Add this label in the YAML file if you want to make it available on the CCE console when you create an OBS PV/PVC.
type	Secret type. The value must be <b>cfe/secure-opaque</b> . When this type is used, the data entered by users is automatically encrypted.

**Step 4** Create the secret.

```
kubectl create -f test-user.yaml
```

----End

## Mounting a Secret When Statically Creating an OBS Volume

After a secret is created using the AK/SK, you can associate the secret with the PV to be created and then use the AK/SK in the secret to mount an OBS volume.

**Step 1** Log in to the OBS console, create an OBS bucket, and record the bucket name and storage class. The parallel file system is used as an example.

**Step 2** Create a YAML file for the PV, for example, **pv-example.yaml**.

```
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:
    nodePublishSecretRef:
      name: test-user
      namespace: default
    driver: obs.csi.everest.io
    fsType: obsfs
    volumeAttributes:
      everest.io/obs-volume-type: STANDARD
      everest.io/region: eu-west-101
      storage.kubernetes.io/csiProvisionerIdentity: everest-csi-provisioner
    volumeHandle: obs-normal-static-pv
  persistentVolumeReclaimPolicy: Delete
  storageClassName: csi-obs
```

Parameter	Description
nodePublishSecretRef	Secret specified during the mounting. <ul style="list-style-type: none"> <li>• <b>name</b>: name of the secret</li> <li>• <b>namespace</b>: namespace of the secret</li> </ul>
fsType	File type, which can be <b>s3fs</b> or <b>obsfs</b> . If the value is <b>s3fs</b> , an OBS bucket is created. If the value is <b>obsfs</b> , an OBS parallel file system is created.
volumeHandle	OBS bucket name.

**Step 3** Create a PV.

**kubectl create -f pv-example.yaml**

After a PV is created, you can create a PVC and associate it with the PV.

**Step 4** Create a YAML file for the PVC, for example, **pvc-example.yaml**.

**Example YAML file for the PVC:**

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  annotations:
    csi.storage.k8s.io/node-publish-secret-name: test-user
    csi.storage.k8s.io/node-publish-secret-namespace: default
    volume.beta.kubernetes.io/storage-provisioner: everest-csi-provisioner
    everest.io/obs-volume-type: STANDARD
    csi.storage.k8s.io/fstype: obsfs
  name: obs-secret
  namespace: default
spec:
  accessModes:
  - ReadWriteMany
  resources:
    requests:
      storage: 1Gi
  storageClassName: csi-obs
  volumeName: pv-obs-example
```

Parameter	Description
csi.storage.k8s.io/node-publish-secret-name	Name of the secret
csi.storage.k8s.io/node-publish-secret-namespace	Namespace of the secret

**Step 5** Create a PVC.

**kubectl create -f pvc-example.yaml**

After the PVC is created, you can create a workload and associate it with the PVC to create volumes.

----End

## Mounting a Secret When Dynamically Creating an OBS Volume

When dynamically creating an OBS volume, you can use the following method to specify a secret:

### Step 1 Create a YAML file for the PVC, for example, **pvc-example.yaml**.

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  annotations:
    csi.storage.k8s.io/node-publish-secret-name: test-user
    csi.storage.k8s.io/node-publish-secret-namespace: default
    everest.io/obs-volume-type: STANDARD
    csi.storage.k8s.io/fstype: obsfs
  name: obs-secret
  namespace: default
spec:
  accessModes:
  - ReadWriteMany
  resources:
    requests:
      storage: 1Gi
  storageClassName: csi-obs
```

Parameter	Description
csi.storage.k8s.io/node-publish-secret-name	Name of the secret
csi.storage.k8s.io/node-publish-secret-namespace	Namespace of the secret

### Step 2 Create a PVC.

```
kubectl create -f pvc-example.yaml
```

After the PVC is created, you can create a workload and associate it with the PVC to create volumes.

----End

## Verification

You can use a secret of an IAM user to mount an OBS volume. Assume that a workload named **obs-secret** is created, the mount path in the container is **/temp**, and the IAM user has the CCE **ReadOnlyAccess** and **Tenant Guest** permissions.

1. Query the name of the workload pod.

```
kubectl get po | grep obs-secret
```

Expected outputs:

```
obs-secret-5cd558f76f-vxslv    1/1    Running    0    3m22s
```

2. Query the objects in the mount path. In this example, the query is successful.

```
kubectl exec obs-secret-5cd558f76f-vxslv -- ls -l /temp/
```

3. Write data into the mount path. In this example, the write operation failed.

```
kubectl exec obs-secret-5cd558f76f-vxslv -- touch /temp/test
```

Expected outputs:

```
touch: setting times of '/temp/test': No such file or directory  
command terminated with exit code 1
```

4. Set the read/write permissions for the IAM user who mounted the OBS volume by referring to the bucket policy configuration.
5. Write data into the mount path again. In this example, the write operation succeeded.

```
kubectl exec obs-secret-5cd558f76f-vxslv -- touch /temp/test
```

6. Check the mount path in the container to see whether the data is successfully written.

```
kubectl exec obs-secret-5cd558f76f-vxslv -- ls -l /temp/
```

Expected outputs:

```
-rwxrwxrwx 1 root root 0 Jun 7 01:52 test
```

## 8.7 Local PVs

### 8.7.1 Overview

#### Introduction

CCE allows you to use LVM to combine data volumes on nodes into a storage pool (VolumeGroup) and create LVs for containers to mount. A PV that uses a local persistent volume as the medium is considered local PV.

Compared with the HostPath volume, the local PV can be used in a persistent and portable manner. In addition, the PV of the local PV has the node affinity configuration. The pod mounted to the local PV is automatically scheduled based on the affinity configuration. You do not need to manually schedule the pod to a specific node.

#### Mount Modes

Local PVs can be mounted only in the following modes:

- **Using a Local PV Through a Dynamic PV:** dynamic creation mode, in which you specify a StorageClass when creating a PVC. Then, an OBS volume and PV will be created automatically.
- **Dynamically Mounting a Local PV to a StatefulSet:** available only for StatefulSets. In this mode, each pod is associated with a unique PVC and PV. After a pod is rescheduled, the original data can still be mounted to it based on the PVC name. This mode applies to StatefulSets with multiple pods.

#### NOTE

Local PVs cannot be used through static PVs. That is, local PVs cannot be manually created and then mounted to workloads through PVCs.

#### Notes and Constraints

- Local PVs are supported only when the cluster version is v1.21.2-r0 or later and the Everest add-on version is 2.1.23 or later. Version 2.1.23 or later is recommended.

- **Deleting, resetting, or scaling** a node will cause the loss of the PVC/PV data of the local PV associated with the node. The lost data cannot be restored, and the affected PVC/PV cannot be used again. In these scenarios, the pod that uses the local PV is evicted from the node. A new pod will be created and stay in the pending state. This is because the PVC used by the pod has a node label, due to which the pod cannot be scheduled. After the node is reset, the pod may be scheduled to the reset node. In this case, the pod remains in the creating state because the underlying logical volume corresponding to the PVC does not exist.
- Do not manually delete the corresponding storage pool or detach data disks from the node. Otherwise, exceptions such as data loss may occur.
- Local PVs are in non-shared mode and cannot be mounted to multiple workloads or tasks concurrently. Additionally, local PVs cannot be mounted to multiple pods of a workload concurrently.

## 8.7.2 Importing a PV to a Storage Pool

CCE allows you to use LVM to combine data volumes on nodes into a storage pool (VolumeGroup) and create LVs for containers to mount. Before creating a local PV, import the data disk of the node to the storage pool.

### Notes and Constraints

- Local PVs are supported only when the cluster version is v1.21.2-r0 or later and the Everest add-on version is 2.1.23 or later. Version 2.1.23 or later is recommended.
- The first data disk (used by container runtime and the kubelet component) on a node cannot be imported as a storage pool.
- Storage pools in striped mode do not support scale-out. After scale-out, fragmented space may be generated and the storage pool cannot be used.
- Storage pools cannot be scaled in or deleted.
- If disks in a storage pool on a node are deleted, the storage pool will malfunction.

### Importing a Storage Pool

#### Imported during node creation

When creating a node, you can add a data disk to the node in **Storage Settings** and import the data disk to the storage pool as a PV. For details, see [Creating a Node](#).

#### Imported manually

If no PV is imported during node creation, or the capacity of the current PV is insufficient, you can manually import a PV.

- Step 1** Go to the ECS console and add a SCSI disk to the node. For details, see [Adding a Disk](#).
- Step 2** Log in to the CCE console and click the cluster name to access the cluster console.
- Step 3** Choose **Storage** in the navigation pane. In the right pane, click the **Storage Pool** tab.

- Step 4** View the node to which the disk has been added and select **Import as PV**. You can select a write mode during the import.

 **NOTE**

If the manually attached disk is not displayed in the storage pool, wait for 1 minute and refresh the list.

- **Linear:** A linear logical volume integrates one or more physical volumes. Data is written to the next physical volume when the previous one is used up.
- **Striped:** A striped logical volume stripes data into blocks of the same size and stores them in multiple physical volumes in sequence. This allows data to be concurrently read and written. Select this option only when there are multiple volumes.

----End

## Expanding a Storage Pool

The capacity of a storage pool can be expanded in either of the following ways:

1. Add more large-capacity disks to the pool using the preceding manual import method.
2. Increase the size of the existing disks on the ECS console. Then, the storage pool capacity will be automatically expanded.

Either of these methods will work.

## 8.7.3 Using a Local PV Through a Dynamic PV

### Prerequisites

- You have created a cluster and installed the CSI add-on ([Everest](#)) in the cluster.
- To create a cluster using commands, ensure kubectl is used. For details, see [Connecting to a Cluster Using kubectl](#).
- You have imported a data disk of a node to the local PV storage pool. For details, see [Importing a PV to a Storage Pool](#).

### Notes and Constraints

- Local PVs are supported only when the cluster version is v1.21.2-r0 or later and the Everest add-on version is 2.1.23 or later. Version 2.1.23 or later is recommended.
- **Deleting, resetting, or scaling** a node will cause the loss of the PVC/PV data of the local PV associated with the node. The lost data cannot be restored, and the affected PVC/PV cannot be used again. In these scenarios, the pod that uses the local PV is evicted from the node. A new pod will be created and stay in the pending state. This is because the PVC used by the pod has a node label, due to which the pod cannot be scheduled. After the node is reset, the pod may be scheduled to the reset node. In this case, the pod remains in the creating state because the underlying logical volume corresponding to the PVC does not exist.
- Do not manually delete the corresponding storage pool or detach data disks from the node. Otherwise, exceptions such as data loss may occur.

- Local PVs are in non-shared mode and cannot be mounted to multiple workloads or tasks concurrently. Additionally, local PVs cannot be mounted to multiple pods of a workload concurrently.

## Automatically Creating a Local PV on the Console

**Step 1** Log in to the CCE console and click the cluster name to access the cluster console.

**Step 2** Dynamically create a PVC and PV.

- Choose **Storage** in the navigation pane. In the right pane, click the **PVCs** tab. Click **Create PVC** in the upper right corner. In the dialog box displayed, configure PVC parameters.

Parameter	Description
PVC Type	In this section, select <b>Local PV</b> .
PVC Name	Enter the PVC name, which must be unique in a namespace.
Creation Method	You can only select <b>Dynamically provision</b> to create a PVC, PV, and underlying storage on the console in cascading mode.
Storage Classes	The default StorageClass of local PVs is <b>csi-local-topology</b> .
Capacity	Requested capacity of the storage volume in GiB or MiB. <b>NOTE</b> To implement local PVs, Logical Volume Manager (LVM) is used with a local extent (LE) of 4 MiB. If the requested PVC capacity is not a multiple of 4 MiB, the LVM logical volume's actual capacity will be rounded up to the nearest multiple of 4 MiB. For example, if you request 401 MiB of PVC capacity, the LVM logical volume's actual capacity will be 404 MiB (101 LEs), which is the same as what is displayed on the page.
Access Mode	Local PVs support only <b>ReadWriteOnce</b> , indicating that a PV can be mounted to one node in read/write mode. For details, see <a href="#">Volume Access Modes</a> .
Storage Pool	View the imported storage pool. For details about how to import a new data volume to the storage pool, see <a href="#">Importing a PV to a Storage Pool</a> .

- Click **Create** to create a PVC and a PV.

You can choose **Storage** in the navigation pane and view the created PVC and PV on the **PVCs** and **PVs** tab pages, respectively.

### NOTE

The volume binding mode of the local storage class (named **csi-local-topology**) is late binding (that is, the value of **volumeBindingMode** is **WaitForFirstConsumer**). In this mode, PV creation and binding are delayed. The corresponding PV is created and bound only when the PVC is used during workload creation.



**Step 3** Create an application.

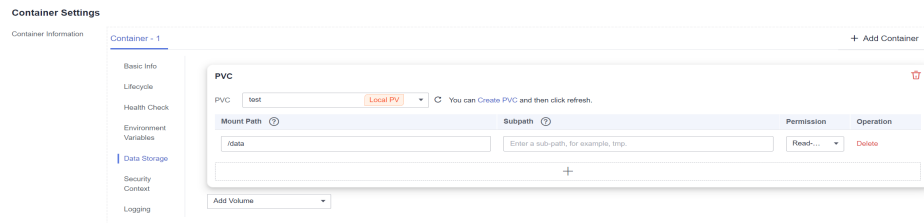
1. Choose **Workloads** in the navigation pane. In the right pane, click the **Deployments** tab.
2. Click **Create Workload** in the upper right corner. On the displayed page, click **Data Storage** in the **Container Settings** area and click **Add Volume** to select **PVC**.

Mount and use storage volumes, as shown in [Table 8-40](#). For details about other parameters, see [Workloads](#).

**Table 8-40** Mounting a storage volume

Parameter	Description
PVC	Select an existing local PV. A local PV can be mounted to only one workload.
Mount Path	Enter a mount path, for example, <b>/tmp</b> . This parameter specifies a container path to which a data volume will be mounted. Do not mount the volume to a system directory such as <b>/</b> or <b>/var/run</b> . This may lead to container errors. Mount the volume to an empty directory. If the directory is not empty, ensure that there are no files that affect container startup. Otherwise, the files will be replaced, leading to container startup failures or workload creation failures. <b>NOTICE</b> If a volume is mounted to a high-risk directory, use an account with minimum permissions to start the container. Otherwise, high-risk files on the host may be damaged.
Subpath	Enter the subpath of the storage volume and mount a path in the storage volume to the container. In this way, different folders of the same storage volume can be used in a single pod. <b>tmp</b> , for example, indicates that data in the mount path of the container is stored in the <b>tmp</b> folder of the storage volume. If this parameter is left blank, the root path will be used by default.
Permission	<ul style="list-style-type: none"> <li>- <b>Read-only</b>: You can only read the data in the mounted volumes.</li> <li>- <b>Read-write</b>: You can modify the data volumes mounted to the path. Newly written data will not be migrated if the container is migrated, which may cause data loss.</li> </ul>

In this example, the disk is mounted to the **/data** path of the container. The container data generated in this path is stored in the local PV.



3. After the configuration, click **Create Workload**.

After the workload is created, the data in the container mount directory will be persistently stored. Verify the storage by referring to [Verifying Data Persistence](#).

----End

## Automatically Creating a Local PV Through kubectl

**Step 1** Use kubectl to access the cluster.

**Step 2** Use **StorageClass** to dynamically create a PVC and PV.

1. Create the **pvc-local.yaml** file.

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: pvc-local
  namespace: default

spec:
  accessModes:
    - ReadWriteOnce          # The value must be ReadWriteOnce for local PVs.
  resources:
    requests:
      storage: 10Gi        # Local PV capacity
      storageClassName: csi-local-topology # StorageClass is local PV.
```

**Table 8-41** Key parameters

Parameter	Mandatory	Description
storage	Yes	Requested PVC capacity in Gi or Mi. <b>NOTE</b> To implement local PVs, Logical Volume Manager (LVM) is used with a local extent (LE) of 4 MiB. If the requested PVC capacity is not a multiple of 4 MiB, the LVM logical volume's actual capacity will be rounded up to the nearest multiple of 4 MiB. For example, if you request 401 MiB of PVC capacity, the LVM logical volume's actual capacity will be 404 MiB (101 LEs), which is the same as what is displayed on the page.
storageClassName	Yes	StorageClass name, which is <b>csi-local-topology</b> for a local PV.

2. Run the following command to create a PVC:

```
kubectl apply -f pvc-local.yaml
```

**Step 3** Create an application.

1. Create a file named **web-local.yaml**. In this example, the local PV is mounted to the **/data** path.

```
apiVersion: apps/v1
kind: StatefulSet
metadata:
  name: web-local
  namespace: default
spec:
  replicas: 1
  selector:
    matchLabels:
      app: web-local
  serviceName: web-local # Headless Service name
  template:
    metadata:
      labels:
        app: web-local
    spec:
      containers:
        - name: container-1
          image: nginx:latest
          volumeMounts:
            - name: pvc-disk # Volume name, which must be the same as the volume name in the
              volumes field.
              mountPath: /data # Location where the storage volume is mounted
      imagePullSecrets:
        - name: default-secret
      volumes:
        - name: pvc-disk # Volume name, which can be customized
          persistentVolumeClaim:
            claimName: pvc-local # Name of the created PVC
---
apiVersion: v1
kind: Service
metadata:
  name: web-local # Headless Service name
  namespace: default
  labels:
    app: web-local
spec:
  selector:
    app: web-local
  clusterIP: None
  ports:
    - name: web-local
      targetPort: 80
      nodePort: 0
      port: 80
      protocol: TCP
  type: ClusterIP
```

2. Run the following command to create a workload to which the local PV is mounted:

```
kubectl apply -f web-local.yaml
```

After the workload is created, the data in the container mount directory will be persistently stored. Verify the storage by referring to [Verifying Data Persistence](#).

----End

## Verifying Data Persistence

- Step 1** View the deployed application and local files.

1. Run the following command to view the created pod:

```
kubectl get pod | grep web-local
```

Expected output:

```
web-local-0      1/1      Running    0          38s
```

2. Run the following command to check whether the local PV has been mounted to the **/data** path:

```
kubectl exec web-local-0 -- df | grep data
```

Expected output:

```
/dev/mapper/vg--everest--localvolume--persistent-pvc-local 10255636 36888 10202364  
0% /data
```

3. Run the following command to check the files in the **/data** path:

```
kubectl exec web-local-0 -- ls /data
```

Expected output:

```
lost+found
```

- Step 2** Run the following command to create a file named **static** in the **/data** path:

```
kubectl exec web-local-0 -- touch /data/static
```

- Step 3** Run the following command to check the files in the **/data** path:

```
kubectl exec web-local-0 -- ls /data
```

Expected output:

```
lost+found  
static
```

- Step 4** Run the following command to delete the pod named **web-local-0**:

```
kubectl delete pod web-local-0
```

Expected output:

```
pod "web-local-0" deleted
```

- Step 5** After the deletion, the StatefulSet controller automatically creates a replica with the same name. Run the following command to check whether the files in the **/data** path have been modified:

```
kubectl exec web-local-0 -- ls /data
```

Expected output:

```
lost+found  
static
```

The **static** file is retained, indicating that the data in the local PV can be stored persistently.

----End

## Related Operations

You can also perform the operations listed in [Table 8-42](#).

**Table 8-42** Related operations

Operation	Description	Procedure
Viewing events	View event names, event types, number of occurrences, Kubernetes events, first occurrence time, and last occurrence time of the PVC or PV.	<ol style="list-style-type: none"> <li>1. Choose <b>Storage</b> in the navigation pane. In the right pane, click the <b>PVCs</b> or <b>PVs</b> tab.</li> <li>2. Click <b>View Events</b> in the <b>Operation</b> column of the target PVC or PV to view events generated within one hour (events are retained for one hour).</li> </ol>
Viewing a YAML file	View, copy, or download the YAML file of a PVC or PV.	<ol style="list-style-type: none"> <li>1. Choose <b>Storage</b> in the navigation pane. In the right pane, click the <b>PVCs</b> or <b>PVs</b> tab.</li> <li>2. Click <b>View YAML</b> in the <b>Operation</b> column of the target PVC or PV to view or download the YAML.</li> </ol>

## 8.7.4 Dynamically Mounting a Local PV to a StatefulSet

### Application Scenarios

Dynamic mounting is available only for creating a **StatefulSet**. It is implemented through a volume claim template (**volumeClaimTemplates** field) and depends on dynamic creation of PVs through StorageClass. In this mode, each pod in a multi-pod StatefulSet is associated with a unique PVC and PV. After a pod is rescheduled, the original data can still be mounted to it based on the PVC name. In the common mounting mode for a Deployment, if ReadWriteMany is supported, multiple pods of the Deployment will be mounted to the same underlying storage.

#### NOTE

When updating a StatefulSet in Kubernetes, it is not allowed to add or delete the **volumeClaimTemplates** field. This field can only be configured during the creation of the StatefulSet.

### Prerequisites

- You have created a cluster and installed the CSI add-on (**Everest**) in the cluster.
- To create a cluster using commands, ensure kubectl is used. For details, see [Connecting to a Cluster Using kubectl](#).
- You have imported a data disk of a node to the local PV storage pool. For details, see [Importing a PV to a Storage Pool](#).

### Dynamically Mounting a Local PV on the Console

**Step 1** Log in to the CCE console and click the cluster name to access the cluster console.

- Step 2** Choose **Workloads** in the navigation pane. In the right pane, click the **StatefulSets** tab.
- Step 3** Click **Create Workload** in the upper right corner. On the displayed page, click **Data Storage** in the **Container Settings** area and click **Add Volume** to select **VolumeClaimTemplate**.
- Step 4** Click **Create PVC**. In the dialog box displayed, configure the volume claim template parameters.

Click **Create**.

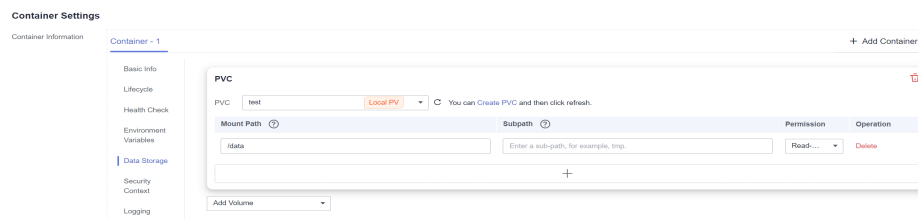
Parameter	Description
PVC Type	In this section, select <b>Local PV</b> .
PVC Name	Enter the name of the PVC. After a PVC is created, a suffix is automatically added based on the number of pods. The format is <i>&lt;Custom PVC name&gt;-&lt;Serial number&gt;</i> , for example, example-0.
Creation Method	You can only select <b>Dynamically provision</b> to create a PVC, PV, and underlying storage on the console in cascading mode.
Storage Classes	The default StorageClass of local PVs is <b>csi-local-topology</b> .
Capacity	Requested capacity of the storage volume in GiB or MiB. <b>NOTE</b> To implement local PVs, Logical Volume Manager (LVM) is used with a local extent (LE) of 4 MiB. If the requested PVC capacity is not a multiple of 4 MiB, the LVM logical volume's actual capacity will be rounded up to the nearest multiple of 4 MiB. For example, if you request 401 MiB of PVC capacity, the LVM logical volume's actual capacity will be 404 MiB (101 LEs), which is the same as what is displayed on the page.
Access Mode	Local PVs support only <b>ReadWriteOnce</b> , indicating that a PV can be mounted to one node in read/write mode. For details, see <a href="#">Volume Access Modes</a> .
Storage Pool	View the imported storage pool. For details about how to import a new data volume to the storage pool, see <a href="#">Importing a PV to a Storage Pool</a> .

- Step 5** Enter the path to which the volume is mounted.

**Table 8-43** Mounting a storage volume

Parameter	Description
Mount Path	<p>Enter a mount path, for example, <b>/tmp</b>.</p> <p>This parameter specifies a container path to which a data volume will be mounted. Do not mount the volume to a system directory such as <b>/</b> or <b>/var/run</b>. This may lead to container errors. Mount the volume to an empty directory. If the directory is not empty, ensure that there are no files that affect container startup. Otherwise, the files will be replaced, leading to container startup failures or workload creation failures.</p> <p><b>NOTICE</b> If a volume is mounted to a high-risk directory, use an account with minimum permissions to start the container. Otherwise, high-risk files on the host may be damaged.</p>
Subpath	<p>Enter the subpath of the storage volume and mount a path in the storage volume to the container. In this way, different folders of the same storage volume can be used in a single pod. <b>tmp</b>, for example, indicates that data in the mount path of the container is stored in the <b>tmp</b> folder of the storage volume. If this parameter is left blank, the root path will be used by default.</p>
Permission	<ul style="list-style-type: none"> <li>● <b>Read-only:</b> You can only read the data in the mounted volumes.</li> <li>● <b>Read-write:</b> You can modify the data volumes mounted to the path. Newly written data will not be migrated if the container is migrated, which may cause data loss.</li> </ul>

In this example, the disk is mounted to the **/data** path of the container. The container data generated in this path is stored in the local PV.



**Step 6** Dynamically mount and use storage volumes. For details about other parameters, see [Creating a StatefulSet](#). After the configuration, click **Create Workload**.

After the workload is created, the data in the container mount directory will be persistently stored. Verify the storage by referring to [Verifying Data Persistence](#).

----End

## Dynamically Mounting a Local PV Through kubectl

**Step 1** Use kubectl to access the cluster.

**Step 2** Create a file named **statefulset-local.yaml**. In this example, the local PV is mounted to the **/data** path.

```
apiVersion: apps/v1
kind: StatefulSet
metadata:
  name: statefulset-local
  namespace: default
spec:
  selector:
    matchLabels:
      app: statefulset-local
  template:
    metadata:
      labels:
        app: statefulset-local
    spec:
      containers:
        - name: container-1
          image: nginx:latest
          volumeMounts:
            - name: pvc-local          # The value must be the same as that in the volumeClaimTemplates field.
              mountPath: /data        # Location where the storage volume is mounted
          imagePullSecrets:
            - name: default-secret
      serviceName: statefulset-local  # Headless Service name
      replicas: 2
      volumeClaimTemplates:
        - apiVersion: v1
          kind: PersistentVolumeClaim
          metadata:
            name: pvc-local
            namespace: default

      spec:
        accessModes:
          - ReadWriteOnce          # The value must be ReadWriteOnce for local PVs.
        resources:
          requests:
            storage: 10Gi        # Storage volume capacity
          storageClassName: csi-local-topology  # StorageClass is local PV.
---
apiVersion: v1
kind: Service
metadata:
  name: statefulset-local  # Headless Service name
  namespace: default
  labels:
    app: statefulset-local
spec:
  selector:
    app: statefulset-local
  clusterIP: None
  ports:
    - name: statefulset-local
      targetPort: 80
      nodePort: 0
      port: 80
      protocol: TCP
  type: ClusterIP
```



**Table 8-44** Key parameters

Parameter	Mandatory	Description
storage	Yes	Requested PVC capacity in Gi or Mi. <b>NOTE</b> To implement local PVs, Logical Volume Manager (LVM) is used with a local extent (LE) of 4 MiB. If the requested PVC capacity is not a multiple of 4 MiB, the LVM logical volume's actual capacity will be rounded up to the nearest multiple of 4 MiB. For example, if you request 401 MiB of PVC capacity, the LVM logical volume's actual capacity will be 404 MiB (101 LEs), which is the same as what is displayed on the page.
storageClassName	Yes	The StorageClass of local PVs is <b>csi-local-topology</b> .

**Step 3** Run the following command to create a workload to which the local PV is mounted:

```
kubectl apply -f statefulset-local.yaml
```

After the workload is created, you can try [Verifying Data Persistence](#).

----End

## Verifying Data Persistence

**Step 1** View the deployed application and files.

1. Run the following command to view the created pod:

```
kubectl get pod | grep statefulset-local
```

Expected output:

```
statefulset-local-0    1/1    Running    0    45s
statefulset-local-1    1/1    Running    0    28s
```

2. Run the following command to check whether the local PV has been mounted to the **/data** path:

```
kubectl exec statefulset-local-0 -- df | grep data
```

Expected output:

```
/dev/mapper/vg--everest--localvolume--persistent-pvc-local    10255636    36888    10202364
0% /data
```

3. Run the following command to check the files in the **/data** path:

```
kubectl exec statefulset-local-0 -- ls /data
```

Expected output:

```
lost+found
```

**Step 2** Run the following command to create a file named **static** in the **/data** path:

```
kubectl exec statefulset-local-0 -- touch /data/static
```

**Step 3** Run the following command to check the files in the **/data** path:

```
kubectl exec statefulset-local-0 -- ls /data
```

Expected output:

```
lost+found
static
```

**Step 4** Run the following command to delete the pod named **web-local-auto-0**:

```
kubectl delete pod statefulset-local-0
```

Expected output:

```
pod "statefulset-local-0" deleted
```

**Step 5** After the deletion, the StatefulSet controller automatically creates a replica with the same name. Run the following command to check whether the files in the **/data** path have been modified:

```
kubectl exec statefulset-local-0 -- ls /data
```

Expected output:

```
lost+found  
static
```

The **static** file is retained, indicating that the data in the local PV can be stored persistently.

----End

## Related Operations

You can also perform the operations listed in [Table 8-45](#).

**Table 8-45** Related operations

Operation	Description	Procedure
Viewing events	View event names, event types, number of occurrences, Kubernetes events, first occurrence time, and last occurrence time of the PVC or PV.	<ol style="list-style-type: none"> <li>1. Choose <b>Storage</b> in the navigation pane. In the right pane, click the <b>PVCs</b> or <b>PVs</b> tab.</li> <li>2. Click <b>View Events</b> in the <b>Operation</b> column of the target PVC or PV to view events generated within one hour (events are retained for one hour).</li> </ol>
Viewing a YAML file	View, copy, or download the YAML file of a PVC or PV.	<ol style="list-style-type: none"> <li>1. Choose <b>Storage</b> in the navigation pane. In the right pane, click the <b>PVCs</b> or <b>PVs</b> tab.</li> <li>2. Click <b>View YAML</b> in the <b>Operation</b> column of the target PVC or PV to view or download the YAML.</li> </ol>

## 8.8 Ephemeral Volumes

### 8.8.1 Overview

#### Introduction

Some applications require additional storage, but whether the data is still available after a restart is not important. For example, although cache services are

limited by memory size, cache services can move infrequently used data to storage slower than memory. As a result, overall performance is not impacted significantly. Other applications require read-only data injected as files, such as configuration data or secrets.

**Ephemeral volumes** (EVs) in Kubernetes are designed for the above scenario. EVs are created and deleted together with pods following the pod lifecycle.

Common EVs in Kubernetes:

- **emptyDir**: empty at pod startup, with storage coming locally from the kubelet base directory (usually the root disk) or memory. emptyDir is allocated from the **EV of the node**. If data from other sources (such as log files or image tiering data) occupies the ephemeral storage, the storage capacity may be insufficient.
- **ConfigMap**: Kubernetes data of the ConfigMap type is mounted to pods as data volumes.
- **Secret**: Kubernetes data of the Secret type is mounted to pods as data volumes.

## emptyDir Types

CCE provides the following emptyDir types:

- **Using a Temporary Path**: Kubernetes-native emptyDir type. Its lifecycle is the same as that of a pod. Memory can be specified as the storage medium. When the pod is deleted, the emptyDir volume is deleted and its data is lost.
- **Using a Local EV**: Local data disks in a node form a **storage pool** (VolumeGroup) through LVM. LVs are created as the storage medium of emptyDir and mounted to pods. LVs deliver better performance than the default storage medium of emptyDir.

## Notes and Constraints

- Local EVs are supported only when the cluster version is v1.21.2-r0 or later and the Everest add-on version is 1.2.29 or later.
- Do not manually delete the corresponding storage pool or detach data disks from the node. Otherwise, exceptions such as data loss may occur.
- Ensure that the `/var/lib/kubelet/pods/` directory is not mounted to the pod on the node. Otherwise, the pod, mounted with such volumes, may fail to be deleted.

## 8.8.2 Importing an EV to a Storage Pool

CCE allows you to use LVM to combine data volumes on nodes into a storage pool (VolumeGroup) and create LVs for containers to mount. Before creating a local EV, import the data disk of the node to the storage pool.

## Notes and Constraints

- Local EVs are supported only when the cluster version is v1.21.2-r0 or later and the Everest add-on version is 1.2.29 or later.
- The first data disk (used by container runtime and the kubelet component) on a node cannot be imported as a storage pool.

- Storage pools in striped mode do not support scale-out. After scale-out, fragmented space may be generated and the storage pool cannot be used.
- Storage pools cannot be scaled in or deleted.
- If disks in a storage pool on a node are deleted, the storage pool will malfunction.

## Importing a Storage Pool

### Imported during node creation

When creating a node, you can add a data disk to the node in **Storage Settings** and import the data disk to the storage pool as an EV. For details, see [Creating a Node](#).

**Figure 8-2** Importing as an EV

**Storage Settings** Configure storage resources for containers and applications on the node.

System Disk: High I/O 50 GB

Data Disk: High I/O 100 GB [Expand](#)

Used by the container runtime and kubelet. Do not uninstall this disk. Otherwise, the node will become unavailable. [How do I set data disk size?](#) [How do I allocate data disk space?](#)

High I/O 100 GB [Hide](#)

For a common data disk, you can choose not to perform any operation (by default) or attach it in a specified mode.

Mount Settings

Default  Mount Disk  Use as PV  Use as ephemeral volume

Data Disk Encryption [?](#)

Encryption

[Add Data Disk](#) Available for creation: 3

Ephemeral Volume Write:  Linear  Striped [?](#)

### Imported manually

If no EV is imported during node creation, or the capacity of the current storage volume is insufficient, you can manually import an EV.

- Step 1** Go to the ECS console and add a SCSI disk to the node. For details, see [Adding a Disk](#).
- Step 2** Log in to the CCE console and click the cluster name to access the cluster console.
- Step 3** Choose **Storage** in the navigation pane. In the right pane, click the **Storage Pool** tab.
- Step 4** View the node to which the disk has been added and select **Import as EV**. You can select a write mode during the import.

#### NOTE

If the manually attached disk is not displayed in the storage pool, wait for 1 minute and refresh the list.

- **Linear:** A linear logical volume integrates one or more physical volumes. Data is written to the next physical volume when the previous one is used up.
- **Striped:** A striped logical volume stripes data into blocks of the same size and stores them in multiple physical volumes in sequence. This allows data to be concurrently read and written. Select this option only when there are multiple volumes.

----End

## Expanding a Storage Pool

The capacity of a storage pool can be expanded in either of the following ways:

1. Add more large-capacity disks to the pool using the preceding manual import method.
2. Increase the size of the existing disks on the ECS console. Then, the storage pool capacity will be automatically expanded.

Either of these methods will work.

### 8.8.3 Using a Local EV

Local Ephemeral Volumes (EVs) are stored in EV [storage pools](#). Local EVs deliver better performance than the default storage medium of native emptyDir and support scale-out.

#### Prerequisites

- You have created a cluster and installed the CSI add-on ([Everest](#)) in the cluster.
- To create a cluster using commands, ensure kubectl is used. For details, see [Connecting to a Cluster Using kubectl](#).
- To use a local EV, import a data disk of a node to the local EV storage pool. For details, see [Importing an EV to a Storage Pool](#).

#### Notes and Constraints

- Local EVs are supported only when the cluster version is v1.21.2-r0 or later and the Everest add-on version is 1.2.29 or later.
- Do not manually delete the corresponding storage pool or detach data disks from the node. Otherwise, exceptions such as data loss may occur.
- Ensure that the `/var/lib/kubelet/pods/` directory is not mounted to the pod on the node. Otherwise, the pod, mounted with such volumes, may fail to be deleted.

## Using the Console to Mount a Local EV

**Step 1** Log in to the CCE console and click the cluster name to access the cluster console.

**Step 2** Choose **Workloads** in the navigation pane. In the right pane, click the **Deployments** tab.

- Step 3** Click **Create Workload** in the upper right corner. On the displayed page, click **Data Storage** in the **Container Settings** area and click **Add Volume** to select **Local Ephemeral Volume (emptyDir)**.
- Step 4** Mount and use storage volumes, as shown in [Table 8-46](#). For details about other parameters, see [Workloads](#).

**Table 8-46** Mounting a local EV

Parameter	Description
Capacity	Capacity of the requested storage volume.
Mount Path	<p>Enter a mount path, for example, <b>/tmp</b>.</p> <p>This parameter specifies a container path to which a data volume will be mounted. Do not mount the volume to a system directory such as <b>/</b> or <b>/var/run</b>. This may lead to container errors. Mount the volume to an empty directory. If the directory is not empty, ensure that there are no files that affect container startup. Otherwise, the files will be replaced, leading to container startup failures or workload creation failures.</p> <p><b>NOTICE</b> If a volume is mounted to a high-risk directory, use an account with minimum permissions to start the container. Otherwise, high-risk files on the host may be damaged.</p>
Subpath	Enter the subpath of the storage volume and mount a path in the storage volume to the container. In this way, different folders of the same storage volume can be used in a single pod. <b>tmp</b> , for example, indicates that data in the mount path of the container is stored in the <b>tmp</b> folder of the storage volume. If this parameter is left blank, the root path will be used by default.
Permission	<ul style="list-style-type: none"> <li>• <b>Read-only:</b> You can only read the data in the mounted volumes.</li> <li>• <b>Read-write:</b> You can modify the data volumes mounted to the path. Newly written data will not be migrated if the container is migrated, which may cause data loss.</li> </ul>

- Step 5** After the configuration, click **Create Workload**.

----End

## Mounting a Local EV Through kubectl

- Step 1** Use kubectl to access the cluster. For details, see [Connecting to a Cluster Using kubectl](#).
- Step 2** Create a file named **nginx-emptydir.yaml** and edit it.

**vi nginx-emptydir.yaml**

Content of the YAML file:

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-emptydir
  namespace: default
spec:
  replicas: 2
  selector:
    matchLabels:
      app: nginx-emptydir
  template:
    metadata:
      labels:
        app: nginx-emptydir
    spec:
      containers:
        - name: container-1
          image: nginx:latest
          volumeMounts:
            - name: vol-emptydir          # Volume name, which must be the same as the volume name in the
              mountPath: /tmp            # Location where the emptyDir is mounted
          imagePullSecrets:
            - name: default-secret
          volumes:
            - name: vol-emptydir          # Volume name, which can be customized
              emptyDir:
                medium: LocalVolume      # If the disk medium of emptyDir is set to LocalVolume, the local EV
is used.
              sizeLimit: 1Gi            # Volume capacity
```

**Step 3** Create a workload.

```
kubectl apply -f nginx-emptydir.yaml
```

```
----End
```

## Handling Local EV Exceptions

If a user manually detaches a disk from ECS or manually runs the `vgremove` command, the EV storage pool may malfunction. To resolve this issue, set the node to be unschedulable by following the procedure described in [Configuring a Node Scheduling Policy in One-Click Mode](#) and then reset the node.

### 8.8.4 Using a Temporary Path

A temporary path is of the Kubernetes-native emptyDir type. Its lifecycle is the same as that of a pod. Memory can be specified as the storage medium. When the pod is deleted, the emptyDir volume is deleted and its data is lost.

#### Using a Temporary Path on the Console

**Step 1** Log in to the CCE console and click the cluster name to access the cluster console.

**Step 2** Choose **Workloads** in the navigation pane. In the right pane, click the **Deployments** tab.

**Step 3** Click **Create Workload** in the upper right corner. On the displayed page, click **Data Storage** in the **Container Settings** area and click **Add Volume** to select **EmptyDir**.

**Step 4** Mount and use storage volumes, as shown in [Table 8-47](#). For details about other parameters, see [Workloads](#).

**Table 8-47** Mounting an EV

Parameter	Description
Storage Medium	<p><b>Memory:</b></p> <ul style="list-style-type: none"> <li>You can select this option to improve the running speed, but the storage capacity is subject to the memory size. This mode is applicable when data volume is small and efficient read and write is required.</li> <li>If this function is disabled, data is stored in hard disks, which applies to a large amount of data with low requirements on reading and writing efficiency.</li> </ul> <p><b>NOTE</b></p> <ul style="list-style-type: none"> <li>If <b>Memory</b> is selected, pay attention to the memory size. If the storage capacity exceeds the memory size, an OOM event occurs.</li> <li>If <b>Memory</b> is selected, the size of an EV is the same as pod specifications.</li> <li>If <b>Memory</b> is not selected, EVs will not occupy the system memory.</li> </ul>
Mount Path	<p>Enter a mount path, for example, <b>/tmp</b>.</p> <p>This parameter specifies a container path to which a data volume will be mounted. Do not mount the volume to a system directory such as <b>/</b> or <b>/var/run</b>. Otherwise, containers will be malfunctional. Mount the volume to an empty directory. If the directory is not empty, ensure that there are no files that affect container startup. Otherwise, the files will be replaced, leading to container startup failures or workload creation failures.</p> <p><b>NOTICE</b></p> <p>If a volume is mounted to a high-risk directory, use an account with minimum permissions to start the container. Otherwise, high-risk files on the host may be damaged.</p>
Subpath	<p>Enter the subpath of the storage volume and mount a path in the storage volume to the container. In this way, different folders of the same storage volume can be used in a single pod. <b>tmp</b>, for example, indicates that data in the mount path of the container is stored in the <b>tmp</b> folder of the storage volume. If this parameter is left blank, the root path is used by default.</p>
Permission	<ul style="list-style-type: none"> <li><b>Read-only:</b> You can only read the data in the mounted volumes.</li> <li><b>Read-write:</b> You can modify the data volumes mounted to the path. Newly written data will not be migrated if the container is migrated, which may cause data loss.</li> </ul>

**Step 5** After the configuration, click **Create Workload**.

----End



## Using a Temporary Path Through kubectl

**Step 1** Use kubectl to access the cluster. For details, see [Connecting to a Cluster Using kubectl](#).

**Step 2** Create a file named `nginx-emptydir.yaml` and edit it.

**vi nginx-emptydir.yaml**

Content of the YAML file:

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-emptydir
  namespace: default
spec:
  replicas: 2
  selector:
    matchLabels:
      app: nginx-emptydir
  template:
    metadata:
      labels:
        app: nginx-emptydir
    spec:
      containers:
        - name: container-1
          image: nginx:latest
          volumeMounts:
            - name: vol-emptydir # Volume name, which must be the same as the volume name in the
volumes field
              mountPath: /tmp # Location where the emptyDir is mounted
          imagePullSecrets:
            - name: default-secret
          volumes:
            - name: vol-emptydir # Volume name, which can be customized
emptyDir:
medium: Memory # EV disk medium: If this parameter is set to Memory, the memory is
enabled. If this parameter is left blank, the native default storage medium is used.
sizeLimit: 1Gi # Volume capacity
```

**Step 3** Create a workload.

**kubectl apply -f nginx-emptydir.yaml**

----End

## 8.9 hostPath

hostPath is used for mounting the file directory of the host where the container is located to the specified mount point of the container. If the container needs to access `/etc/hosts`, use hostPath to map `/etc/hosts`.

### NOTICE

- Avoid using hostPath volumes as much as possible, as they are prone to security risks. If hostPath volumes must be used, they can only be applied to files or directories and mounted in read-only mode.
- After the pod to which a hostPath volume is mounted is deleted, the data in the hostPath volume is retained.

## Mounting a hostPath Volume on the Console

You can mount a path on the host to a specified container path. A hostPath volume is usually used to **store workload logs permanently** or used by workloads that need to **access internal data structure of the Docker engine on the host**.

- Step 1** Log in to the CCE console.
- Step 2** When creating a workload, click **Data Storage** in **Container Settings**. Click **Add Volume** and choose **hostPath** from the drop-down list.
- Step 3** Set parameters for adding a local volume, as listed in [Table 8-48](#).

**Table 8-48** Setting parameters for mounting a hostPath volume

Parameter	Description
Volume Type	Select <b>HostPath</b> .
HostPath	<p>Path of the host to which the local volume is to be mounted, for example, <b>/etc/hosts</b>.</p> <p><b>NOTE</b>  <b>HostPath</b> cannot be set to the root directory <b>/</b>. Otherwise, the mounting fails. Mount paths can be as follows:</p> <ul style="list-style-type: none"> <li>• <b>/opt/xxxx</b> (excluding <b>/opt/cloud</b>)</li> <li>• <b>/mnt/xxxx</b> (excluding <b>/mnt/paas</b>)</li> <li>• <b>/tmp/xxx</b></li> <li>• <b>/var/xxx</b> (excluding key directories such as <b>/var/lib</b>, <b>/var/script</b>, and <b>/var/paas</b>)</li> <li>• <b>/xxxx</b> (It cannot conflict with the system directory, such as <b>bin</b>, <b>lib</b>, <b>home</b>, <b>root</b>, <b>boot</b>, <b>dev</b>, <b>etc</b>, <b>lost+found</b>, <b>mnt</b>, <b>proc</b>, <b>sbin</b>, <b>srv</b>, <b>tmp</b>, <b>var</b>, <b>media</b>, <b>opt</b>, <b>selinux</b>, <b>sys</b>, and <b>usr</b>.)</li> </ul> <p>Do not set this parameter to <b>/home/paas</b>, <b>/var/paas</b>, <b>/var/lib</b>, <b>/var/script</b>, <b>/mnt/paas</b>, or <b>/opt/cloud</b>. Otherwise, the system or node installation will fail.</p>
Mount Path	<p>Enter a mount path, for example, <b>/tmp</b>.</p> <p>This parameter specifies a container path to which a data volume will be mounted. Do not mount the volume to a system directory such as <b>/</b> or <b>/var/run</b>. Otherwise, containers will be malfunctional. Mount the volume to an empty directory. If the directory is not empty, ensure that there are no files that affect container startup. Otherwise, the files will be replaced, leading to container startup failures or workload creation failures.</p> <p><b>NOTICE</b>            If a volume is mounted to a high-risk directory, use an account with minimum permissions to start the container. Otherwise, high-risk files on the host may be damaged.</p>

Parameter	Description
Subpath	Enter the subpath of the storage volume and mount a path in the storage volume to the container. In this way, different folders of the same storage volume can be used in a single pod. <b>tmp</b> , for example, indicates that data in the mount path of the container is stored in the <b>tmp</b> folder of the storage volume. If this parameter is left blank, the root path is used by default.
Permission	<ul style="list-style-type: none"> <li>● <b>Read-only</b>: You can only read the data in the mounted volumes.</li> <li>● <b>Read-write</b>: You can modify the data volumes mounted to the path. Newly written data will not be migrated if the container is migrated, which may cause data loss.</li> </ul>

**Step 4** After the configuration, click **Create Workload**.

----End

## Mounting a hostPath Volume Through kubectl

**Step 1** Use kubectl to access the cluster.

**Step 2** Create a file named **nginx-hostpath.yaml** and edit it.

**vi nginx-hostpath.yaml**

The content of the YAML file is as follows. Mount the **/data** directory on the node to the **/data** directory in the container.

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-hostpath
  namespace: default
spec:
  replicas: 2
  selector:
    matchLabels:
      app: nginx-hostpath
  template:
    metadata:
      labels:
        app: nginx-hostpath
    spec:
      containers:
        - name: container-1
          image: nginx:latest
          volumeMounts:
            - name: vol-hostpath          # Volume name, which must be the same as the volume name in the
volumes field.
              mountPath: /data          # Location where the storage volume is mounted
          imagePullSecrets:
            - name: default-secret
          volumes:
            - name: vol-hostpath          # Volume name, which can be customized.
              hostPath:
                path: /data          # Directory location on the host node.
```

**Step 3** Create a workload.

```
kubectl apply -f nginx-hostpath.yaml
```

```
----End
```

## 8.10 StorageClass

### Introduction

A StorageClass in Kubernetes is a resource that categorizes storage types in a cluster. It can also serve as a configuration template for creating PVs. When creating a PVC or PV, you must specify a StorageClass. This resource object defines the storage mode and configuration parameters for dynamic volume provisioning, such as volume type, access mode, and volume lifecycle policy.

When creating a PVC, you only need to specify the **StorageClassName**. Kubernetes will then automatically create the PVs and underlying storage based on the StorageClass. This greatly reduces the need for manual creation and maintenance of PVs.

In addition to the [default StorageClasses](#) provided by CCE, you can customize StorageClasses. For details, see [Application Scenarios of Custom Storage](#).

### 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 delay
csi-nas             everest-csi-provisioner  17d   # SFS 1.0
csi-obs             everest-csi-provisioner  17d   # OBS
csi-sfsturbo       everest-csi-provisioner  17d   # SFS Turbo
```

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 8-49** Key parameters

Parameter	Description
provisioner	Specifies the storage resource provider, which is the Everest add-on for CCE. Set this parameter to <b>everest-csi-provisioner</b> .
parameters	Specifies the storage parameters, which vary with storage types. For details, see <a href="#">Table 8-50</a> .
reclaimPolicy	Specifies the value of <b>persistentVolumeReclaimPolicy</b> for creating a PV. The value can be <b>Delete</b> or <b>Retain</b> . If <b>reclaimPolicy</b> is not specified when a StorageClass object is created, the value defaults to <b>Delete</b> . <ul style="list-style-type: none"> <li>• <b>Delete</b>: 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.</li> <li>• <b>Retain</b>: 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.</li> </ul>
allowVolumeExpansion	Specifies whether the PV of this StorageClass supports dynamic capacity expansion. The default value is <b>false</b> . Dynamic capacity expansion is implemented by the underlying storage add-on. This is only a switch.
volumeBindingMode	Specifies the volume binding mode, that is, the time when a PV is dynamically created. The value can be <b>Immediate</b> or <b>WaitForFirstConsumer</b> . <ul style="list-style-type: none"> <li>• <b>Immediate</b>: After a PVC is created, the storage resources and PV will be created and associated with the PVC without delay.</li> <li>• <b>WaitForFirstConsumer</b>: 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.</li> </ul>
mountOptions	This field must be supported by the underlying storage. If this field is not supported but is specified, the PV creation will fail.

**Table 8-50** Parameters

V o l u m e T y p e	Param eter	Ma n d a t o r y	Description	
E V S	csi.stor age.k8s .io/csi- driver- name	Yes	Driver type. If an EVS disk is used, the parameter value is fixed at <b>disk.csi.everest.io</b> .	
	csi.stor age.k8s .io/ fstype	Yes	If an EVS disk is used, the parameter value can be <b>ext4</b> .	
	everest. io/disk- volume -type	Yes	EVS disk type. All letters are in uppercase. <ul style="list-style-type: none"> <li>• <b>SAS</b>: high I/O</li> <li>• <b>SSD</b>: ultra-high I/O</li> </ul>	
	everest. io/ passth rough	Yes	The parameter value is fixed at <b>true</b> , which indicates that the EVS device type is <b>SCSI</b> . Other parameter values are not allowed.	
	SFS Turbo	csi. sto rag e.k 8s.i o/ csi- dri ver - na me	Yes	Driver type. If SFS Turbo is used, the parameter value is fixed at <b>sfsturbo.csi.everest.io</b> .

V o l u m e T y p e	Param eter	Ma n d a t o r y	Description	
		csi. sto rag e.k 8s.i o/ fst ype	Yes	If SFS Turbo is used, the value can be <b>nfs</b> .
ev er es t.i o/ sh ar e- ac ce ss - to		Yes	VPC ID of the cluster.	
ev er es t.i o/ sh ar e- ex p a n d- ty p e		No	Extension type. The default value is <b>bandwidth</b> , indicating an enhanced file system. This parameter does not take effect.	

V o l u m e T y p e	Param eter	Ma n d a t o r y	Description	
ev er es t.i o/ sh ar e- so ur ce		Yes	The parameter value is fixed at <b>sfs-turbo</b> .	
ev er es t.i o/ sh ar e- v ol u m e- ty p e		No	SFS Turbo StorageClass. The default value is <b>STANDARD</b> , indicating standard and standard enhanced editions. This parameter does not take effect.	
O B S	csi.stor age.k8s .io/csi- driver- name	Yes	Driver type. If OBS is used, the parameter value is fixed at <b>obs.csi.everest.io</b> .	
	csi.stor age.k8s .io/ fstype	Yes	Instance type, which can be <b>obsfs</b> or <b>s3fs</b> . <ul style="list-style-type: none"><li>● <b>obsfs</b>: a parallel file system</li><li>● <b>s3fs</b>: object bucket</li></ul>	



V o l u m e T y p e	Param eter	Ma n d a t o r y	Description	
	everest.io/obs-volume-type	Yes	OBS StorageClass. <ul style="list-style-type: none"> <li>If <b>fsType</b> is set to <b>s3fs</b>, <b>STANDARD</b> (standard bucket) and <b>WARM</b> (infrequent access bucket) are supported.</li> <li>This parameter is invalid when <b>fsType</b> is set to <b>obsfs</b>.</li> </ul>	

## Application Scenarios of Custom Storage

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 a 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
```

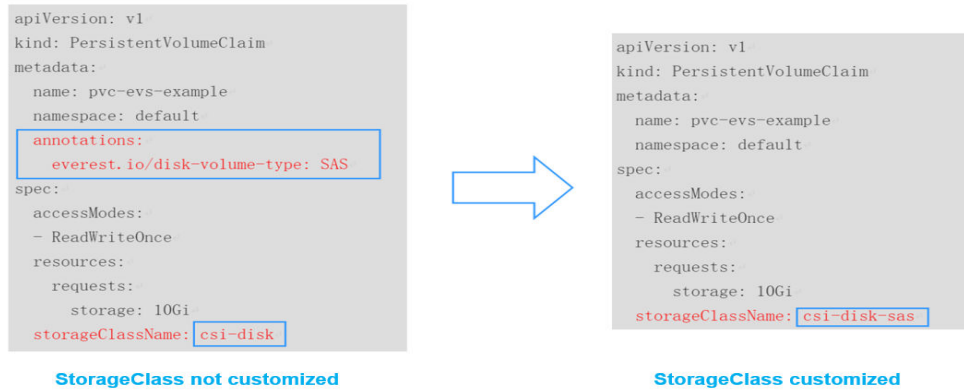
To specify the EVS disk type on CCE, use the **everest.io/disk-volume-type** field. SAS indicates the EVS disk type.

The preceding is a basic method of using StorageClass. In real-world scenarios, you can use StorageClass to perform other operations.

Application Scenario	Solution	Procedure
<p>When <b>annotations</b> is used to specify storage configuration, the configuration is complex. For example, the <b>everest.io/disk-volume-type</b> field is used to specify the EVS disk type.</p>	<p>Define PVC annotations in the <b>parameters</b> field of StorageClass. When compiling a YAML file, you only need to specify <b>StorageClassName</b>.</p> <p>For example, you can define SAS EVS disk and SSD EVS disk as a StorageClass, respectively. If a StorageClass named <b>csi-disk-sas</b> is defined, it is used to create SAS storage.</p>	<p><b>Scenario 1: Specify the Disk Type in a StorageClass</b></p>
<p>When a user migrates services from an on-premises Kubernetes cluster or other Kubernetes services to CCE, the StorageClass used in the original application YAML file is different from that used in CCE. As a result, a large number of YAML files or Helm chart packages need to be modified when the storage is used, which is complex and error-prone.</p>	<p>Create a StorageClass with the same name as that in the original application YAML file in the CCE cluster. After the migration, you do not need to modify the <b>StorageClassName</b> in the application YAML file.</p> <p>For example, the EVS disk StorageClass used before the migration is <b>disk-standard</b>. After migrating services to a CCE cluster, you can copy the YAML file of the <b>csi-disk</b> StorageClass in the CCE cluster, change its name to <b>disk-standard</b>, and create another StorageClass.</p>	
<p><b>StorageClassName</b> must be specified in the YAML file to use the storage. If not, the storage cannot be created.</p>	<p>If you set the default StorageClass in the cluster, you can create storage without specifying the <b>StorageClassName</b> in the YAML file.</p>	<p><b>Scenario 2: Specify the Default StorageClass</b></p>
<p>When creating storage resources, you must specify an enterprise project. Configuring annotations in each PVC can be a complex process.</p>	<p>Add an enterprise project to a StorageClass. In this case, you do not need to specify an enterprise project when creating a PVC. This is because storage resources will automatically be created in the enterprise project specified in the StorageClass.</p>	<p><b>Scenario 3: Specify an Enterprise Project for a StorageClass</b></p>

## Scenario 1: Specifying the Disk Type in a StorageClass

This section uses the custom StorageClass of EVS disks as an example to describe how to define SAS EVS disk and SSD EVS disk as a StorageClass, respectively. For example, if you define a StorageClass named **csi-disk-sas**, which is used to create SAS storage, the differences are shown in the following figure. When editing a PVC's YAML file, you only need to specify **StorageClassName**.



- You can customize a high I/O StorageClass 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 StorageClass, 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: SAS # High I/O EVS disk type, which cannot be customized.
  everest.io/passthrough: "true"
provisioner: everest-csi-provisioner
reclaimPolicy: Delete
volumeBindingMode: Immediate
allowVolumeExpansion: true # true indicates that capacity expansion is allowed.

```

- For an ultra-high I/O StorageClass, 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 StorageClass, 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 reclaim 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, both the PV and underlying storage resources will be retained. You need to manually delete these resources. After the PVC is deleted, the PV is in the **Released** state and cannot be bound to a PVC again.

If high data security is required, 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 **StorageClass** again. The command output is as follows:

```
# 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
```

## Scenario 2: Specifying the Default StorageClass

You can specify a StorageClass as the default class. In this way, any PVCs created without specifying **StorageClassName** will automatically use the default StorageClass.

For example, to specify **csi-disk-ssd** as the default StorageClass, 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" # Specify the default StorageClass in a cluster. There
can only be one default StorageClass per cluster.
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
```

Run the **kubectl create** command to create a StorageClass and then check it.

```
# kubectl create -f ssd.yaml
storageclass.storage.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 9s
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
```

## Scenario 3: Specifying an Enterprise Project for a StorageClass

When creating EVS or OBS PVCs using a StorageClass in CCE, you can specify an enterprise project to assign the created storage resources (EVS disks and OBS) to.

**This enterprise project can either be the default one or the same one as the cluster belongs to.**

If you do not specify any enterprise project, the enterprise project in StorageClass will be used by default. The storage resources created using the CCE's csi-disk or csi-obs StorageClass belong to the **default** enterprise project.

If you want the storage resources created using a StorageClass to be in the same enterprise project as the cluster, you can customize a StorageClass and specify the enterprise project ID in it.

 **NOTE**

To use this function, the Everest add-on must be upgraded to v1.2.33 or later.

```
kind: StorageClass
apiVersion: storage.k8s.io/v1
metadata:
  name: csi-disk-epid # Customize a StorageClass 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
```

# 9 Observability

---

## 9.1 Logging

### 9.1.1 Overview

Kubernetes logs allow you to locate and rectify faults. This section describes how you can manage Kubernetes logs generated for CCE in multiple ways.

You can manage Kubernetes logs generated for CCE in the following ways:

- Connect CCE to AOM. For details, see [Collecting Container Logs Using ICAgent](#).
- Collect control plane component logs and Kubernetes audit logs from the CCE control plane and add them to the LTS log streams in your account. For details, see [Collecting Control Plane Component Logs](#) and [Collecting Kubernetes Audit Logs](#).

### 9.1.2 Collecting Container Logs

#### 9.1.2.1 Collecting Container Logs Using ICAgent


CCE works with AOM to collect workload logs. When a node is created, ICAgent (a DaemonSet named **icagent** in the **kube-system** namespace of a cluster) of AOM is installed by default. ICAgent collects workload logs and reports them to AOM. You can view workload logs on the CCE or AOM console.

#### Constraints

ICAgent only collects text logs in .log, .trace, and .out formats.

#### Using ICAgent to Collect Logs

**Step 1** When [creating a workload](#), set logging for the container.

**Step 2** Click  to add a log policy.

**Step 3 Set Volume Type to `hostPath` or `emptyDir`.**

**Table 9-1** Configuring log policies

Parameter	Description
Volume Type	<ul style="list-style-type: none"> <li>● <b>hostPath</b>: A host path is mounted to the specified container path (mount path). In the node host path, you can view the container logs output into the mount path.</li> <li>● <b>emptyDir</b>: A temporary path of the node is mounted to the specified path (mount path). Log data that exists in the temporary path but is not reported by the collector to AOM will disappear after the pod is deleted.</li> </ul>
hostPath	Enter a host path, for example, <code>/var/paas/sys/log/nginx</code> .
Mount Path	<p>Container path (for example, <code>/tmp</code>) to which the storage resources will be mounted.</p> <p><b>NOTICE</b></p> <ul style="list-style-type: none"> <li>● Do not mount storage to a system directory such as <code>/</code> or <code>/var/run</code>; this action may cause a container error to occur. You are advised to mount the container to an empty directory. If the directory is not empty, ensure that there are no files affecting container startup in the directory. Otherwise, such files will be replaced, resulting in failures to start the container and create the workload.</li> <li>● If the container is mounted to a high-risk directory, you are advised to use an account with minimum permissions to start the container; otherwise, high-risk files on the host may be damaged.</li> <li>● AOM collects only the first 20 logs that have been modified recently. It collects logs from 2 levels of subdirectories by default.</li> <li>● AOM only collects <code>.log</code>, <code>.trace</code>, and <code>.out</code> text logs in mounting paths.</li> <li>● For details about how to set permissions for mount points in a container, see <a href="#">Configure a Security Context for a Pod or Container</a>.</li> </ul>
Extended Host Path	<p>This parameter is mandatory only if <b>Volume Type</b> is set to <b>HostPath</b>.</p> <p>Extended host paths contain pod IDs or container names to distinguish different containers into which the host path is mounted.</p> <p>A level-3 directory is added to the original volume directory/subdirectory. You can easily obtain the files output by a single Pod.</p> <ul style="list-style-type: none"> <li>● <b>None</b>: No extended path is configured.</li> <li>● <b>PodUID</b>: ID of a pod.</li> <li>● <b>PodName</b>: name of a pod.</li> <li>● <b>PodUID/ContainerName</b>: ID of a pod or name of a container.</li> <li>● <b>PodName/ContainerName</b>: name of a pod or container.</li> </ul>

Parameter	Description
Log Dump	<p>Log dump refers to rotating log files on a local host.</p> <ul style="list-style-type: none"> <li>• <b>Enabled:</b> AOM scans log files every minute. When a log file exceeds 50 MB, it is dumped. A new <b>.zip</b> file is generated in the directory where the log file locates. For a log file, AOM stores only the latest 20 <b>.zip</b> files. When the number of <b>.zip</b> files exceeds 20, earlier <b>.zip</b> files will be deleted.</li> <li>• <b>Disabled:</b> AOM does not dump log files.</li> </ul> <p><b>NOTE</b></p> <ul style="list-style-type: none"> <li>• AOM rotates log files using copytruncate. Before enabling log dumping, ensure that log files are written in the append mode. Otherwise, file holes may occur.</li> <li>• Currently, mainstream log components such as Log4j and Logback support log file rotation. If you have already set rotation for log files, skip the configuration. Otherwise, conflicts may occur.</li> <li>• You are advised to configure log file rotation for your own services to flexibly control the size and number of rolled files.</li> </ul>

**Step 4** Click **OK**.

----End

## YAML Example

You can set the container log storage path by defining a YAML file.

As shown in the following figure, an emptyDir volume is mounted a temporary path to **/var/log/nginx**. In this way, the ICAgent collects logs in **/var/log/nginx**. The **policy** field is customized by CCE and allows the ICAgent to identify and collect logs.

```

apiVersion: apps/v1
kind: Deployment
metadata:
  name: testlog
  namespace: default
spec:
  selector:
    matchLabels:
      app: testlog
  template:
    replicas: 1
    metadata:
      labels:
        app: testlog
    spec:
      containers:
        - image: 'nginx:alpine'
          name: container-0
          resources:
            requests:
              cpu: 250m
              memory: 512Mi
            limits:
              cpu: 250m
              memory: 512Mi

```



```

volumeMounts:
  - name: vol-log
    mountPath: /var/log/nginx
    policy:
      logs:
        rotate: "
volumes:
  - emptyDir: {}
    name: vol-log
imagePullSecrets:
  - name: default-secret

```

The following shows how to use a `hostPath` volume. Compared with `emptyDir`, the type of `volumes` is changed to `hostPath`, and the path on the host needs to be configured for this `hostPath` volume. In the following example, `/tmp/log` on the host is mounted to `/var/log/nginx`. In this way, the ICAgent can collect logs in `/var/log/nginx`, without deleting the logs from `/tmp/log`.

```

apiVersion: apps/v1
kind: Deployment
metadata:
  name: testlog
  namespace: default
spec:
  replicas: 1
  selector:
    matchLabels:
      app: testlog
  template:
    metadata:
      labels:
        app: testlog
    spec:
      containers:
        - image: 'nginx:alpine'
          name: container-0
          resources:
            requests:
              cpu: 250m
              memory: 512Mi
            limits:
              cpu: 250m
              memory: 512Mi
          volumeMounts:
            - name: vol-log
              mountPath: /var/log/nginx
              readOnly: false
              extendPathMode: PodUID
            policy:
              logs:
                rotate: Hourly
            annotations:

      volumes:
        - hostPath:
            path: /tmp/log
            name: vol-log
          imagePullSecrets:
            - name: default-secret

```

**Table 9-2** Parameter description

Parameter	Description	Description
extendPath Mode	Extended host path	<p>Extended host paths contain pod IDs or container names to distinguish different containers into which the host path is mounted.</p> <p>A level-3 directory is added to the original volume directory/subdirectory. You can easily obtain the files output by a single Pod.</p> <ul style="list-style-type: none"> <li>• <b>None:</b> No extended path is configured.</li> <li>• <b>PodUID:</b> ID of a pod.</li> <li>• <b>PodName:</b> name of a pod.</li> <li>• <b>PodUID/ContainerName:</b> ID of a pod or name of a container.</li> <li>• <b>PodName/ContainerName:</b> name of a pod or container.</li> </ul>
policy.logs.rotate	Log dump	<p>Log dump refers to rotating log files on a local host.</p> <ul style="list-style-type: none"> <li>• <b>Enabled:</b> AOM scans log files every minute. When a log file exceeds 50 MB, it is dumped immediately. A new <b>.zip</b> file is generated in the directory where the log file locates. For a log file, AOM stores only the latest 20 <b>.zip</b> files. When the number of <b>.zip</b> files exceeds 20, earlier <b>.zip</b> files will be deleted. After the dump is complete, the log file in AOM will be cleared.</li> <li>• <b>Disabled:</b> AOM does not dump log files.</li> </ul> <p><b>NOTE</b></p> <ul style="list-style-type: none"> <li>• AOM rotates log files using copytruncate. Before enabling log dumping, ensure that log files are written in the append mode. Otherwise, file holes may occur.</li> <li>• Currently, mainstream log components such as Log4j and Logback support log file rotation. If you have already set rotation for log files, skip the configuration. Otherwise, conflicts may occur.</li> <li>• You are advised to configure log file rotation for your own services to flexibly control the size and number of rolled files.</li> </ul>

## Viewing Logs

After a log collection path is configured and the workload is created, the ICAgent collects log files from the configured path. The collection takes about 1 minute.

After the log collection is complete, go to the workload details page and click **Logs** in the upper right corner to view logs.

You can also view logs on the AOM console.

You can also run the **kubectl logs** command to view the container stdout.

```
# View logs of a specified pod.
kubectl logs <pod_name>
kubectl logs -f <pod_name> # Similar to tail -f

# View logs of a specified container in a specified pod.
kubectl logs <pod_name> -c <container_name>

kubectl logs pod_name -c container_name -n namespace (one-off query)
kubectl logs -f <pod_name> -n namespace (real-time query in tail -f mode)
```

### 9.1.3 Collecting Control Plane Component Logs

CCE supports logging for master nodes. On the **Logging** page, you can select one or more control plane components (kube-controller-manager, kube-apiserver, and kube-scheduler) whose logs need to be reported.

#### Constraints

- The cluster version must be v1.21.7-r0 or later, v1.23.5-r0 or later, or 1.25.
- There is required LTS resource quota.

#### Control Plane Components

There are three control plane log types. Each log stream corresponds to a component of the Kubernetes control plane. To learn more about these components, see [Kubernetes Components](#).

**Table 9-3** Control plane components

Log Type	Component	Log Stream	Description
Control plane component logs	kube-apiserver	kube-apiserver-{{clusterID}}	It exposes Kubernetes APIs. For more information, see <a href="#">kube-apiserver</a> .
	kube-controller-manager	kube-controller-manager-{{clusterID}}	It manages controllers and embeds the core control loops shipped with Kubernetes. For more information, see <a href="#">kube-controller-manager</a> .
	kube-scheduler	kube-scheduler-{{clusterID}}	It manages when and where to run Pods in your cluster. For more information, see <a href="#">kube-scheduler</a> .

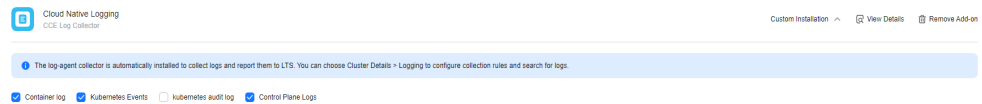
#### Enabling Control Plane Logging

##### Enabling control plane logging during cluster creation

1. Log in to the CCE console.
2. In the upper right corner, click **Buy Cluster**. Then, configure the parameters and click **Next: Select Add-on**.

3. On the displayed page, select **Cloud Native Logging** and click **Next: Add-on Configuration**.
4. On the displayed page, select **Control Plane Logs** for **Cloud Native Logging**.

**Figure 9-1** Enabling control plane logging during cluster creation

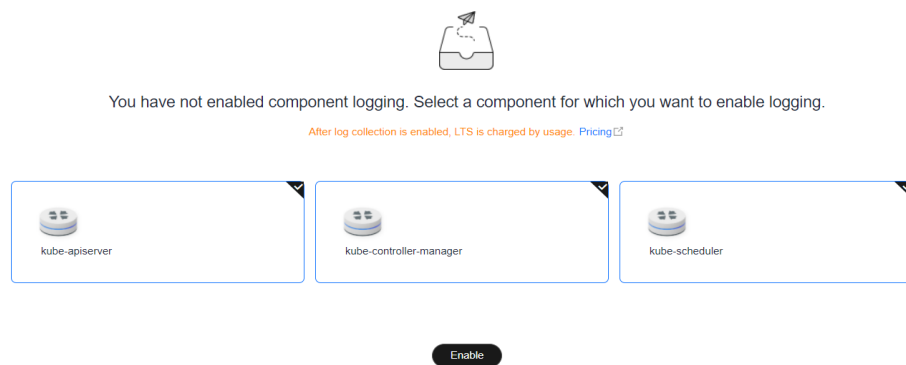


5. Click **Next: Confirm Configuration**.

### Enabling control plane logging for an existing cluster

1. Log in to the CCE console and click the cluster name to access the cluster console. In the navigation pane, choose **Logging**.
2. Click the **Control Plane Logs** tab, select one or more control plane components whose logs need to be collected, and click **Enable**.

**Figure 9-2** Selecting control plane components

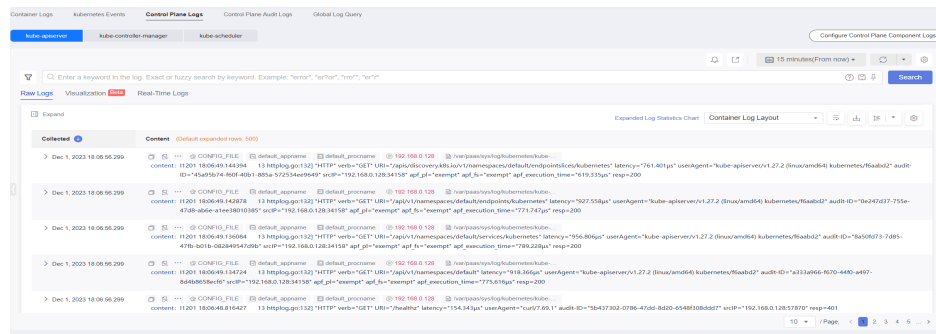


## Viewing Control Plane Component Logs of the Target Cluster

### Viewing control plane component logs of the target cluster on the CCE console

1. Log in to the CCE console and click the cluster name to access the cluster console. In the navigation pane, choose **Logging**.
2. Click the **Control Plane Logs** tab and select the topic of logs to be viewed. For details about available control plane log types, see [Control Plane Components](#).

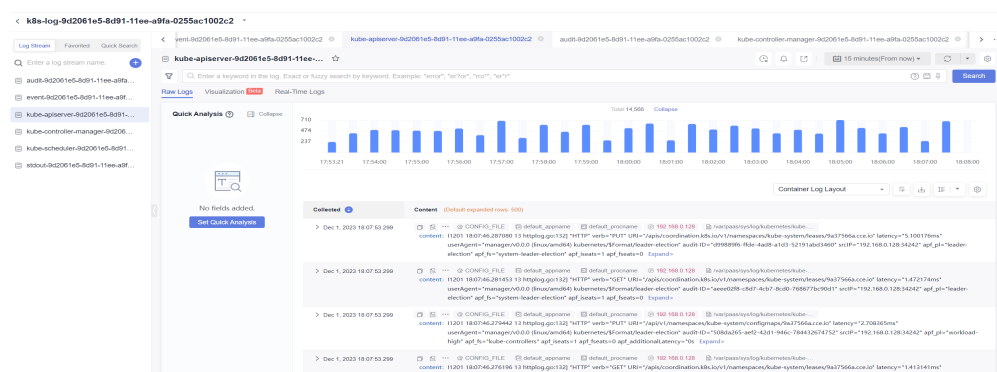
Figure 9-3 Viewing control plane component logs



### Viewing control plane component logs of the target cluster on the LTS console

1. Log in to the LTS console and choose **Log Management**.
2. Search for the log group by cluster ID and click the log group name to view the log streams.

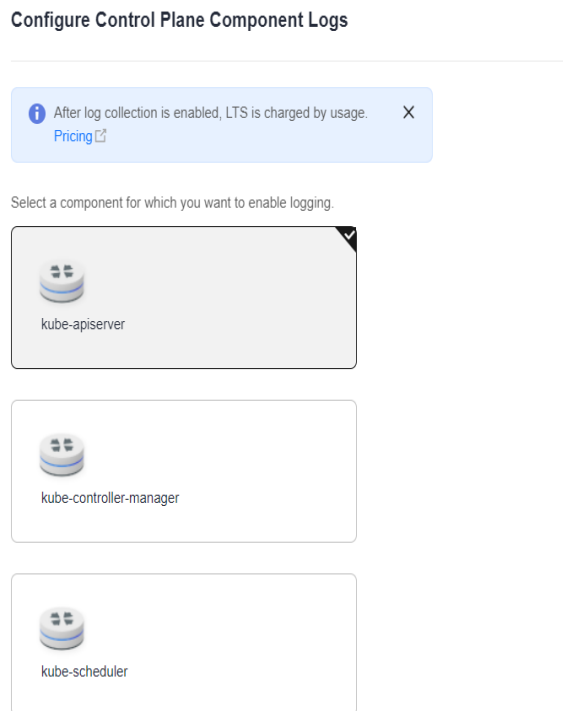
Figure 9-4 Viewing control plane component logs on the LTS console



### Disabling Control Plane Logging

1. Log in to the CCE console and click the cluster name to access the cluster console. In the navigation pane, choose **Logging**.
2. Click the **Control Plane Logs** tab. In the upper right corner, click **Configure Control Plane Component Logs**. Then, modify the log setting.

**Figure 9-5** Disabling control plane logging



3. Determine whether to enable logging for each component. If yes, click **OK**.

## 9.1.4 Collecting Kubernetes Audit Logs

CCE supports logging for master nodes. On the **Control Plane Audit Logs** tab of **Logging**, you can determine whether to report Kubernetes audit logs to LTS.

### Constraints

- The cluster version must be v1.21.7-r0 or later, v1.23.5-r0 or later, or 1.25.
- There is required LTS resource quota.

### Kubernetes Audit Logs

**Table 9-4** Kubernetes audit logs

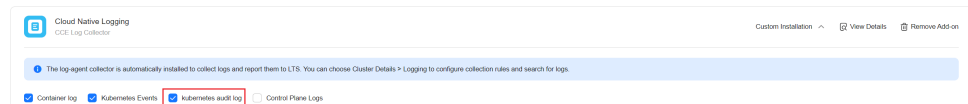
Log Type	Component	Log Stream	Description
Control plane audit logs	audit	audit- {{clusterID}}	An audit log is a chronological record of user operations on Kubernetes APIs and control plane activities for security.

## Enabling Control Plane Audit Logging

### Enabling control plane audit logging during cluster creation

1. Log in to the CCE console.
2. In the upper right corner, click **Buy Cluster**. Then, configure the parameters and click **Next: Select Add-on**.
3. On the displayed page, select **Cloud Native Logging** and click **Next: Add-on Configuration**.
4. On the displayed page, select **Kubernetes Audit Logs** for **Cloud Native Logging**.

**Figure 9-6** Enabling audit logging during cluster creation



5. Click **Next: Confirm Configuration**.

### Enabling control plane audit logging for an existing cluster

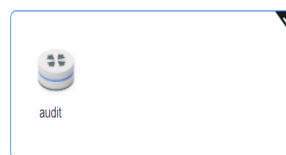
1. Log in to the CCE console and click the cluster name to access the cluster console. In the navigation pane, choose **Logging**.
2. Click the **Control Plane Audit Logs** tab, select the audit component, and click **Enable**.

**Figure 9-7** Enabling control plane audit logging for an existing cluster



You have not enabled audit logs. Select a component for which you want to enable audit logs.

After log collection is enabled, LTS is charged by usage. [Pricing](#)



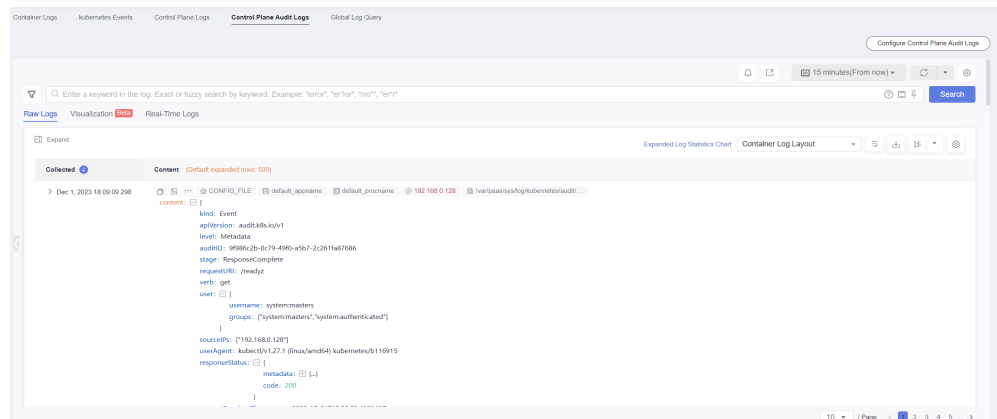
Enable

## Viewing Control Plane Audit Logs

### Viewing control plane audit logs on the CCE console

1. Log in to the CCE console and click the cluster name to access the cluster console. In the navigation pane, choose **Logging**.
2. Click the **Control Plane Audit Logs** tab to view audit logs in the cluster.

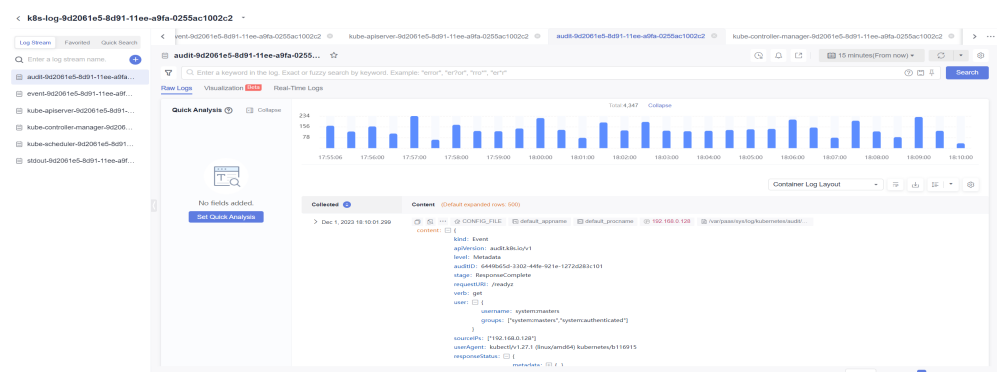
**Figure 9-8** Viewing control plane audit logs on the CCE console



### Viewing control plane audit logs on the TLS console

1. Log in to the LTS console and choose **Log Management**.
2. Search for the log group by cluster ID and click the log group name to view the log streams.

**Figure 9-9** Viewing control plane audit logs on the TLS console

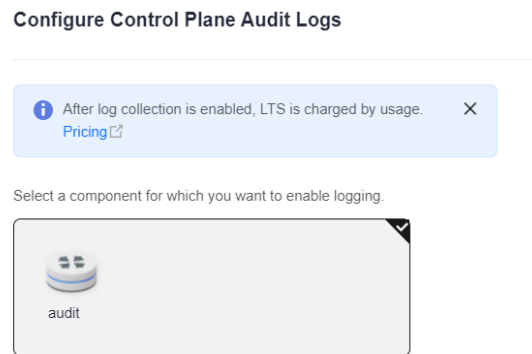


### Disabling Control Plane Audit Logging

1. Log in to the CCE console and click the cluster name to access the cluster console. In the navigation pane, choose **Logging**.
2. Click the **Control Plane Audit Logs** tab. In the upper right corner, click **Configure Control Plane Audit Logs**. Then, determine whether to enable control plane audit logging.



**Figure 9-10** Disabling control plane audit logging



3. Deselect **audit** and click **OK**.

**NOTE**

After you disable control plane audit logging, logs are no longer written to the original log stream, but the existing logs will not be deleted and expenses may be incurred for this.

## 9.2 Best Practices

### 9.2.1 Monitoring Custom Metrics Using Cloud Native Cluster Monitoring

CCE provides the Cloud Native Cluster Monitoring add-on to monitor custom metrics using Prometheus.

The following procedure uses an Nginx application as an example to describe how to use Prometheus to monitor custom metrics:

1. **Installing and Accessing Cloud Native Cluster Monitoring**

CCE provides an add-on that integrates Prometheus functions. You can install it with several clicks.

2. **Preparing an Application**

Prepare an application image. The application must provide a metric monitoring API for Prometheus to collect data, and the monitoring data must **comply with the Prometheus specifications**.

3. **Monitoring Custom Metrics**

Use the application image to deploy a workload in a cluster. Custom metrics will be automatically reported to Prometheus.

Use one of the following methods to monitor custom metrics:

- **Method 1: Configuring Pod Annotations**
- **Method 2: Configuring Service Annotations**
- **Method 3: Configuring PodMonitor**
- **Method 4: Configuring ServiceMonitor**

## Prometheus Monitoring Data Collection

Prometheus periodically calls the metric monitoring API (`/metrics` by default) of an application to obtain monitoring data. The application needs to provide the metric monitoring API for Prometheus to call, and the monitoring data must meet the following specifications of Prometheus:

```
# TYPE nginx_connections_active gauge
nginx_connections_active 2
# TYPE nginx_connections_reading gauge
nginx_connections_reading 0
```

Prometheus provides clients in various languages. For details about the clients, see [Prometheus CLIENT LIBRARIES](#). For details about how to develop an exporter, see [WRITING EXPORTERS](#). The Prometheus community provides various third-party exporters that can be directly used. For details, see [EXPORTERS AND INTEGRATIONS](#).

## Constraints

- To use Prometheus to monitor custom metrics, the application needs to provide a metric monitoring API. For details, see [Prometheus Monitoring Data Collection](#).
- Currently, metrics in the **kube-system** and **monitoring** namespaces cannot be collected when pod and service annotations are used. To collect metrics in the two namespaces, use PodMonitor and ServiceMonitor.
- The `nginx/nginx-prometheus-exporter:0.9.0` image is pulled for the Nginx application. You need to add an EIP for the node where the application is deployed or upload the image to SWR to prevent application deployment failures.

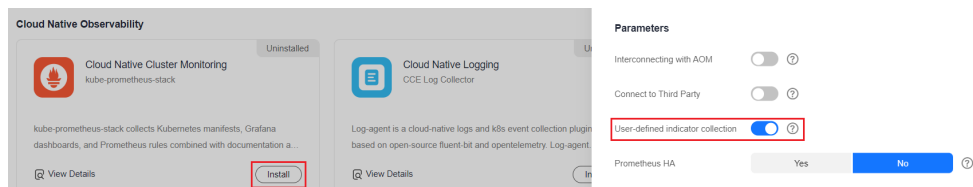
## Installing and Accessing Cloud Native Cluster Monitoring

**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**. On the displayed page, locate the Cloud Native Cluster Monitoring add-on and click **Install**. In addition to the monitoring capabilities, this add-on interconnects monitoring data with Monitoring Center.

When installing this add-on, pay attention to the following configurations. Configure other parameters as required. For details, see [Cloud Native Cluster Monitoring](#).

- For 3.8.0 or later, enable custom metric collection.



- For 3.8.0 or earlier, do not enable custom metric collection.

**Step 3** After this add-on is installed, deploy workloads and Services. The Prometheus server will be deployed as a StatefulSet in the **monitoring** namespace.

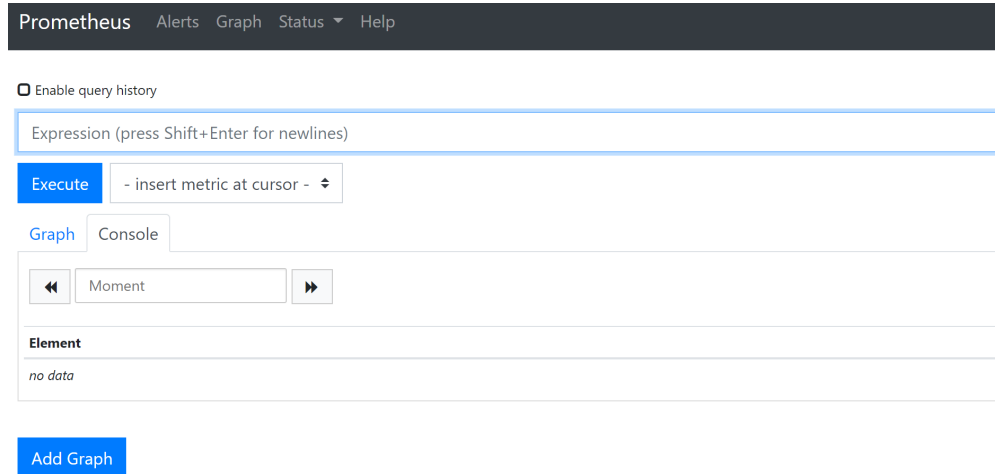
You can create a public network [LoadBalancer Service](#) so that Prometheus can be accessed from external networks.

1. Log in to the CCE console and click the name of the cluster with Prometheus installed to access the cluster console. In the navigation pane, choose **Services & Ingresses**.
2. Click **Create from YAML** in the upper right corner to create a public network LoadBalancer Service.

```
apiVersion: v1
kind: Service
metadata:
  name: prom-lb # Service name, which is customizable.
  namespace: monitoring
  labels:
    app: prometheus
    component: server
  annotations:
    kubernetes.io/elb.id: 038ff*** # Replace 038ff*** with the ID of the public network load balancer in the VPC that the cluster belongs to.
spec:
  ports:
    - name: cce-service-0
      protocol: TCP
      port: 88 # Service port, which is customizable.
      targetPort: 9090 # Default Prometheus port. Retain the default value.
  selector: # The label selector can be adjusted based on the label of a Prometheus server instance.
    app.kubernetes.io/name: prometheus
    prometheus: server
  type: LoadBalancer
```

3. After the Service is created, enter *Public IP address of the load balancer:Service port* in the address box of the browser to access Prometheus.

**Figure 9-11** Accessing Prometheus



----End

## Preparing an Application

User-developed applications must provide a metric monitoring API, and the monitoring data must comply with the Prometheus specifications. For details, see [Prometheus Monitoring Data Collection](#).

This section uses Nginx as an example to describe how to collect monitoring data. There is a module named **ngx\_http\_stub\_status\_module** in Nginx, which provides basic monitoring functions. You can configure the **nginx.conf** file to provide an interface for external systems to access Nginx monitoring data.

**Step 1** Log in to a Linux VM that can access to the Internet and run Docker commands.

**Step 2** Create an **nginx.conf** file. Add the server configuration under **http** to enable Nginx to provide an interface for the external systems to access the monitoring data.

```
user nginx;
worker_processes auto;

error_log /var/log/nginx/error.log warn;
pid /var/run/nginx.pid;

events {
    worker_connections 1024;
}

http {
    include /etc/nginx/mime.types;
    default_type application/octet-stream;
    log_format main '$remote_addr - $remote_user [$time_local] "$request" '
        '$status $body_bytes_sent "$http_referer" '
        '"$http_user_agent" "$http_x_forwarded_for"';

    access_log /var/log/nginx/access.log main;
    sendfile on;
    #tcp_nopush on;
    keepalive_timeout 65;
    #gzip on;
    include /etc/nginx/conf.d/*.conf;

    server {
        listen 8080;
        server_name localhost;
        location /stub_status {
            stub_status on;
            access_log off;
        }
    }
}
```


**Step 3** Use this configuration to create an image and a Dockerfile file.

```
vi Dockerfile
```

The content of Dockerfile is as follows:

```
FROM nginx:1.21.5-alpine
ADD nginx.conf /etc/nginx/nginx.conf
EXPOSE 80
CMD ["nginx", "-g", "daemon off;"]
```

**Step 4** Use this Dockerfile to create an image and upload it to SWR. The image name is **nginx:exporter**.

1. In the navigation pane, choose **My Images**. In the upper right corner, click **Upload Through Client**. In the displayed dialog box, click **Generate a temporary login command**. Then, click  to copy the command.
2. Run the login command copied in the previous step on the node. If the login is successful, the message "Login Succeeded" is displayed.
3. Run the following command to build an image named nginx. The image version is exporter.  

```
docker build -t nginx:exporter .
```
4. Tag the image and upload it to the image repository. Change the image repository address and organization name based on your requirements.

**Step 5** View application metrics.

1. Use **nginx:exporter** to create a workload.
2. [Access the container](#) and use `http://<ip_address>:8080/stub_status` to obtain nginx monitoring data. **<ip\_address>** indicates the IP address of the container. Information similar to the following is displayed.

```
# curl http://127.0.0.1:8080/stub_status
Active connections: 3
server accepts handled requests
146269 146269 212
Reading: 0 Writing: 1 Waiting: 2
```

----End

## Method 1: Configuring Pod Annotations

When the annotation settings of pods comply with the Prometheus data collection rules, Prometheus automatically collects the metrics exposed by the pods.

The format of the monitoring data provided by **nginx:exporter** does not meet the requirements of Prometheus. Convert the data format to the format required by Prometheus. To convert the format of Nginx metrics, use [nginx-prometheus-exporter](#). Deploy **nginx:exporter** and **nginx-prometheus-exporter** in the same pod and add the following annotations during deployment. Then Prometheus can automatically collect metrics.

```
kind: Deployment
apiVersion: apps/v1
metadata:
  name: nginx-exporter
  namespace: default
spec:
  replicas: 1
  selector:
    matchLabels:
      app: nginx-exporter
  template:
    metadata:
      labels:
        app: nginx-exporter
      annotations:
        prometheus.io/scrape: "true"
        prometheus.io/port: "9113"
        prometheus.io/path: "/metrics"
        prometheus.io/scheme: "http"
    spec:
      containers:
        - name: container-0
          image: 'nginx:exporter' # Replace it with the address of the image you uploaded to SWR.
          resources:
            limits:
              cpu: 250m
              memory: 512Mi
            requests:
              cpu: 250m
              memory: 512Mi
        - name: container-1
          image: 'nginx/nginx-prometheus-exporter:0.9.0'
          command:
            - nginx-prometheus-exporter
          args:
            - '-nginx.scrape-uri=http://127.0.0.1:8080/stub_status'
      imagePullSecrets:
        - name: default-secret
```

Where,

- **prometheus.io/scrape** indicates whether to enable Prometheus to collect pod monitoring data. The value is **true**.
- **prometheus.io/port** indicates the port for collecting monitoring data, which varies depending on the application. In this example, the port is 9113.
- **prometheus.io/path** indicates the URL of the API for collecting monitoring data. If this parameter is not set, the default value **/metrics** is used.
- **prometheus.io/scheme**: protocol used for data collection. The value can be **http** or **https**.

After the application is successfully deployed, [access Prometheus](#) to query custom metrics by job name.

The custom metrics related to Nginx can be queried. In the following, the job name indicates that the metrics are reported based on the pod configuration.

```
nginx_connections_accepted{cluster="2048c170-8359-11ee-9527-0255ac1000cf", cluster_category="CCE", cluster_name="cce-test", container="container-0", instance="10.0.0.46:9113", job="monitoring/kubernetes-pods", kubernetes_namespace="default", kubernetes_pod="nginx-exporter-77bf4d4948-zsb59", namespace="default", pod="nginx-exporter-77bf4d4948-zsb59", prometheus="monitoring/server"}
```

## Method 2: Configuring Service Annotations

When the annotation settings of Services comply with the Prometheus data collection rules, Prometheus automatically collects the metrics exposed by the Services.

You can use Service annotations in the same way as pod annotations. However, their application scenarios are different. Pod annotations focus on pod resource usage metrics while Service annotations focus on metrics such as requests for a Service.

The following is an example configuration:

```
kind: Deployment
apiVersion: apps/v1
metadata:
  name: nginx-test
  namespace: default
spec:
  replicas: 1
  selector:
    matchLabels:
      app: nginx-test
  template:
    metadata:
      labels:
        app: nginx-test
    spec:
      containers:
        - name: container-0
          image: 'nginx:exporter' # Replace it with the address of the image you uploaded to SWR.
          resources:
            limits:
              cpu: 250m
              memory: 512Mi
            requests:
              cpu: 250m
              memory: 512Mi
        - name: container-1
          image: 'nginx/nginx-prometheus-exporter:0.9.0'
          command:
            - nginx-prometheus-exporter
```

```
args:
  - '-nginx.scrape-uri=http://127.0.0.1:8080/stub_status'
imagePullSecrets:
  - name: default-secret
```

The following is an example Service configuration:

```
apiVersion: v1
kind: Service
metadata:
  name: nginx-test
  labels:
    app: nginx-test
  namespace: default
  annotations:
    prometheus.io/scrape: "true" # Value true indicates that service discovery is enabled.
    prometheus.io/port: "9113" # Set it to the port on which metrics are exposed.
    prometheus.io/path: "/metrics" # Enter the URI path under which metrics are exposed. Generally, the
value is /metrics.
spec:
  selector:
    app: nginx-test
  externalTrafficPolicy: Cluster
  ports:
    - name: cce-service-0
      targetPort: 80
      nodePort: 0
      port: 8080
      protocol: TCP
    - name: cce-service-1
      protocol: TCP
      port: 9113
      targetPort: 9113
  type: NodePort
```

After the application is successfully deployed, [access Prometheus](#) to query custom metrics. In the following, the Service name indicates the metrics are reported based on the Service configuration.

```
nginx_connections_accepted{app="nginx-test", cluster="2048c170-8359-11ee-9527-0255ac1000cf",
cluster_category="CCE", cluster_name="cce-test", instance="10.0.0.38:9113", job="nginx-test",
kubernetes_namespace="default", kubernetes_service="nginx-test", namespace="default", pod="nginx-
test-78cfb65889-gtv7z", prometheus="monitoring/server", service="nginx-test"}
```

### Method 3: Configuring PodMonitor

Cloud Native Cluster Monitoring allows you to configure metric collection tasks based on PodMonitor and ServiceMonitor. Prometheus Operator watches PodMonitor. The reload mechanism of Prometheus is used to trigger a hot update of the Prometheus collection tasks to the Prometheus instance.

To use CRDs defined by Prometheus Operator on GitHub, visit <https://github.com/prometheus-community/helm-charts/tree/main/charts/kube-prometheus-stack/charts/crds/crds>.

The following is an example configuration:

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-test2
  namespace: default
spec:
  replicas: 1
  selector:
    matchLabels:
```

```
  app: nginx-test2
template:
  metadata:
    labels:
      app: nginx-test2
  spec:
    containers:
      - image: nginx:exporter # Replace it with the address of the image you uploaded to SWR.
        name: container-0
        ports:
          - containerPort: 9113 # Port on which metrics are exposed.
            name: nginx-test2 # Application name used when PodMonitor is configured.
            protocol: TCP
        resources:
          limits:
            cpu: 250m
            memory: 300Mi
          requests:
            cpu: 100m
            memory: 100Mi
      - name: container-1
        image: 'nginx/nginx-prometheus-exporter:0.9.0'
        command:
          - nginx-prometheus-exporter
        args:
          - '-nginx.scrape-uri=http://127.0.0.1:8080/stub_status'
    imagePullSecrets:
      - name: default-secret
```

The following is an example PodMonitor configuration:

```
apiVersion: monitoring.coreos.com/v1
kind: PodMonitor
metadata:
  name: podmonitor-nginx # PodMonitor name
  namespace: monitoring # Namespace that PodMonitor belongs to. monitoring is recommended.
spec:
  namespaceSelector: # An selector matching the namespace where the workload is located
    matchNames:
      - default # Namespace that the workload belongs to
  jobLabel: podmonitor-nginx
  podMetricsEndpoints:
    - interval: 15s
      path: /metrics # Path under which metrics are exposed by the workload
      port: nginx-test2 # Port on which metrics are exposed by the workload
      tlsConfig:
        insecureSkipVerify: true
  selector:
    matchLabels:
      app: nginx-test2 # Label carried by the pod, which can be selected by the selector
```

After the application is successfully deployed, [access Prometheus](#) to query custom metrics. In the following, the job name indicates the metrics are reported based on the PodMonitor configuration.

```
nginx_connections_accepted{cluster="2048c170-8359-11ee-9527-0255ac1000cf", cluster_category="CCE",
cluster_name="cce-test", container="container-0", endpoint="nginx-test2", instance="10.0.0.44:9113",
job="monitoring/podmonitor-nginx", namespace="default", pod="nginx-test2-746b7f8fdd-krzfp",
prometheus="monitoring/server"}
```

## Method 4: Configuring ServiceMonitor

Cloud Native Cluster Monitoring allows you to configure metric collection tasks based on PodMonitor and ServiceMonitor. Prometheus Operator watches ServiceMonitor. The reload mechanism of Prometheus is used to trigger a hot update of the Prometheus collection tasks to the Prometheus instance.



To use CRDs defined by Prometheus Operator on GitHub, visit <https://github.com/prometheus-community/helm-charts/tree/main/charts/kube-prometheus-stack/charts/crds/crds>.

The following is an example configuration:

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-test3
  namespace: default
spec:
  replicas: 1
  selector:
    matchLabels:
      app: nginx-test3
  template:
    metadata:
      labels:
        app: nginx-test3
    spec:
      containers:
        - image: nginx:exporter          # Replace it with the address of the image you uploaded to SWR.
          name: container-0
          resources:
            limits:
              cpu: 250m
              memory: 300Mi
            requests:
              cpu: 100m
              memory: 100Mi
        - name: container-1
          image: 'nginx/nginx-prometheus-exporter:0.9.0'
          command:
            - nginx-prometheus-exporter
          args:
            - '-nginx.scrape-uri=http://127.0.0.1:8080/stub_status'
      imagePullSecrets:
        - name: default-secret
```

The following is an example Service configuration:

```
apiVersion: v1
kind: Service
metadata:
  name: nginx-test3
  labels:
    app: nginx-test3
  namespace: default
spec:
  selector:
    app: nginx-test3
  externalTrafficPolicy: Cluster
  ports:
    - name: cce-service-0
      targetPort: 80
      nodePort: 0
      port: 8080
      protocol: TCP
    - name: servicemonitor-ports
      protocol: TCP
      port: 9113
      targetPort: 9113
  type: NodePort
```

The following is an example ServiceMonitor configuration:

```
apiVersion: monitoring.coreos.com/v1
kind: ServiceMonitor
```

```

metadata:
  name: servicemonitor-nginx
  namespace: monitoring
spec:
  # Configure the name of the port on which metrics are exposed.
  endpoints:
  - path: /metrics
    port: servicemonitor-ports
  jobLabel: servicemonitor-nginx
  # Application scope of a collection task. If this parameter is not set, the default value default is used.
  namespaceSelector:
    matchNames:
    - default
  selector:
    matchLabels:
      app: nginx-test3
  
```

After the application is successfully deployed, [access Prometheus](#) to query custom metrics. In the following, the endpoint name indicates the metrics are reported based on the ServiceMonitor configuration.

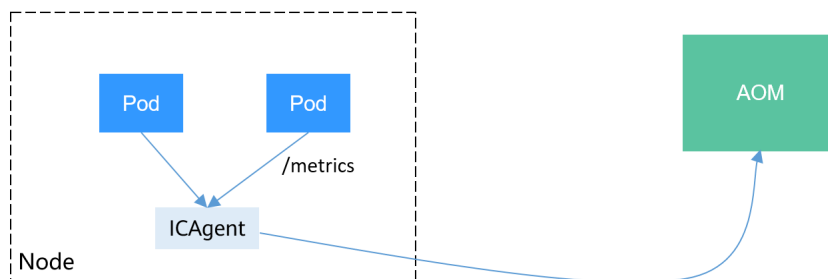
```

nginx_connections_accepted{cluster="2048c170-8359-11ee-9527-0255ac1000cf", cluster_category="CCE",
cluster_name="cce-test", endpoint="servicemonitor-ports", instance="10.0.0.47:9113", job="nginx-test3",
namespace="default", pod="nginx-test3-6f8bccd9-f27hv", prometheus="monitoring/server", service="nginx-
test3"}
  
```

## 9.2.2 Monitoring Custom Metrics on AOM

CCE allows you to upload custom metrics to AOM. ICAgent on a node periodically calls the metric monitoring API configured on a workload to read monitoring data and then uploads the data to AOM.

**Figure 9-12** Using ICAgent to collect monitoring metrics



The custom metric API of a workload can be configured when the workload is created. The following procedure uses an Nginx application as an example to describe how to report custom metrics to AOM.

### 1. Preparing an Application

Prepare an application image. The application must provide a metric monitoring API for ICAgent to collect data, and the monitoring data must **comply with the Prometheus specifications**.

### 2. Deploying Applications and Converting Nginx Metrics

Use the application image to deploy a workload in a cluster. Custom metrics are automatically reported.

### 3. Verification

Go to AOM to check whether the custom metrics are successfully collected.

## Constraints

- The ICAgent is compatible with the monitoring data specifications of [Prometheus](#). The custom metrics provided by pods can be collected by the ICAgent only when they meet the monitoring data specifications of Prometheus. For details, see [Prometheus Monitoring Data Collection](#).
- The ICAgent supports only [Gauge](#) metrics.
- The interval for the ICAgent to call the custom metric API is 1 minute, which cannot be changed.

## Prometheus Monitoring Data Collection

Prometheus periodically calls the metric monitoring API (`/metrics` by default) of an application to obtain monitoring data. The application needs to provide the metric monitoring API for Prometheus to call, and the monitoring data must meet the following specifications of Prometheus:

```
# TYPE nginx_connections_active gauge
nginx_connections_active 2
# TYPE nginx_connections_reading gauge
nginx_connections_reading 0
```

Prometheus provides clients in various languages. For details about the clients, see [Prometheus CLIENT LIBRARIES](#). For details about how to develop an exporter, see [WRITING EXPORTERS](#). The Prometheus community provides various third-party exporters that can be directly used. For details, see [EXPORTERS AND INTEGRATIONS](#).

## Preparing an Application

User-developed applications must provide a metric monitoring API, and the monitoring data must comply with the Prometheus specifications. For details, see [Prometheus Monitoring Data Collection](#).

This section uses Nginx as an example to describe how to collect monitoring data. There is a module named `ngx_http_stub_status_module` in Nginx, which provides basic monitoring functions. You can configure the `nginx.conf` file to provide an interface for external systems to access Nginx monitoring data.

**Step 1** Log in to a Linux VM that can access to the Internet and run Docker commands.

**Step 2** Create an `nginx.conf` file. Add the server configuration under `http` to enable Nginx to provide an interface for the external systems to access the monitoring data.

```
user nginx;
worker_processes auto;

error_log /var/log/nginx/error.log warn;
pid /var/run/nginx.pid;

events {
    worker_connections 1024;
}

http {
    include /etc/nginx/mime.types;
    default_type application/octet-stream;
    log_format main '$remote_addr - $remote_user [$time_local] "$request" '
        '$status $body_bytes_sent "$http_referer" ';
```

```
    "$http_user_agent" "$http_x_forwarded_for";

access_log /var/log/nginx/access.log main;
sendfile    on;
#tcp_nopush    on;
keepalive_timeout 65;
#gzip    on;
include /etc/nginx/conf.d/*.conf;

server {
    listen 8080;
    server_name localhost;
    location /stub_status {
        stub_status on;
        access_log off;
    }
}
```


**Step 3** Use this configuration to create an image and a Dockerfile file.

```
vi Dockerfile
```

The content of Dockerfile is as follows:

```
FROM nginx:1.21.5-alpine
ADD nginx.conf /etc/nginx/nginx.conf
EXPOSE 80
CMD ["nginx", "-g", "daemon off;"]
```

**Step 4** Use this Dockerfile to create an image and upload it to SWR. The image name is **nginx:exporter**.

1. In the navigation pane, choose **My Images**. In the upper right corner, click **Upload Through Client**. In the displayed dialog box, click **Generate a temporary login command**. Then, click  to copy the command.
2. Run the login command copied in the previous step on the node. If the login is successful, the message "Login Succeeded" is displayed.
3. Run the following command to build an image named nginx. The image version is exporter.

```
docker build -t nginx:exporter .
```
4. Tag the image and upload it to the image repository. Change the image repository address and organization name based on your requirements.

**Step 5** View application metrics.

1. Use **nginx:exporter** to create a workload.
2. [Access the container](#) and use `http://<ip_address>:8080/stub_status` to obtain nginx monitoring data. **<ip\_address>** indicates the IP address of the container. Information similar to the following is displayed.

```
# curl http://127.0.0.1:8080/stub_status
Active connections: 3
server accepts handled requests
146269 146269 212
Reading: 0 Writing: 1 Waiting: 2
```

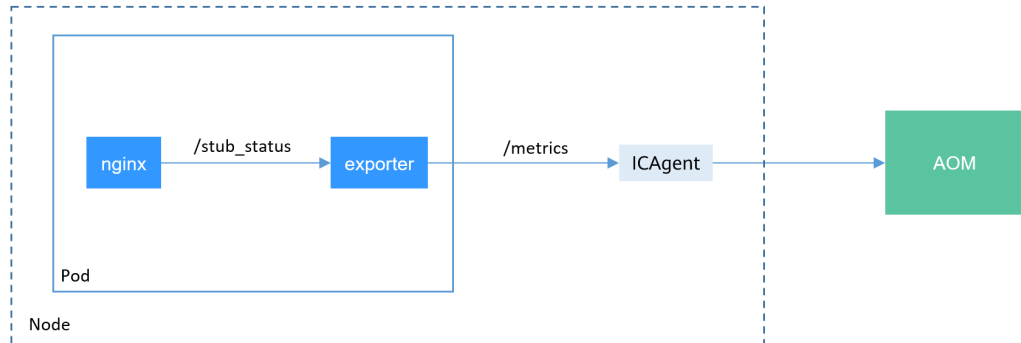
----End

## Deploying Applications and Converting Nginx Metrics

The format of the monitoring data provided by **nginx:exporter** does not meet the requirements of Prometheus. Convert the data format to the format required by

Prometheus. To convert the format of Nginx metrics, use **nginx-prometheus-exporter**, as shown in the following figure.

**Figure 9-13** Using exporter to convert the data format



Deploy **nginx:exporter** and **nginx-prometheus-exporter** in the same pod.

```

kind: Deployment
apiVersion: apps/v1
metadata:
  name: nginx-exporter
  namespace: default
spec:
  replicas: 1
  selector:
    matchLabels:
      app: nginx-exporter
  template:
    metadata:
      labels:
        app: nginx-exporter
    annotations:
      metrics.alpha.kubernetes.io/custom-endpoints: '[{"api":"prometheus","path":"/metrics","port":"9113","names":""}]'
    spec:
      containers:
        - name: container-0
          image: 'nginx:exporter' # Replace it with the address of the image you uploaded to SWR.
          resources:
            limits:
              cpu: 250m
              memory: 512Mi
            requests:
              cpu: 250m
              memory: 512Mi
        - name: container-1
          image: 'nginx/nginx-prometheus-exporter:0.9.0'
          command:
            - nginx-prometheus-exporter
          args:
            - '-nginx.scrape-uri=http://127.0.0.1:8080/stub_status'
      imagePullSecrets:
        - name: default-secret
  
```

**NOTE**

The **nginx/nginx-prometheus-exporter:0.9.0** image needs to be pulled from the public network. Therefore, a public IP address needs to be bound to each node in the cluster.

nginx-prometheus-exporter requires a startup command. **nginx-prometheus-exporter -nginx.scrape-uri=http://127.0.0.1:8080/stub\_status** is used to obtain Nginx monitoring data.

In addition, add an annotation `metrics.alpha.kubernetes.io/custom-endpoints: '[{"api": "prometheus", "path": "/metrics", "port": "9113", "names": ""}]'` to the pod.

## Verification

After an application is deployed, you can access Nginx to construct some access data and check whether the corresponding monitoring data can be obtained in AOM.

### Step 1 Obtain the pod name of Nginx.

```
$ kubectl get pod
NAME                                READY STATUS RESTARTS AGE
nginx-exporter-78859765db-6j8sw 2/2   Running 0      4m
```

### Step 2 Log in to the container and run commands to access Nginx.

```
$ kubectl exec -it nginx-exporter-78859765db-6j8sw -- /bin/sh
Defaulting container name to container-0.
Use 'kubectl describe pod/nginx-exporter-78859765db-6j8sw -n default' to see all of the containers in this pod.
/ # curl http://localhost
<!DOCTYPE html>
<html>
<head>
<title>Welcome to nginx!</title>
<style>
html { color-scheme: light dark; }
body { width: 35em; margin: 0 auto;
font-family: Tahoma, Verdana, Arial, sans-serif; }
</style>
</head>
<body>
<h1>Welcome to nginx!</h1>
<p>If you see this page, the nginx web server is successfully installed and
working. Further configuration is required.</p>

<p>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>.</p>

<p><em>Thank you for using nginx.</em></p>
</body>
</html>
/ #
```

### Step 3 Log in to AOM. In the navigation pane, choose **Monitoring > Metric Monitoring** to view Nginx-related metrics, for example, `nginx_connections_active`.

----End

## 9.2.3 Monitoring Metrics of Master Node Components Using Prometheus

This section describes how to use Prometheus to monitor the kube-apiserver, kube-controller, kube-scheduler and etcd-server components on the master node.

### Collecting Metrics of Master Node Components Using Self-Built Prometheus

This section describes how to collect metrics of master node components using self-built Prometheus.

**NOTICE**

- The cluster version must be 1.19 or later.
- You need to install self-built Prometheus using Helm by referring to [Prometheus](#). You need to use prometheus-operator to manage installed Prometheus by referring to [Prometheus Operator](#).

The Prometheus ([Prometheus](#)) add-on is end of maintenance and does not support this function. Therefore, do not use this add-on.

**Step 1** Use [kubectl](#) to connect to the cluster.

**Step 2** Modify the ClusterRole of Prometheus.

```
kubectl edit ClusterRole prometheus -n {namespace}
```

Add the following content under the **rules** field:

```
rules:
...
- apiGroups:
  - proxy.exporter.k8s.io
  resources:
  - "*"
  verbs: ["get", "list", "watch"]
```

**Step 3** Create a file named **kube-apiserver.yaml** and edit it.

```
vi kube-apiserver.yaml
```

Example file content:

```
apiVersion: monitoring.coreos.com/v1
kind: ServiceMonitor
metadata:
  labels:
    app.kubernetes.io/name: apiserver
  name: kube-apiserver
  namespace: monitoring # Change it to the namespace where Prometheus will be installed.
spec:
  endpoints:
    - bearerTokenFile: /var/run/secrets/kubernetes.io/serviceaccount/token
      interval: 30s
      metricRelabelings:
        - action: keep
          regex: (aggregator_unavailable_apiservice|
apiserver_admission_controller_admission_duration_seconds_bucket|
apiserver_admission_webhook_admission_duration_seconds_bucket|
apiserver_admission_webhook_admission_duration_seconds_count|
apiserver_client_certificate_expiration_seconds_bucket|apiserver_client_certificate_expiration_seconds_count|
apiserver_current_inflight_requests|apiserver_request_duration_seconds_bucket|apiserver_request_total|
go_goroutines|kubernetes_build_info|process_cpu_seconds_total|process_resident_memory_bytes|
rest_client_requests_total|workqueue_adds_total|workqueue_depth|
workqueue_queue_duration_seconds_bucket|aggregator_unavailable_apiservice_total|
rest_client_request_duration_seconds_bucket)
      sourceLabels:
        - __name__
        - action: drop
          regex: apiserver_request_duration_seconds_bucket;(0.15|0.25|0.3|0.35|0.4|0.45|0.6|0.7|0.8|0.9|1.25|1.5|1.75|
2.5|3|3.5|4.5|6|7|8|9|15|25|30|50)
      sourceLabels:
        - __name__
        - le
      port: https
      scheme: https
      tlsConfig:
        caFile: /var/run/secrets/kubernetes.io/serviceaccount/ca.crt
        serverName: kubernetes
      jobLabel: component
```

```
namespaceSelector:
  matchNames:
    - default
  selector:
    matchLabels:
      component: apiserver
      provider: kubernetes
```

Create a ServiceMonitor:

```
kubectl apply -f kube-apiserver.yaml
```

#### Step 4 Create a file named **kube-controller.yaml** and edit it.

```
vi kube-controller.yaml
```

Example file content:

```
apiVersion: monitoring.coreos.com/v1
kind: ServiceMonitor
metadata:
  labels:
    app.kubernetes.io/name: kube-controller
  name: kube-controller-manager
  namespace: monitoring # Change it to the namespace where Prometheus will be installed.
spec:
  endpoints:
    - bearerTokenFile: /var/run/secrets/kubernetes.io/serviceaccount/token
      interval: 15s
      honorLabels: true
      port: https
      relabelings:
        - regex: (.+)
          replacement: /apis/proxy.exporter.k8s.io/v1beta1/kube-controller-proxy/${1}/metrics
          sourceLabels:
            - __address__
          targetLabel: __metrics_path__
        - regex: (.+)
          replacement: ${1}
          sourceLabels:
            - __address__
          targetLabel: instance
        - replacement: kubernetes.default.svc.cluster.local:443
          targetLabel: __address__
      scheme: https
      tlsConfig:
        caFile: /var/run/secrets/kubernetes.io/serviceaccount/ca.crt
  jobLabel: app
  namespaceSelector:
    matchNames:
      - kube-system
  selector:
    matchLabels:
      app: kube-controller-proxy
  version: v1
```

Create a ServiceMonitor:

```
kubectl apply -f kube-controller.yaml
```

#### Step 5 Create a file named **kube-scheduler.yaml** and edit it.

```
vi kube-scheduler.yaml
```

Example file content:

```
apiVersion: monitoring.coreos.com/v1
kind: ServiceMonitor
metadata:
  labels:
    app.kubernetes.io/name: kube-scheduler
  name: kube-scheduler
  namespace: monitoring # Change it to the namespace where Prometheus will be installed.
```



```
spec:
  endpoints:
    - bearerTokenFile: /var/run/secrets/kubernetes.io/serviceaccount/token
      interval: 15s
      honorLabels: true
      port: https
      relabelings:
        - regex: (.+)
          replacement: /apis/proxy.exporter.k8s.io/v1beta1/kube-scheduler-proxy/${1}/metrics
          sourceLabels:
            - __address__
          targetLabel: __metrics_path__
        - regex: (.+)
          replacement: ${1}
          sourceLabels:
            - __address__
          targetLabel: instance
        - replacement: kubernetes.default.svc.cluster.local:443
          targetLabel: __address__
      scheme: https
      tlsConfig:
        caFile: /var/run/secrets/kubernetes.io/serviceaccount/ca.crt
  jobLabel: app
  namespaceSelector:
    matchNames:
      - kube-system
  selector:
    matchLabels:
      app: kube-scheduler-proxy
  version: v1
```

Create a ServiceMonitor:

```
kubectl apply -f kube-scheduler.yaml
```

## Step 6 Create a file named **etcd-server.yaml** and edit it.

```
vi etcd-server.yaml
```

Example file content:

```
apiVersion: monitoring.coreos.com/v1
kind: ServiceMonitor
metadata:
  labels:
    app.kubernetes.io/name: etcd-server
  name: etcd-server
  namespace: monitoring # Change it to the namespace where Prometheus will be installed.
spec:
  endpoints:
    - bearerTokenFile: /var/run/secrets/kubernetes.io/serviceaccount/token
      interval: 15s
      honorLabels: true
      port: https
      relabelings:
        - regex: (.+)
          replacement: /apis/proxy.exporter.k8s.io/v1beta1/etcd-server-proxy/${1}/metrics
          sourceLabels:
            - __address__
          targetLabel: __metrics_path__
        - regex: (.+)
          replacement: ${1}
          sourceLabels:
            - __address__
          targetLabel: instance
        - replacement: kubernetes.default.svc.cluster.local:443
          targetLabel: __address__
      scheme: https
      tlsConfig:
        caFile: /var/run/secrets/kubernetes.io/serviceaccount/ca.crt
  jobLabel: app
```

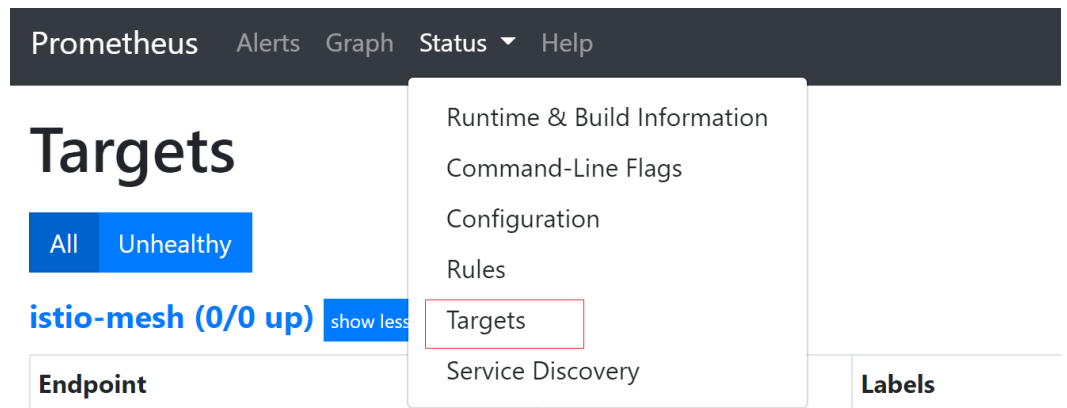
```
namespaceSelector:  
  matchNames:  
    - kube-system  
selector:  
  matchLabels:  
    app: etcd-server-proxy  
    version: v1
```

Create a ServiceMonitor:

```
kubectl apply -f etcd-server.yaml
```

**Step 7** Access Prometheus and choose **Status > Targets**.

The preceding master node components are displayed.



----End

# 10 Auto Scaling

---

## 10.1 Overview

Auto scaling is a service that automatically and economically adjusts service resources based on your service requirements and configured policies.

### Context

More and more applications run on Kubernetes. It becomes increasingly important to quickly scale out applications on Kubernetes to cope with service peaks and to scale in applications during off-peak hours to save resources and reduce costs.

In a Kubernetes cluster, auto scaling involves pods and nodes. A pod is an application instance. Each pod contains one or more containers and runs on a node (VM or bare-metal server). If a cluster does not have sufficient nodes to run new pods, add nodes to the cluster to ensure service running.

In CCE, auto scaling is used for online services, large-scale computing and training, deep learning GPU or shared GPU training and inference, periodic load changes, and many other scenarios.

### Auto Scaling in CCE

**CCE supports auto scaling for workloads and nodes.**

- **Workload scaling:** Auto scaling at the scheduling layer to change the scheduling capacity of workloads. For example, you can use the HPA, a scaling component at the scheduling layer, to adjust the number of replicas of an application. Adjusting the number of replicas changes the scheduling capacity occupied by the current workload, thereby enabling scaling at the scheduling layer.
- **Node scaling:** Auto scaling at the resource layer. When the planned cluster nodes cannot allow workload scheduling, ECS resources are provided to support scheduling.

### Components

## Workload Scaling Types

**Table 10-1** Workload scaling types

Type	Component	Component Description	Reference
HPA	HorizontalPodAutoscaler (built-in Kubernetes component)	HorizontalPodAutoscaler is a built-in component of Kubernetes for Horizontal Pod Autoscaling (HPA). CCE incorporates the application-level cooldown time window and scaling threshold functions into Kubernetes HPA.	<a href="#">Creating an HPA Policy</a>
CustomedHPA	<a href="#">CCE Advanced HPA</a>	An enhanced auto scaling feature, used for auto scaling of Deployments based on metrics (CPU usage and memory usage) or at a periodic interval (a specific time point every day, every week, every month, or every year).	<a href="#">Creating a CustomedHPA Policy</a>
CronHPA	<a href="#">CCE Advanced HPA</a>	CronHPA can scale in or out a cluster at a fixed time. It can work with HPA policies to periodically adjust the HPA scaling scope, implementing workload scaling in complex scenarios.	<a href="#">Creating a Scheduled CronHPA Policy</a>

## Node Scaling Types

**Table 10-2** Node scaling types

Component Name	Component Description	Application Scenario	Reference
<a href="#">CCE Cluster Autoscaler</a>	An open source Kubernetes component for horizontal scaling of nodes, which is optimized by CCE in scheduling, auto scaling, and costs.	Online services, deep learning, and large-scale computing with limited resource budgets	<a href="#">Creating a Node Scaling Policy</a>

# 10.2 Scaling a Workload

## 10.2.1 Workload Scaling Rules

### How HPA Works

HPA is a controller that controls horizontal pod scaling. HPA periodically checks the pod metrics, calculates the number of pods required to meet the target values configured for HPA resources, and then adjusts the value of the **replicas** field in the target resource object (such as a Deployment).

A prerequisite for auto scaling is that your container running data can be collected, such as number of cluster nodes/pods, and CPU and memory usage of containers. Kubernetes does not have built-in monitoring capabilities, but you can use extensions like **Metrics Server** to monitor and collect data.

- **Metrics Server** is a cluster-wide aggregator of resource utilization data. Metrics Server collects metrics from the Summary API exposed by kubelet. These metrics are set for core Kubernetes resources, such as pods, nodes, containers, and Services. Metrics Server provides a set of standard APIs for external systems to collect these metrics.

HPA can work with Metrics Server for auto scaling based on the CPU and memory usage.

#### Two core modules of HPA:

- Data Source Monitoring

The community provided only CPU- and memory-based HPA at the early stage. With the population of Kubernetes, developers need more custom metrics or monitoring information at the access layer for their own applications, for example, the QPS of the load balancer and the number of online users of the website. In response, the community defines a set of standard metric APIs to provide services externally through these aggregated APIs.

  - **metrics.k8s.io** provides monitoring metrics related to the CPU and memory of pods and nodes.
  - **custom.metrics.k8s.io** provides custom monitoring metrics related to Kubernetes objects.
  - **external.metrics.k8s.io** provides metrics that come from external systems and are irrelevant to any Kubernetes resource metrics.
- Scaling Decision-Making Algorithms

The HPA controller calculates the scaling ratio based on the current metric values and desired metric values using the following formula:

**Desired number of pods = Rounded up value of [Number of current pods x (Current metric value/Target value)]**

For example, if the current metric value is 200m and the target value is 100m, the desired number of pods will be doubled according to the formula. In practice, pods may be constantly added or reduced. To ensure stability, the HPA controller is optimized from the following aspects:

  - Cooldown interval: In v1.11 and earlier versions, Kubernetes introduced the startup parameters **horizontal-pod-autoscaler-downscale-stabilization-window** and **horizontal-pod-autoscaler-upscale-stabilization-window** to indicate the cooldown intervals after a scale-in

and scale-out, respectively, in which no scaling operation will not be performed. In versions later than v1.14, the scheduling queue is introduced to store all decision-making suggestions detected within a period of time. Then, the system makes decisions based on all valid decision-making suggestions to minimize changes of the desired number of pods to ensure stability.

- Tolerance: It can be considered as a buffer zone. If the pod number changes can be tolerated, the number of pods remains unchanged.

**ratio = Current metric value/Target value**

When  $|\text{ratio} - 1.0| \leq \text{tolerance}$ , scaling will not be performed.

When  $|\text{ratio} - 1.0| > \text{tolerance}$ , the desired value is calculated using the formula mentioned above.

The default value is 0.1 in the current community version.

The HPA performs scaling based on metric thresholds. Common metrics include the CPU and memory usage. You can also set custom metrics, such as the QPS and number of connections, to trigger scaling. However, metric-based scaling brings in latency of minutes generated during data collection, determination, and scaling phases. Such latency may cause high CPU usage and slow response. To solve this problem, CCE allows you to configure scheduled policies to scale resources regularly for applications with periodic changes.

## 10.2.2 Creating an HPA Policy

Horizontal Pod Autoscaling (HPA) in Kubernetes implements horizontal scaling of pods. In a CCE HPA policy, you can configure different cooldown time windows and scaling thresholds for different applications based on the Kubernetes HPA.

### Prerequisites

To use HPA, install an add-on that provides metrics APIs. Select one of the following add-ons based on your cluster version and service requirements.

- **Kubernetes Metrics Server**: provides basic resource usage metrics, such as container CPU and memory usage. It is supported by all cluster versions.
- **Cloud Native Cluster Monitoring**: available only in clusters of v1.17 or later.
  - Auto scaling based on basic resource metrics: Prometheus needs to be registered as a metrics API. For details, see [Providing Resource Metrics Through the Metrics API](#).
  - Auto scaling based on custom metrics: Custom metrics need to be aggregated to the Kubernetes API server. For details, see [Creating an HPA Policy Using Custom Metrics](#).
- **Prometheus** : Prometheus needs to be registered as a metrics API. For details, see [Providing Resource Metrics Through the Metrics API](#). This add-on supports only clusters of v1.21 or earlier.

### Notes and Constraints

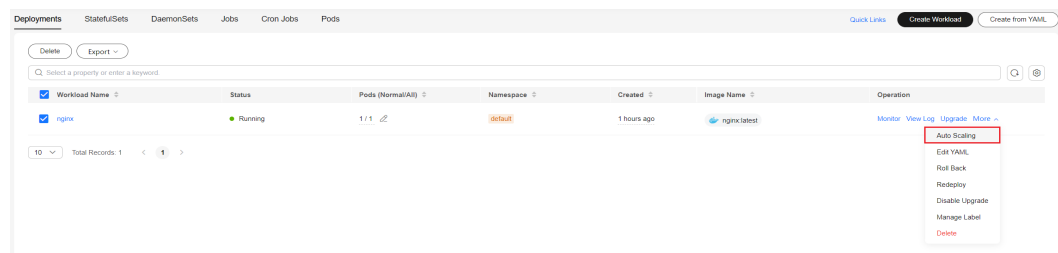
- HPA policies can be created only for clusters of v1.13 or later.
- For clusters earlier than v1.19.10, if an HPA policy is used to scale out a workload with EVS volumes mounted, the existing pods cannot be read or written when a new pod is scheduled to another node.

For clusters of v1.19.10 and later, if an HPA policy is used to scale out a workload with EVS volumes mounted, a new pod cannot be started because EVS disks cannot be attached.

## Procedure

- Step 1** Log in to the CCE console and click the cluster name to access the cluster console.
- Step 2** Choose **Workloads** in the navigation pane. Locate the target workload and choose **More > Auto Scaling** in the **Operation** column.

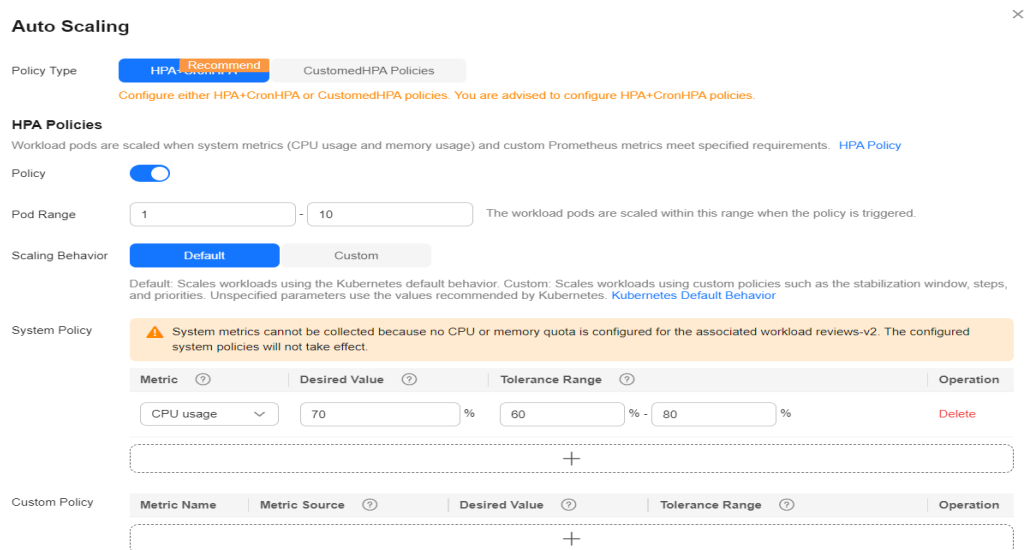
**Figure 10-1** Scaling a workload



- Step 3** Set **Policy Type** to **HPA+CronHPA**, enable the created HPA policy, and configure parameters.

This section describes only HPA policies. To enable CronHPA, see [Creating a Scheduled CronHPA Policy](#).

**Figure 10-2** Enabling the HPA policy



**Table 10-3** HPA policy

Parameter	Description
Pod Range	Minimum and maximum numbers of pods. When a policy is triggered, the workload pods are scaled within this range.

Parameter	Description
Cooldown Period	<p>Interval between a scale-in and a scale-out. The unit is minute. <b>The interval cannot be shorter than 1 minute.</b></p> <p><b>This parameter is supported only in clusters of v1.15 to v1.23.</b></p> <p>This parameter indicates the interval between consecutive scaling operations. The cooldown period ensures that a scaling operation is initiated only when the previous one is completed and the system is running stably.</p>
Scaling Behavior	<p><b>This parameter is supported only in clusters of v1.25 or later.</b></p> <ul style="list-style-type: none"> <li>● <b>Default:</b> scales workloads using the Kubernetes default behavior. For details, see <a href="#">Default Behavior</a>.</li> <li>● <b>Custom:</b> scales workloads using custom policies such as stabilization window, steps, and priorities. Unspecified parameters use the values recommended by Kubernetes. <ul style="list-style-type: none"> <li>– <b>Disable scale-out/scale-in:</b> Select whether to disable scale-out or scale-in.</li> <li>– <b>Stabilization Window:</b> a period during which CCE continuously checks whether the metrics used for scaling keep fluctuating. CCE triggers scaling if the desired state is not maintained for the entire window. This window restricts the unwanted flapping of pod count due to metric changes.</li> <li>– <b>Step:</b> specifies the scaling step. You can set the number or percentage of pods to be scaled in or out within a specified period. If there are multiple policies, you can select the policy that maximizes or minimizes the number of pods.</li> </ul> </li> </ul>
System Policy	<ul style="list-style-type: none"> <li>● <b>Metric:</b> You can select <b>CPU usage</b> or <b>Memory usage</b>. <b>NOTE</b> Usage = Average resource usage of all pods in a workload/ Requested resources</li> <li>● <b>Desired Value:</b> Enter the desired average resource usage. This parameter indicates the desired value of the selected metric. Number of target pods (rounded up) = (Current metric value/Desired value) x Number of current pods <b>NOTE</b> When calculating the number of pods to be added or reduced, the HPA policy uses the maximum number of pods in the last 5 minutes.</li> <li>● <b>Tolerance Range:</b> Scaling is not triggered when the metric value is within the tolerance range. The desired value must be within the tolerance range. If the metric value is greater than the scale-in threshold and less than the scale-out threshold, no scaling is triggered. <b>This parameter is supported only in clusters of v1.15 or later.</b></li> </ul>



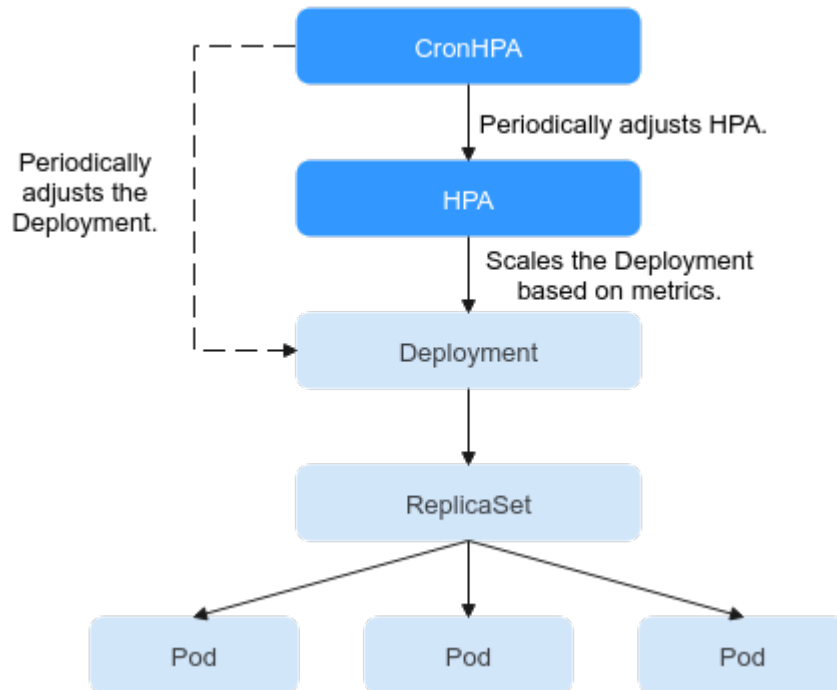
Parameter	Description
Custom Policy (supported only in clusters of v1.15 or later)	<p><b>NOTE</b> Before creating a custom policy, install an add-on that supports custom metric collection (for example, Prometheus) in the cluster. Ensure that the add-on can collect and report the custom metrics of the workloads. For details, see <a href="#">Monitoring Custom Metrics Using Cloud Native Cluster Monitoring</a>.</p> <ul style="list-style-type: none"> <li>• <b>Metric Name:</b> name of the custom metric. You can select a name as prompted.</li> <li>• <b>Metric Source:</b> Select an object type from the drop-down list. You can select <b>Pod</b>.</li> <li>• <b>Desired Value:</b> the average metric value of all pods. Number of pods to be scaled (rounded up) = (Current metric value/ Desired value) x Number of current pods</li> </ul> <p><b>NOTE</b> When calculating the number of pods to be added or reduced, the HPA policy uses the maximum number of pods in the last 5 minutes.</p> <ul style="list-style-type: none"> <li>• <b>Tolerance Range:</b> Scaling is not triggered when the metric value is within the tolerance range. The desired value must be within the tolerance range.</li> </ul>

**Step 4** Click **Create**.

----End

### 10.2.3 Creating a Scheduled CronHPA Policy

There are predictable and unpredictable traffic peaks for some services. For such services, CCE CronHPA allows you to scale resources in fixed periods. It can work with HPA policies to periodically adjust the HPA scaling scope, implementing workload scaling.



CronHPA can periodically adjust the maximum and minimum numbers of pods in the HPA policy or directly adjust the number of pods of a Deployment.

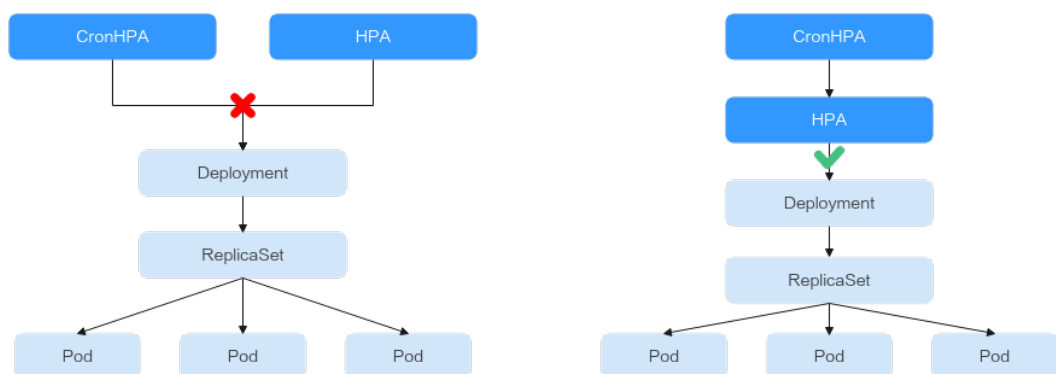
## Prerequisites

The add-on [CCE Advanced HPA](#) of v1.2.13 or later has been installed.

## Using CronHPA to Adjust the HPA Scaling Scope

CronHPA can periodically scale out/in pods in HPA policies to satisfy complex services.

HPA and CronHPA associate scaling objects using the `scaleTargetRef` field. If a Deployment is the scaling object for both CronHPA and HPA, the two scaling policies are independent of each other. The operation performed later overwrites the operation performed earlier. As a result, the scaling effect does not meet the expectation.



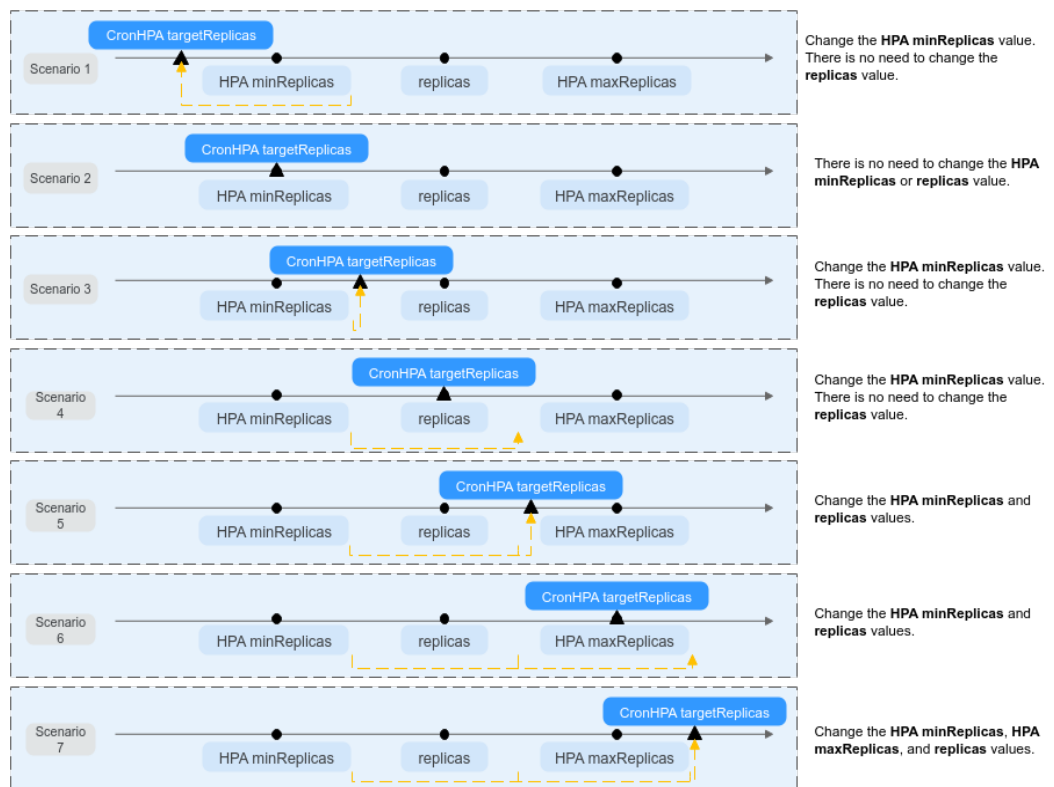
When CronHPA and HPA are used together, CronHPA rules take effect based on the HPA policy. CronHPA uses HPA to perform operations on the Deployment.

Understanding the following parameters can better understand the working rules of the CronHPA.

- **targetReplicas**: Number of pods set for CronHPA. When CronHPA takes effect, this parameter adjusts the maximum or minimum number of pods in HPA policies to adjust the number of Deployment pods.
- **minReplicas**: Minimum number of Deployment pods.
- **maxReplicas**: Maximum number of Deployment pods.
- **replicas**: Number of pods in a Deployment before the CronHPA policy takes effect.

When the CronHPA rule takes effect, the maximum or minimum number of pods are adjusted by comparing the number of **targetReplicas** with the actual number of pods and combining the minimum or maximum number of pods in the HPA policy.

**Figure 10-3** CronHPA scaling scenarios



**Figure 10-3** shows possible scaling scenarios. The following examples detail how CronHPA modifies the number of pods in HPAs.

**Table 10-4** CronHPA scaling parameters

Scenario	Scenario Description	Scaling Condition			Result	Operation Description
		Target CronHPA Pods (targetReplicas)	Deployment Pods (replicas)	Upper and Lower Limits of HPA Pods (minReplicas / maxReplicas)		
1	<b>targetReplicas</b> < minReplicas ≤ replicas ≤ maxReplicas	4	5	5/10	HPA: 4/10 Deployment: 5	When the value of <b>targetReplicas</b> is smaller than that of <b>minReplicas</b> : <ul style="list-style-type: none"> <li>• Change the value of <b>minReplicas</b>.</li> <li>• The value of <b>replicas</b> requires no change.</li> </ul>
2	<b>targetReplicas</b> = minReplicas ≤ replicas ≤ maxReplicas	5	6	5/10	HPA: 5/10 Deployment: 6	When the value of <b>targetReplicas</b> is equal to that of <b>minReplicas</b> : <ul style="list-style-type: none"> <li>• The value of <b>minReplicas</b> requires no change.</li> <li>• The value of <b>replicas</b> requires no change.</li> </ul>

Scenario	Scenario Description	Scaling Condition			Result	Operation Description
		Target CronHPA Pods (target Replicas)	Deployment Pods (replicas)	Upper and Lower Limits of HPA Pods (minReplicas / maxReplicas)		
3	minReplicas < <b>targetReplicas</b> < replicas ≤ maxReplicas	4	5	1/10	HPA: 4/10 Deployment: 5	When the value of <b>targetReplicas</b> is greater than that of <b>minReplicas</b> and smaller than that of <b>replicas</b> : <ul style="list-style-type: none"> <li>• Change the value of <b>minReplicas</b>.</li> <li>• The value of <b>replicas</b> requires no change.</li> </ul>
4	minReplicas < <b>targetReplicas</b> = replicas < maxReplicas	5	5	1/10	HPA: 5/10 Deployment: 5	When the value of <b>targetReplicas</b> is greater than that of <b>minReplicas</b> and equal to that of <b>replicas</b> : <ul style="list-style-type: none"> <li>• Change the value of <b>minReplicas</b>.</li> <li>• The value of <b>replicas</b> requires no change.</li> </ul>

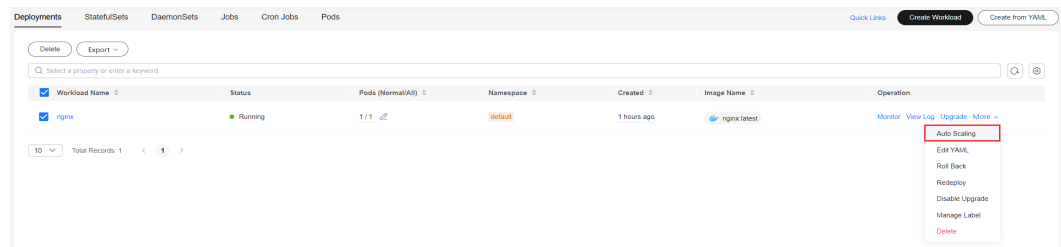
Scenario	Scenario Description	Scaling Condition			Result	Operation Description
		Target CronHPA Pods (targetReplicas)	Deployment Pods (replicas)	Upper and Lower Limits of HPA Pods (minReplicas / maxReplicas)		
5	$\text{minReplicas} \leq \text{replicas} < \text{targetReplicas}$ $< \text{maxReplicas}$	6	5	1/10	HPA: 6/10 Deployment: 6	When the value of <b>targetReplicas</b> is greater than that of <b>replicas</b> and less than that of <b>maxReplicas</b> : <ul style="list-style-type: none"> <li>• Change the value of <b>minReplicas</b>.</li> <li>• Change the value of <b>replicas</b>.</li> </ul>
6	$\text{minReplicas} \leq \text{replicas} < \text{targetReplicas}$ $= \text{maxReplicas}$	10	5	1/10	HPA: 10/10 Deployment: 10	When the value of <b>targetReplicas</b> is greater than that of <b>replicas</b> and equal to that of <b>maxReplicas</b> : <ul style="list-style-type: none"> <li>• Change the value of <b>minReplicas</b>.</li> <li>• Change the value of <b>replicas</b>.</li> </ul>

Scenario	Scenario Description	Scaling Condition			Result	Operation Description
		Target CronHPA Pods (target Replicas)	Deployment Pods (replicas)	Upper and Lower Limits of HPA Pods (minReplicas / maxReplicas)		
7	$\text{minReplicas} \leq \text{replicas} \leq \text{maxReplicas} < \text{targetReplicas}$	11	5	5/10	HPA: 11/11 Deployment: 11	When the value of <b>targetReplicas</b> is greater than that of <b>maxReplicas</b> : <ul style="list-style-type: none"> <li>• Change the value of <b>minReplicas</b>.</li> <li>• Change the value of <b>maxReplicas</b>.</li> <li>• Change the value of <b>replicas</b>.</li> </ul>

### Using the CCE console

- Step 1** Log in to the CCE console and click the cluster name to access the cluster console.
- Step 2** Choose **Workloads** in the navigation pane. Locate the target workload and choose **More > Auto Scaling** in the **Operation** column.

**Figure 10-4** Scaling a workload



- Step 3** Set **Policy Type** to **HPA+CronHPA** and enable HPA and CronHPA policies.

CronHPA periodically adjusts the maximum and minimum numbers of pods using the HPA policy.

**Step 4** Configure the HPA policy. For details, see [Creating an HPA Policy](#).

**Figure 10-5** Enabling the HPA policy

The screenshot shows the 'Auto Scaling' configuration page. At the top right is a close button (X). Under 'Policy Type', there are two tabs: 'HPA' (selected and highlighted in blue) and 'Customized HPA Policies'. Below this is a note: 'Configure either HPA+CronHPA or CustomizedHPA policies. You are advised to configure HPA+CronHPA policies.' Under 'HPA Policies', there is a sub-section 'HPA Policies' with a description: 'Workload pods are scaled when system metrics (CPU usage and memory usage) and custom Prometheus metrics meet specified requirements. [HPA Policy](#)'. Below this is a 'Policy' toggle switch, which is turned on. Under 'Pod Range', there are two input fields: '1' and '10', with a minus sign between them. A note says: 'The workload pods are scaled within this range when the policy is triggered.' Under 'Scaling Behavior', there are two tabs: 'Default' (selected and highlighted in blue) and 'Custom'. Below this is a note: 'Default: Scales workloads using the Kubernetes default behavior. Custom: Scales workloads using custom policies such as the stabilization window, steps, and priorities. Unspecified parameters use the values recommended by Kubernetes. [Kubernetes Default Behavior](#)'. Under 'System Policy', there is a warning message: 'System metrics cannot be collected because no CPU or memory quota is configured for the associated workload reviews-v2. The configured system policies will not take effect.' Below this is a table for system metrics:

Metric	Desired Value	Tolerance Range	Operation
CPU usage	70 %	60 % - 80 %	Delete

Below the system metrics table is a dashed box with a plus sign (+) for adding more metrics. Under 'Custom Policy', there is another table for custom metrics:

Metric Name	Metric Source	Desired Value	Tolerance Range	Operation
+				


**Table 10-5** HPA policy

Parameter	Description
Pod Range	Minimum and maximum numbers of pods. When a policy is triggered, the workload pods are scaled within this range.
Cooldown Period	Interval between a scale-in and a scale-out. The unit is minute. <b>The interval cannot be shorter than 1 minute.</b> <b>This parameter is supported only in clusters of v1.15 to v1.23.</b> This parameter indicates the interval between consecutive scaling operations. The cooldown period ensures that a scaling operation is initiated only when the previous one is completed and the system is running stably.

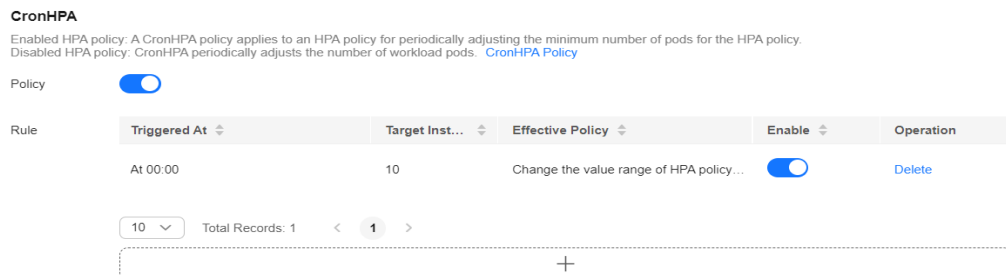


Parameter	Description
Scaling Behavior	<p><b>This parameter is supported only in clusters of v1.25 or later.</b></p> <ul style="list-style-type: none"> <li>● <b>Default:</b> scales workloads using the Kubernetes default behavior. For details, see <a href="#">Default Behavior</a>.</li> <li>● <b>Custom:</b> scales workloads using custom policies such as stabilization window, steps, and priorities. Unspecified parameters use the values recommended by Kubernetes. <ul style="list-style-type: none"> <li>– <b>Disable scale-out/scale-in:</b> Select whether to disable scale-out or scale-in.</li> <li>– <b>Stabilization Window:</b> a period during which CCE continuously checks whether the metrics used for scaling keep fluctuating. CCE triggers scaling if the desired state is not maintained for the entire window. This window restricts the unwanted flapping of pod count due to metric changes.</li> <li>– <b>Step:</b> specifies the scaling step. You can set the number or percentage of pods to be scaled in or out within a specified period. If there are multiple policies, you can select the policy that maximizes or minimizes the number of pods.</li> </ul> </li> </ul>
System Policy	<ul style="list-style-type: none"> <li>● <b>Metric:</b> You can select <b>CPU usage</b> or <b>Memory usage</b>.  <b>NOTE</b>  Usage = Average resource usage of all pods in a workload/  Requested resources</li> <li>● <b>Desired Value:</b> Enter the desired average resource usage. This parameter indicates the desired value of the selected metric. Number of target pods (rounded up) = (Current metric value/Desired value) x Number of current pods  <b>NOTE</b>  When calculating the number of pods to be added or reduced, the HPA policy uses the maximum number of pods in the last 5 minutes.</li> <li>● <b>Tolerance Range:</b> Scaling is not triggered when the metric value is within the tolerance range. The desired value must be within the tolerance range.  If the metric value is greater than the scale-in threshold and less than the scale-out threshold, no scaling is triggered.  <b>This parameter is supported only in clusters of v1.15 or later.</b></li> </ul>

Parameter	Description
Custom Policy (supported only in clusters of v1.15 or later)	<p><b>NOTE</b> Before creating a custom policy, install an add-on that supports custom metric collection (for example, Prometheus) in the cluster. Ensure that the add-on can collect and report the custom metrics of the workloads. For details, see <a href="#">Monitoring Custom Metrics Using Cloud Native Cluster Monitoring</a>.</p> <ul style="list-style-type: none"> <li>• <b>Metric Name:</b> name of the custom metric. You can select a name as prompted.</li> <li>• <b>Metric Source:</b> Select an object type from the drop-down list. You can select <b>Pod</b>.</li> <li>• <b>Desired Value:</b> the average metric value of all pods. Number of pods to be scaled (rounded up) = (Current metric value/ Desired value) x Number of current pods</li> </ul> <p><b>NOTE</b> When calculating the number of pods to be added or reduced, the HPA policy uses the maximum number of pods in the last 5 minutes.</p> <ul style="list-style-type: none"> <li>• <b>Tolerance Range:</b> Scaling is not triggered when the metric value is within the tolerance range. The desired value must be within the tolerance range.</li> </ul>

**Step 5** Click  in the CronHPA policy rule. In the dialog box displayed, configure scaling policy parameters.

**Figure 10-6** Enabling the CronHPA policy



**Table 10-6** CronHPA policy parameters

Parameter	Description
Target Instances	When the policy is triggered, CCE will adjust the number of HPA policy pods based on service requirements. For details, see <a href="#">Table 10-4</a> .
Trigger Time	You can select a specific time every day, every week, every month, or every year. <b>NOTE</b> This time indicates the local time of where the node is deployed.
Enable	Enable or disable the policy rule.

**Step 6** After configuring the preceding parameters, click **OK**. Then, the added policy rule is displayed in the rule list. Repeat the preceding steps to add multiple policy rules, but the triggering time of the policies must be different.

**Step 7** Click **Create**.

----End

### Using kubectl

When the CronHPA is compatible with the HPA policy, the **scaleTargetRef** field in CronHPA must be set to the HPA policy, and the **scaleTargetRef** field in the HPA policy must be set to Deployment. In this way, CronHPA adjusts the maximum and minimum numbers of pods in the HPA policy at a fixed time and the scheduled scaling is compatible with the auto scaling.

**Step 1** Create an HPA policy for the Deployment.

```
apiVersion: autoscaling/v1
kind: HorizontalPodAutoscaler
metadata:
  name: hpa-test
  namespace: default
spec:
  maxReplicas: 10          # Maximum number of pods
  minReplicas: 5          # Minimum number of pods
  scaleTargetRef:         # Associate a Deployment.
    apiVersion: apps/v1
    kind: Deployment
    name: nginx
  targetCPUUtilizationPercentage: 50
```

**Step 2** Create a CronHPA policy and associate it with the HPA policy created in [Step 1](#).

```
apiVersion: autoscaling.cce.io/v2alpha1
kind: CronHorizontalPodAutoscaler
metadata:
  name: cctest
  namespace: default
spec:
  scaleTargetRef:         # Associate an HPA policy.
    apiVersion: autoscaling/v1
    kind: HorizontalPodAutoscaler
    name: hpa-test
  rules:
  - ruleName: "scale-down"
    schedule: "15 * * * *" # Running time and period of a job. For details, see Cron, for example, 0 * * *
    * or @hourly.
    targetReplicas: 1      # Number of target pods
    disable: false
  - ruleName: "scale-up"
    schedule: "13 * * * *"
    targetReplicas: 11
    disable: false
```

**Table 10-7** Key fields of CronHPA

Field	Description
apiVersion	API version. The value is fixed at <b>autoscaling.cce.io/v2alpha1</b> .

Field	Description
kind	API type. The value is fixed at <b>CronHorizontalPodAutoscaler</b> .
metadata.name	Name of a CronHPA policy.
metadata.namespace	Namespace to which the CronHPA policy belongs.
spec.scaleTargetRef	Specifies the scaling object of CronHPA. The following fields can be configured: <ul style="list-style-type: none"> <li>• <b>apiVersion</b>: API version of the CronHPA scaling object.</li> <li>• <b>kind</b>: API type of the CronHPA scaling object.</li> <li>• <b>name</b>: Name of the CronHPA scaling object.</li> </ul> CronHPA supports HPA policies or Deployments. For details, see <a href="#">Using CronHPA to Adjust the HPA Scaling Scope</a> or <a href="#">Using CronHPA to Directly Adjust the Number of Deployment Pods</a> .
spec.rules	CronHPA policy rule. Multiple rules can be added. The following fields can be configured for each rule: <ul style="list-style-type: none"> <li>• <b>ruleName</b>: CronHPA rule name, which must be unique.</li> <li>• <b>schedule</b>: Running time and period of a job. For details, see <a href="#">Cron</a>, for example, <code>0 * * * *</code> or <code>@hourly</code>.</li> </ul> <p><b>NOTE</b> This time indicates the local time of where the node is deployed.</p> <ul style="list-style-type: none"> <li>• <b>targetReplicas</b>: indicates the number of pods to be scaled in or out.</li> <li>• <b>disable</b>: The value can be <b>true</b> or <b>false</b>. <b>false</b> indicates that the rule takes effect, and <b>true</b> indicates that the rule does not take effect.</li> </ul>

----End

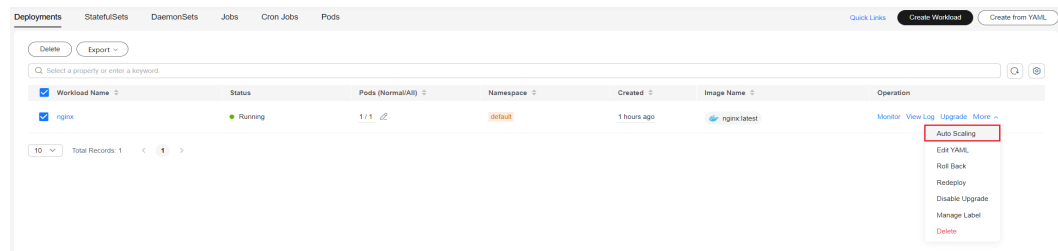
## Using CronHPA to Directly Adjust the Number of Deployment Pods

CronHPA adjusts associated Deployments separately to periodically adjust the number of Deployment pods. The method is as follows:

### Using the CCE console

- Step 1** Log in to the CCE console and click the cluster name to access the cluster console.
- Step 2** Choose **Workloads** in the navigation pane. Locate the target workload and choose **More > Auto Scaling** in the **Operation** column.

**Figure 10-7** Scaling a workload

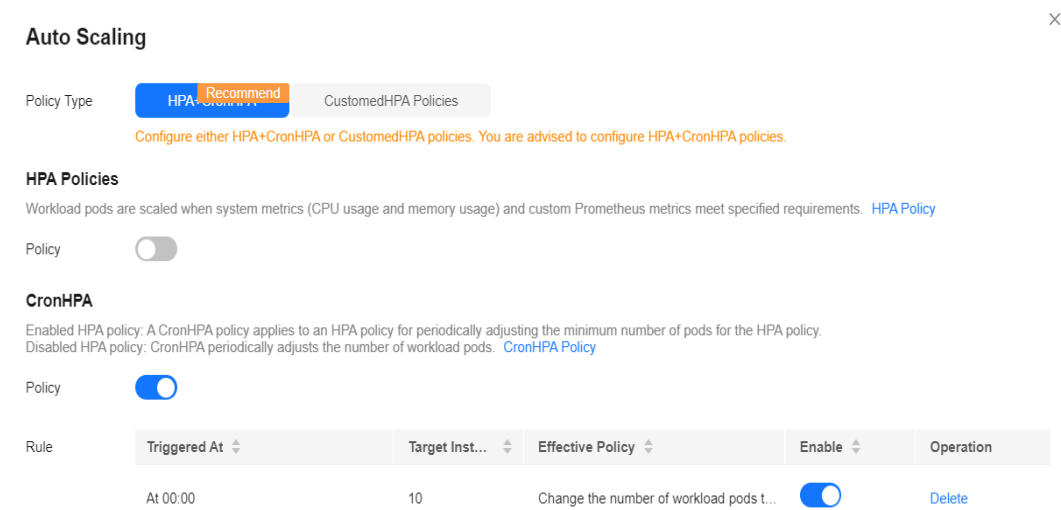


**Step 3** Set **Policy Type** to **HPA+CronHPA**, disable HPA, and enable CronHPA.

CronHPA periodically adjusts the number of workload pods.

**Step 4** Click **+** in the CronHPA policy rule. In the dialog box displayed, configure scaling policy parameters.

**Figure 10-8** Using CronHPA to adjust the number of workload pods



**Table 10-8** CronHPA policy parameters

Parameter	Description
Target Instances	When a policy is triggered, the number of workload pods will be adjusted to the value of this parameter.
Trigger Time	You can select a specific time every day, every week, every month, or every year. <b>NOTE</b> This time indicates the local time of where the node is deployed.
Enable	Enable or disable the policy rule.

**Step 5** After configuring the preceding parameters, click **OK**. Then, the added policy rule is displayed in the rule list. Repeat the preceding steps to add multiple policy rules, but the triggering time of the policies must be different.

**Step 6 Click Create.**

----End

**Using kubectl**

```
apiVersion: autoscaling.cce.io/v2alpha1
kind: CronHorizontalPodAutoscaler
metadata:
  name: cctest
  namespace: default
spec:
  scaleTargetRef:      # Associate a Deployment.
    apiVersion: apps/v1
    kind: Deployment
    name: nginx
  rules:
  - ruleName: "scale-down"
    schedule: "08 * * * *" # Running time and period of a job. For details, see Cron, for example, 0 * * * * or
    @hourly.
    targetReplicas: 1
    disable: false
  - ruleName: "scale-up"
    schedule: "05 * * * *"
    targetReplicas: 3
    disable: false
```

## 10.2.4 Creating a CustomedHPA Policy

A CustomedHPA policy scales Deployments based on metrics (such as CPU usage and memory usage) or at a periodic interval (a specific time point every day, every week, every month, or every year). This type of policy is a CCE-enhanced auto scaling capability.

Supported functions:

- Scaling can be performed based on the percentage of the current number of pods.
- The minimum scaling step can be set.
- Different scaling operations can be performed based on the actual metric values.

### Prerequisites

The [CCE Advanced HPA](#) add-on must be installed. If the add-on version is earlier than 1.2.11, [Prometheus](#) must be installed. If the add-on version is 1.2.11 or later, one of the following add-ons that can provide metrics APIs must be installed based on your cluster version and service requirements:

- [Kubernetes Metrics Server](#): provides basic resource usage metrics, such as container CPU and memory usage. It is supported by all cluster versions.
- [Cloud Native Cluster Monitoring](#): available only in clusters of v1.17 or later.
  - Auto scaling based on basic resource metrics: Prometheus needs to be registered as a metrics API. For details, see [Providing Resource Metrics Through the Metrics API](#).
  - Auto scaling based on custom metrics: Custom metrics need to be aggregated to the Kubernetes API server. For details, see [Creating an HPA Policy Using Custom Metrics](#).

- **Prometheus** : Prometheus needs to be registered as a metrics API. For details, see [Providing Resource Metrics Through the Metrics API](#). This add-on supports only clusters of v1.21 or earlier.

## Notes and Constraints


- CustomedHPA policies apply only to clusters of v1.15 or later.
- For clusters earlier than v1.19.10, if an HPA policy is used to scale out a workload with EVS volumes mounted, the existing pods cannot be read or written when a new pod is scheduled to another node.  
For clusters of v1.19.10 and later, if an HPA policy is used to scale out a workload with EVS volumes mounted, a new pod cannot be started because EVS disks cannot be attached.
- The specifications of the CCE Advanced HPA add-on are decided based on the total number of containers in the cluster and the number of scaling policies. Configure 500m CPU cores and 1000 MiB memory for every 5000 containers, and 100m CPU cores and 500 MiB memory for every 1000 scaling policies.
- After a CustomedHPA policy is created, the type of its associated workload cannot be changed.

## Procedure

- Step 1** Log in to the CCE console and click the cluster name to access the cluster console.
- Step 2** Choose **Workloads** in the navigation pane. Locate the target workload and choose **More > Auto Scaling** in the **Operation** column.
- Step 3** Set **Policy Type** to **CustomedHPA** and configure policy parameters.

**Table 10-9** CustomedHPA policy parameters

Parameter	Description
Pod Range	Minimum and maximum numbers of pods. When a policy is triggered, the workload pods are scaled within this range.
Cooldown Period	Enter an interval, in minutes. This parameter indicates the interval between consecutive scaling operations. The cooldown period ensures that a scaling operation is initiated only when the previous one is completed and the system is running stably. <b>NOTE</b> If you are using CCE Cluster Autoscaler version 1.3.10 or later, the cooldown period will only apply to metric policies. Periodic policies will not be affected by the cooldown period.

Parameter	Description
Rules	<p>Click . In the dialog box displayed, set the following parameters:</p> <ul style="list-style-type: none"> <li>• <b>Type:</b> You can select <b>Metric-based</b> (Table 10-10) or <b>Periodic</b> (Table 10-11). Then, configure trigger conditions and actions.</li> <li>• <b>Enable:</b> Enable or disable the policy rule.</li> </ul> <p>After configuring the preceding parameters, click <b>OK</b>. Then, the added policy rule is displayed in the rule list.</p>

**Table 10-10** Metric-based rules

Parameter	Description
Trigger	<p>Select <b>CPU usage</b> or <b>Memory usage</b>, choose &gt; or &lt;, and enter a percentage.</p> <p><b>NOTE</b> Usage = Average resource usage of all pods in a workload/Requested resources</p>
Action	<p>Set an action to be performed when the trigger condition is met. Multiple actions can be added.</p> <ul style="list-style-type: none"> <li>• <b>Scale To:</b> Adjust the number of pods to the specified value. Both a number and a percentage will do. This action can be used to scale in or out pods. If the current number of pods is less than the target value or the target percentage is greater than 100%, the number of pods will be scaled out to the target value. If the current number of pods is greater than the target value or the target percentage is less than 100%, the number of pods will be scaled in to the target value.</li> <li>• <b>Add:</b> Configure this parameter when <b>Trigger</b> is set to &gt;. Add the number of pods. You can specify a number or a percentage. This action can only be used to scale out pods.</li> <li>• <b>Reduce:</b> Configure this parameter when <b>Trigger</b> is set to &lt;. Reduce the number of pods. You can specify a number or a percentage. This action can only be used to scale in pods.</li> </ul> <p><b>NOTE</b> You can enter a number or a percentage for the preceding actions. When entering a percentage, you are required to specify the minimum number of available pods. Final number of pods = Number of current pods x Percentage. The result is rounded up. If the result is smaller than the minimum number of available pods, the preset value is used. Otherwise, the calculation result is used.</p>



**Table 10-11** Periodic-based rules

Parameter	Description
Trigger Time	You can select a specific time every day, every week, every month, or every year.
Action	<p>Set an action to be performed at the <b>Triggered Time</b>.</p> <ul style="list-style-type: none"> <li>• <b>Scale To:</b> Adjust the number of pods to the specified value. Both a number and a percentage will do. This action can be used to scale in or out pods. If the current number of pods is less than the target value or the target percentage is greater than 100%, the number of pods will be scaled out to the target value. If the current number of pods is greater than the target value or the target percentage is less than 100%, the number of pods will be scaled in to the target value.</li> <li>• <b>Add:</b> Add the number of pods. You can specify a number or a percentage. This action can only be used to scale out pods.</li> <li>• <b>Reduce:</b> Reduce the number of pods. You can specify a number or a percentage. This action can only be used to scale in pods.</li> </ul> <p><b>NOTE</b> You can enter a number or a percentage for the preceding actions. When entering a percentage, you are required to specify the minimum number of available pods. Final number of pods = Number of current pods x Percentage. The result is rounded up. If the result is smaller than the minimum number of available pods, the preset value is used. Otherwise, the calculation result is used.</p>

**Step 4** Click **Create**.

----End

## 10.2.5 Managing Workload Scaling Policies

### Scenario

After a workload scaling policy is created, you can update and delete the policy, as well as edit the YAML file.

### Procedure

You can view the rules, latest status, and events of a workload scaling policy and handle exceptions based on the error information displayed.

**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 **Policies**. On the **Scaling Policies** tab page, click the **HPA Policies/CronHPA Policies** tab based on the scaling policy type.

**Step 3** Check the latest status, rules, and associated workloads of a scaling policy.

 **NOTE**

You can also check a created scaling policy on the workload details page.

1. Log in to the CCE console and click the cluster name to access the cluster console.
2. In the navigation pane, choose **Workloads**. Click the workload name to view its details.
3. On the workload details page, switch to the **Auto Scaling** tab page to obtain the scaling policies. You can also obtain the scaling policies you configured on the **Policies** page.

**Step 4** Manage scaling policies.

Scaling Policy Type	Operation
HPA	<ul style="list-style-type: none"> <li>• <b>View Events:</b> Check HPA policy events. If an error occurred, locate and rectify the fault based on the error message displayed on the page.</li> <li>• <b>Edit YAML:</b> In the dialog box displayed, edit, copy, or download the YAML file.</li> <li>• <b>Edit:</b> On the <b>Edit HPA Policy</b> page, configure policy parameters listed in <a href="#">Table 10-3</a>.</li> <li>• <b>Clone:</b> Duplicate an existing auto scaling policy and modify the parameter settings as required.</li> <li>• <b>Delete:</b> In the dialog box displayed, click <b>Yes</b>.</li> </ul>
CronHPA	<ul style="list-style-type: none"> <li>• <b>View YAML:</b> In the dialog box displayed, copy or download the YAML file but you are not allowed to modify it.</li> <li>• <b>Delete:</b> In the dialog box displayed, click <b>Yes</b>.</li> </ul>

----End

## 10.3 Scaling a Node

### 10.3.1 Node Scaling Rules

HPA is designed for pod-level scaling and can dynamically adjust the number of replicas based on workload metrics. However, if cluster resources are insufficient and new replicas cannot run, you can only scale out the cluster.

**CCE Cluster Autoscaler** is a node scaling component provided by Kubernetes. It 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.

#### Prerequisites

Before using the node scaling function, you must install the **CCE Cluster Autoscaler** add-on of v1.13.8 or later in the cluster.

## How Cluster Autoscaler Works

**Cluster Autoscaler** goes through two processes.

- **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.

### NOTE

When Autoscaler checks unscheduled pods for scale outs, it uses the scheduling algorithm consistent with the Kubernetes community version for simulated scheduling calculation. If non-built-in kube-schedulers or other non-Kubernetes community scheduling policies are used for application scheduling, when Autoscaler is used to expand the capacity for such applications, the capacity may fail to be expanded or may be expanded more than expected due to inconsistent scheduling algorithms.

- **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.

When a cluster node is idle for a period of time (10 minutes by default), cluster scale-in is triggered, and the node is automatically deleted. However, a node cannot be deleted from a cluster if the following pods exist:

- Pods that do not meet specific requirements set in Pod Disruption Budgets (**PodDisruptionBudget**)
- Pods that cannot be scheduled to other nodes due to constraints such as affinity and anti-affinity policies
- Pods that have the **cluster-autoscaler.kubernetes.io/safe-to-evict: 'false'** annotation
- Pods (except those created by DaemonSets in the kube-system namespace) that exist in the kube-system namespace on the node
- Pods that are not created by the controller (Deployment/ReplicaSet/job/StatefulSet)

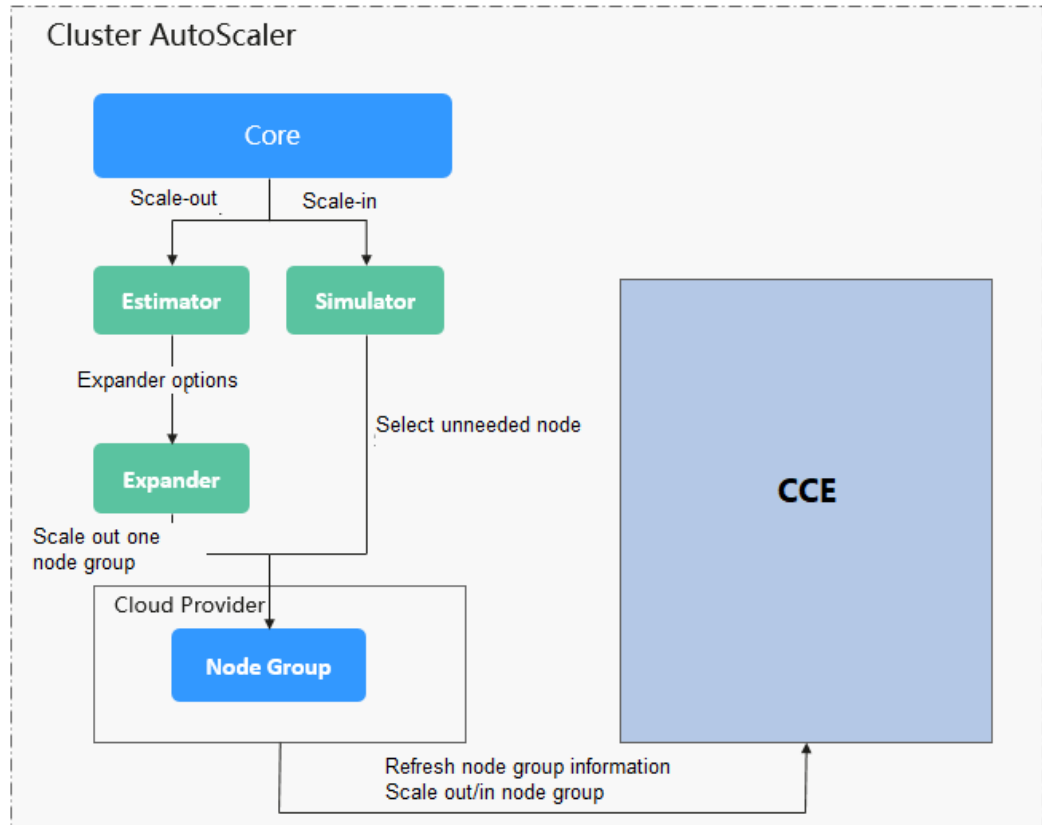
### NOTE

When a node meets the scale-in conditions, Autoscaler adds the **DeletionCandidateOfClusterAutoscaler** taint to the node in advance to prevent pods from being scheduled to the node. After the Autoscaler add-on is uninstalled, if the taint still exists on the node, manually delete it.

## Cluster Autoscaler Architecture

**Figure 10-9** shows the Cluster Autoscaler architecture and its core modules.

Figure 10-9 Cluster Autoscaler architecture



### Description

- **Estimator:** Evaluates the number of nodes to be added to each node pool to host unschedulable pods.
- **Simulator:** Finds the nodes that meet the scale-in conditions in the scale-in scenario.
- **Expander:** Selects an optimal node from the node pool picked out by the Estimator based on the user-defined policy in the scale-out scenario. [Table 10-12](#) lists the Expander policies.

**Table 10-12 Expander policies supported by CCE**

Policy	Description	Application Scenario	Example
Random	Randomly selects a schedulable node pool to perform the scale-out.	This policy is typically used as a basic backup for other complex policies. Only use this policy if the other policies cannot be used.	<p>Assume that auto scaling is enabled for node pools 1 and 2 in the cluster and the scale-out upper limit is not reached. The policy for scaling out the number of pods for a workload is as follows:</p> <ol style="list-style-type: none"> <li>1. Pending pods trigger the Autoscaler to determine the scale-out process.</li> <li>2. Autoscaler simulates the scheduling phase and evaluates that some pending pods can be scheduled to the added nodes in both node pools 1 and 2.</li> <li>3. Autoscaler randomly selects node pool 1 or node pool 2 for scale-out.</li> </ol>

Policy	Description	Application Scenario	Example
most - pods	<p>A combined policy. It takes precedence over the random policy.</p> <p>Preferentially selects the node pool that can schedule the most pods after scale-out. If multiple node pools meet the condition, the random policy is used for further decision-making.</p>	<p>This policy is based on the maximum number of pods that can be scheduled.</p>	<p>Assume that auto scaling is enabled for node pools 1 and 2 in the cluster and the scale-out upper limit is not reached. The policy for scaling out the number of pods for a workload is as follows:</p> <ol style="list-style-type: none"> <li>1. Pending pods trigger the Autoscaler to determine the scale-out process.</li> <li>2. Autoscaler simulates the scheduling phase and evaluates that some pending pods can be scheduled to the added nodes in both node pools 1 and 2.</li> <li>3. Autoscaler evaluates that node pool 1 can schedule 20 new pods and node pool 2 can schedule only 10 new pods after scale-out. Therefore, Autoscaler selects node pool 1 for scale-out.</li> </ol>

Policy	Description	Application Scenario	Example
least-waste	<p>A combined policy. It takes precedence over the random policy.</p> <p>Autoscaler evaluates the overall CPU or memory allocation rate of the node pools and selects the node pool with the minimum CPU or memory waste. If multiple node pools meet the condition, the random policy is used for further decision-making.</p>	<p>This policy uses the minimum waste score of CPU or memory resources as the selection criteria.</p> <p>The formula for calculating the minimum waste score (wastedScore) is as follows:</p> <ul style="list-style-type: none"> <li>• <math>wastedCPU = (Total\ number\ of\ CPUs\ of\ the\ nodes\ to\ be\ scaled\ out - Total\ number\ of\ CPUs\ of\ the\ pods\ to\ be\ scheduled) / Total\ number\ of\ CPUs\ of\ the\ nodes\ to\ be\ scaled\ out</math></li> <li>• <math>wastedMemory = (Total\ memory\ size\ of\ nodes\ to\ be\ scaled\ out - Total\ memory\ size\ of\ pods\ to\ be\ scheduled) / Total\ memory\ size\ of\ nodes\ to\ be\ scaled\ out</math></li> <li>• <math>wastedScore = wastedCPU + wastedMemory</math></li> </ul>	<p>Assume that auto scaling is enabled for node pools 1 and 2 in the cluster and the scale-out upper limit is not reached. The policy for scaling out the number of pods for a workload is as follows:</p> <ol style="list-style-type: none"> <li>1. Pending pods trigger the Autoscaler to determine the scale-out process.</li> <li>2. Autoscaler simulates the scheduling phase and evaluates that some pending pods can be scheduled to the added nodes in both node pools 1 and 2.</li> <li>3. Autoscaler evaluates that the minimum waste score of node pool 1 after scale-out is smaller than that of node pool 2. Therefore, Autoscaler selects node pool 1 for scale-out.</li> </ol>

Policy	Description	Application Scenario	Example
priority	<p>A combined policy. The priorities for the policies are as follows: priority &gt; least-waste &gt; random.</p> <p>It is an enhanced least-waste policy configured based on the node pool or scaling group priority. If multiple node pools meet the condition, the least-waste policy is used for further decision-making.</p>	<p>This policy allows you to configure the priorities of node pools or scaling groups through the console or API, while the least-waste policy can effectively minimize resource waste in various scenarios. The priority policy is used as the <b>default preferred policy</b> thanks to its good universality.</p>	<p>Assume that auto scaling is enabled for node pools 1 and 2 in the cluster and the scale-out upper limit is not reached. The policy for scaling out the number of pods for a workload is as follows:</p> <ol style="list-style-type: none"> <li>1. Pending pods trigger the Autoscaler to determine the scale-out process.</li> <li>2. Autoscaler simulates the scheduling phase and evaluates that some pending pods can be scheduled to the added nodes in both node pools 1 and 2.</li> <li>3. Autoscaler evaluates that node pool 1 has a higher priority than node pool 2. Therefore, Autoscaler selects node pool 1 for scale-out.</li> </ol>



Policy	Description	Application Scenario	Example
priority-ratio	<p>A combined policy. The priorities for the policies are as follows: priority &gt; priority-ratio &gt; least-waste &gt; random.</p> <p>If there are multiple node pools with the same priority, evaluate the CPU to memory ratios for the nodes in the cluster. Then compare that ratio, for what was allocated to what had been requested. Finally, you should preferentially select the node pools where these two ratios are the closest.</p>	<p>This policy is used for rescheduling global resources for pods or nodes (instead of only adding nodes) to reduce the overall resource fragmentation rate of the cluster. Use this policy only in rescheduling scenarios.</p>	<p>Assume that auto scaling is enabled for node pools 1 and 2 in the cluster and the scale-out upper limit is not reached. The policy for scaling out the number of pods for a workload is as follows:</p> <ol style="list-style-type: none"> <li>1. Pending pods trigger the Autoscaler to determine the scale-out process.</li> <li>2. Autoscaler simulates the scheduling phase and evaluates that some pending pods can be scheduled to the added nodes in both node pools 1 and 2.</li> <li>3. Autoscaler determines a preferentially selected node pool and evaluates that the CPU/memory ratio of pods is 1:4. The node flavor in node pool 1 is 2 vCPUs and 8 GiB of memory (the CPU/memory ratio is 1:4), and the node flavor in node pool 2 is 2 vCPUs and 4 GiB of memory (the CPU/memory ratio is 1:2). Therefore, node pool 1 is preferred for this scale-out.</li> </ol>

### 10.3.2 Creating a Node Scaling Policy

CCE provides auto scaling through the **CCE Cluster Autoscaler** add-on. Nodes with different flavors can be automatically added across AZs on demand.

If both a node scaling policy and the configuration in the Autoscaler add-on take effect, for example, there are pods that cannot be scheduled and the value of a

metric reaches the threshold, scale-out is performed first for the unschedulable pods.

- If the scale-out succeeds for the unschedulable pods, the system skips the metric-based rule logic and enters the next loop.
- If the scale-out fails for the unschedulable pods, the metric-based rule is executed.

## Prerequisites

Before using the node scaling function, you must install the [CCE Cluster Autoscaler](#) add-on of v1.13.8 or later in the cluster.

## Notes and Constraints

- If there are no nodes in a node pool, Autoscaler cannot obtain the CPU or memory data of the node, and the node scaling rule triggered using these metrics will not take effect.
- If the driver of a GPU or NPU node is not installed, Autoscaler determines that the node is not fully available and the node scaling rules triggered using the CPU or memory metrics will not take effect.
- When Autoscaler is used, some taints or annotations may affect auto scaling. Therefore, do not use the following taints or annotations in clusters:
  - **ignore-taint.cluster-autoscaler.kubernetes.io**: The taint works on nodes. Kubernetes-native Autoscaler supports protection against abnormal scale outs and periodically evaluates the proportion of available nodes in the cluster. When the proportion of non-ready nodes exceeds 45%, protection will be triggered. In this case, all nodes with the **ignore-taint.cluster-autoscaler.kubernetes.io** taint in the cluster are filtered out from the Autoscaler template and recorded as non-ready nodes, which affects cluster scaling.
  - **cluster-autoscaler.kubernetes.io/enable-ds-eviction**: The annotation works on pods, which determines whether DaemonSet pods can be evicted by Autoscaler. For details, see [Well-Known Labels, Annotations and Taints](#).

## Configuring Node Pool Scaling Policies

**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 **Nodes**. On the **Node Pools** tab, locate the row containing the target node pool and click **Auto Scaling**.

- If Autoscaler has not been installed, configure add-on parameters based on service requirements, click **Install**, and wait until the add-on is installed. For details about add-on configurations, see [CCE Cluster Autoscaler](#).
- If Autoscaler has been installed, directly configure scaling policies.

**Step 3** Configure auto scaling policies.

### AS Configuration

- **Customized Rule**: Click **Add Rule**. In the dialog box displayed, configure parameters. You can add multiple node scaling policies, a maximum of one

CPU usage-based rule, and one memory usage-based rule. The total number of rules cannot exceed 10.

The following table lists custom rules.

**Table 10-13** Custom rules

Rule Type	Configuration
Metric-based	<p>– <b>Trigger:</b> Select <b>CPU allocation rate</b> or <b>Memory allocation rate</b> and enter a value. The percentage must be greater than the value specified in the node resource requirements for a node scale-in when you configure a scaling policy (<a href="#">Configuring an Auto Scaling Policy for a Cluster</a>).</p> <p><b>NOTE</b></p> <ul style="list-style-type: none"> <li>▪ Resource allocation (%) = Resources requested by pods in the node pool/Resources allocatable to pods in the node pool</li> <li>▪ <b>If multiple rules meet the conditions, the rules are executed in either of the following modes:</b> If rules based on the <b>CPU allocation rate</b> and <b>memory allocation rate</b> are configured and two or more rules meet the scale-out conditions, the rule that will add the most nodes will be executed. If a rule is configured based on the <b>CPU allocation rate</b> and a <b>periodic rule</b> and both the rules meet the scale-out conditions, the periodic rule executed early changes the node pool to the scaling state. As a result, the metric-based rule cannot be executed. After the periodic rule is executed and the node pool status becomes normal, the metric-based rule will not be executed. If the metric-based rule is executed early, the periodic rule will be executed after the metric-based rule is executed.</li> <li>▪ If a rule is configured based on the <b>CPU allocation rate</b> and <b>memory allocation rate</b>, the policy detection period varies with the processing logic of each loop of the Autoscaler add-on. A scale-out is triggered once the conditions are met, but it is constrained by other factors such as the cooldown period and node pool status.</li> <li>▪ If the number of nodes reaches the upper limit of the cluster scale, <a href="#">the upper limit of the nodes supported in a node pool</a>, or <a href="#">the upper limit of the nodes of a specific flavor</a>, a metric-based scale-out will not be triggered.</li> <li>▪ If the number of nodes, CPUs, or memory resources reaches <a href="#">the upper limit for a node scale-out</a>, a metric-based scale-out will not be triggered.</li> </ul> <p>– <b>Action:</b> Configure an action to be performed when the triggering condition is met.</p> <ul style="list-style-type: none"> <li>▪ <b>Custom:</b> Add a specified number of nodes to a node pool.</li> <li>▪ <b>Auto calculation:</b> When the trigger condition is met, nodes are automatically added and the allocation rate is restored to a value lower than the threshold. The formula is as follows: Number of nodes to be added = [Resource request of pods in the node pool/(Available resources of a single node x Target allocation rate)] – Number of current nodes + 1</li> </ul>

Rule Type	Configuration
Periodic	<ul style="list-style-type: none"> <li>- <b>Trigger Time:</b> You can select a specific time every day, every week, every month, or every year.</li> <li>- <b>Action:</b> specifies an action to be carried out when the trigger time is reached. A specified number of nodes will be added to the node pool.</li> </ul>

- **Nodes:** 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.

### AS Object

- **Specifications:** Configure whether to enable auto scaling for node flavors in a node pool.

#### NOTE

If multiple flavors are configured for a node pool, you can specify the upper limit for the number of nodes and the priority for each flavor separately.

**Step 4** View cluster-level auto scaling configurations, which take effect for all node pools in the cluster. On this page, you can only view cluster-level auto scaling policies. To modify these policies, go to the **Settings** page. For details, see [Configuring an Auto Scaling Policy for a Cluster](#).

**Step 5** Click **OK**.

----End

## Configuring an Auto Scaling Policy for a Cluster

#### NOTE

An auto scaling policy takes effect on all node pools in a cluster. After the policy is modified, the Autoscaler add-on will be restarted.

**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 **Auto Scaling** tab.

- If Autoscaler has not been installed, configure add-on parameters based on service requirements, click **Install**, and wait until the add-on is installed. For details about add-on configurations, see [CCE Cluster Autoscaler](#).
- If Autoscaler has been installed, directly configure scaling policies.

**Step 3** Configure for an elastic scale-out.

- **Auto Scale-out when the load cannot be scheduled:** When workload pods in a cluster cannot be scheduled (pods remain in pending state), CCE automatically adds nodes to the slave node pool. If a pod has been scheduled to a node, the node will not be involved in an automatic scale-out. Such auto scaling typically works with an HPA policy. For details, see [Using HPA and CA for Auto Scaling of Workloads and Nodes](#).

If this function is not enabled, [custom scaling rules](#) are the only option for performing a scale-out.

- **Upper limit of resources to be expanded:** the upper limit for the cluster's resources, such as the number of nodes, CPU cores, and memory. Once this limit is reached, no new nodes will be automatically added.
- **Scale-Out Priority:** You can drag and drop the node pools in a list to adjust their scale-out priorities.

**Step 4** Configure for an elastic scale-in. Elastic scale-in is disabled by default. After it is enabled, the following configurations are supported:

**Node Scale-In Conditions:** Nodes in a cluster are automatically scaled in when the scale-in conditions are met.

- **Node Resource Condition:** When the requested cluster node resources (both CPU and memory) are lower than a certain percentage (50% by default) for a period of time (10 minutes by default), a cluster scale-in is triggered.
- **Node Status Condition:** If a node is unavailable for a specified period of time, the node will be automatically reclaimed. The default value is 20 minutes.
- **Scale-in Exception Scenarios:** When a node meets the following exception scenarios, CCE will not scale in the node even if the node resources or status meets scale-in conditions:
  - a. Resources on other nodes in the cluster are insufficient.
  - b. Scale-in protection is enabled on the node. To enable or disable node scale-in protection, choose **Nodes** in the navigation pane and then click the **Nodes** tab. Locate the target node, choose **More**, and then enable or disable node scale-in protection in the **Operation** column.
  - c. There is a pod with the non-scale label on the node.
  - d. Policies such as reliability have been configured on some containers on the node.
  - e. There are non-DaemonSet containers in the **kube-system** namespace on the node.
  - f. (Optional) A container managed by a third-party pod controller is running on a node. Third-party pod controllers are for custom workloads except Kubernetes-native workloads such as Deployments and StatefulSets. Such controllers can be created using [CustomResourceDefinitions](#).

#### Node Scale-in Policy

- **Number of Concurrent Scale-In Requests:** maximum number of idle nodes that can be concurrently deleted. Default value: 10.

Only idle nodes can be concurrently scaled in. Nodes that are not idle can only be scaled in one by one.

#### NOTE

During a node scale-in, if the pods on the node do not need to be evicted (such as DaemonSet pods), the node is idle. Otherwise, the node is not idle.

- **Node Recheck Timeout:** interval for rechecking a node that could not be removed. Default value: 5 minutes.
- **Cooldown Time**

- **Scale-in Cooldown Time After Node Deletion:** cooldown period for starting scale-in evaluation again after an auto scale-in is triggered in a cluster. Default value: 10 minutes.
- **Scale-in Cooldown Time After Scale-out:** cooldown period for starting scale-in evaluation again after an auto scale-out is triggered in a cluster. Default value: 10 minutes.

 **NOTE**

If both auto scale-out and scale-in exist in a cluster, set **Scale-in Cooldown Time After Scale-out** to 0 minutes. This prevents the node scale-in from being blocked due to continuous scale-out of some node pools or retries upon a scale-out failure, which results in unexpected waste of node resources.

- **Scale-in Cooldown Time After Failure:** cooldown period for starting scale-in evaluation again after an auto scale-in is failed in a cluster. Default value: 3 minutes. For details, see [Cooldown Period](#).

**Step 5** Click **Confirm configuration**.

----End

## Cooldown Period

The impact and relationship between the two cooldown periods configured for a node pool are as follows:

### Cooldown Period During a Scale-out

This interval indicates the period during which nodes added to the current node pool after a scale-out cannot be deleted. This setting takes effect in the entire node pool.

### Cooldown Period During a Scale-in

The interval after a scale-out indicates the period during which the entire cluster cannot be scaled in after the Autoscaler add-on triggers a scale-out (due to the unschedulable pods, metrics, and scaling policies). This interval takes effect in the entire cluster.

The interval after a node is deleted indicates the period during which the cluster cannot be scaled in after the Autoscaler add-on triggers a scale-in. This setting takes effect in the entire cluster.

The interval after a failed scale-in indicates the period during which the cluster cannot be scaled in after the Autoscaler add-on triggers a scale-in. This setting takes effect in the entire cluster.

## Period for Autoscaler to Retry a Scale-out

If a node pool failed to scale out, for example, due to insufficient quota, or an error occurred during node installation, Autoscaler can retry the scale-out in the node pool or switch to another node pool. The retry period varies depending on failure causes:

- When the user quota is insufficient, Autoscaler cools down the node pool for 5 minutes, 10 minutes, or 20 minutes. The maximum cooldown duration is 30 minutes. Then, Autoscaler switches to another node pool for a scale-out in

the next 10 seconds until the expected node is added or all node pools are cooled down.

- If an error occurred during node installation in a node pool, the node pool enters a 5-minute cooldown period. After the period expires, Autoscaler can trigger a node pool scale-out again. If the faulty node is automatically reclaimed, Cluster Autoscaler re-evaluates the cluster status within 1 minute and triggers a node pool scale-out as needed.
- During a node pool scale-out, if a node remains in the installing state for a long time, Cluster Autoscaler tolerates the node for a maximum of 15 minutes. After the tolerance period expires, Cluster Autoscaler re-evaluates the cluster status and triggers a node pool scale-out as needed.

## Example YAML

The following is a YAML example of a node scaling policy:

```
apiVersion: autoscaling.cce.io/v1alpha1
kind: HorizontalNodeAutoscaler
metadata:
  name: xxxx
  namespace: kube-system
spec:
  disable: false
  rules:
  - action:
    type: ScaleUp
    unit: Node
    value: 1
    cronTrigger:
      schedule: 47 20 * * *
    disable: false
    ruleName: cronrule
    type: Cron
  - action:
    type: ScaleUp
    unit: Node
    value: 2
    disable: false
    metricTrigger:
      metricName: Cpu
      metricOperation: '>'
      metricValue: "40"
      unit: Percent
    ruleName: metricrule
    type: Metric
  targetNodepoolIds:
  - 7d48eca7-3419-11ea-bc29-0255ac1001a8
```

**Table 10-14** Key parameters

Parameter	Type	Description
spec.disable	Bool	Whether to enable the scaling policy. This parameter takes effect for all rules in the policy.
spec.rules	Array	All rules in a scaling policy.
spec.rules[x].ruleName	String	Rule name.



Parameter	Type	Description
spec.rules[x].type	String	Rule type. <b>Cron</b> and <b>Metric</b> are supported.
spec.rules[x].disable	Bool	Rule switch. Currently, only <b>false</b> is supported.
spec.rules[x].action.type	String	Rule action type. Currently, only <b>ScaleUp</b> is supported.
spec.rules[x].action.unit	String	Rule action unit. Currently, only <b>Node</b> is supported.
spec.rules[x].action.value	Integer	Rule action value.
spec.rules[x].cronTrigger	N/A	Optional. This parameter is valid only in periodic rules.
spec.rules[x].cronTrigger.schedule	String	Cron expression of a periodic rule.
spec.rules[x].metricTrigger	N/A	Optional. This parameter is valid only in metric-based rules.
spec.rules[x].metricTrigger.metricName	String	Metric of a metric-based rule. Currently, <b>Cpu</b> and <b>Memory</b> are supported.
spec.rules[x].metricTrigger.metricOperation	String	Comparison operator of a metric-based rule. Currently, only <b>&gt;</b> is supported.
spec.rules[x].metricTrigger.metricValue	String	Threshold of the metric rule. The value can be an integer ranging from 1 to 100 and must be a character. If the value is set to <b>-1</b> , the threshold is automatically calculated.
spec.rules[x].metricTrigger.Unit	String	Unit of the metric-based rule threshold. Currently, only <b>%</b> is supported.
spec.targetNodepoolIds	Array	All node pools associated with the scaling policy.
spec.targetNodepoolIds[x]	String	UID of the node pool associated with the scaling policy.

## 10.3.3 Managing Node Scaling Policies

### Scenario

After a node scaling policy is created, you can delete, edit, disable, enable, or clone the policy.

### Viewing a Node Scaling Policy

You can view the associated node pool, rules, and scaling history of a node scaling policy and rectify faults according to the error information displayed.

- 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 **Nodes**. On the page displayed, click the **Node Pools** tab and then the name of the node pool for which an auto scaling policy has been created to view the node pool details.
- Step 3** On the node pool details page, click the **Auto Scaling** tab to view the auto scaling configuration and scaling records.

#### NOTE

You can obtain created auto scaling policies on the **Policies** page.

1. Log in to the CCE console and click the cluster name to access the cluster console.
2. In the navigation pane, choose **Policies**. On the page displayed, click the **Node Scaling Policies** tab.
3. Check the configuration of the auto scaling policies. Choose **More > Scaling History** for the target policy to check the scaling records of the policy.

----End

### Deleting a Node Scaling Policy

- 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 **Policies**. On the page displayed, click the **Node Scaling Policies** tab, locate the row containing the target policy and choose **More > Delete** in the **Operation** column.
- Step 3** In the **Delete Node Scaling Policy** dialog box displayed, confirm whether to delete the policy.
- Step 4** Click **Yes** to delete the policy.

----End

### Editing a Node Scaling Policy

- 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 **Policies**. On the page displayed, click the **Node Scaling Policies** tab, locate the row containing the target policy and click **Edit** in the **Operation** column.
- Step 3** On the **Edit Node Scaling Policy** page displayed, configure policy parameters listed in [Table 10-14](#).

**Step 4** After the configuration is complete, click **OK**.

----End

## Cloning a Node Scaling Policy

**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 **Policies**. On the page displayed, click the **Node Scaling Policies** tab, locate the row containing the target policy and choose **More > Clone** in the **Operation** column.

**Step 3** On the **Create Node Scaling Policy** page displayed, certain parameters have been cloned. Add or modify other policy parameters based on service requirements.

**Step 4** Click **OK**.

----End

## Enabling or Disabling a Node Scaling Policy

**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 **Policies**. On the page displayed, click the **Node Scaling Policies** tab, locate the row containing the target policy click **Disable** in the **Operation** column. If the policy is in the disabled state, click **Enable** in the **Operation** column.

**Step 3** In the dialog box displayed, confirm whether to disable or enable the node policy.

----End

# 10.4 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.

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. Pod scaling can effectively resolve this issue. If the resource usage on the node increases to a certain extent, newly added pods cannot be scheduled to this node. In this case, CCE will add nodes accordingly.

## Solution

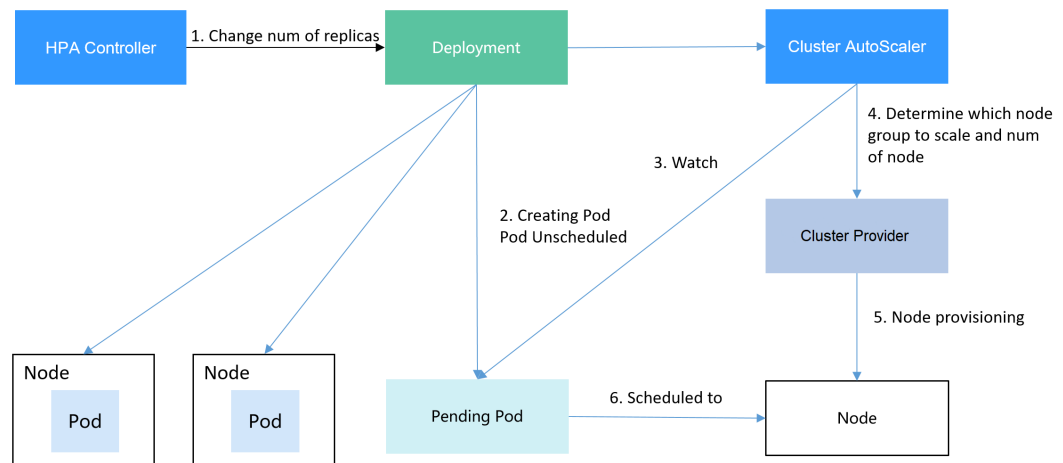
Two major auto scaling policies are HPA (Horizontal Pod Autoscaling) and CA (Cluster AutoScaling). HPA is for workload auto scaling and CA is for node auto scaling.

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 10-10**, 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.

**Figure 10-10** 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 cores of vCPUs and 4 GiB of memory, or a higher specification, 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.

**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  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 {Image repository address}/group/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 {Image repository address}/group/hpa-example: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 defaults for other parameters.

**Step 3** Locate the row containing the newly created node pool and click **Auto Scaling** in the upper right corner.

If the CCE Cluster Autoscaler add-on is not installed in the cluster, install it first.

- **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

**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:
            limits: # The value of limits must be the same as that of requests to prevent flapping
              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.

```
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 hpa-example 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
```

## 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
```

 **NOTE**

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.

Observe the scaling process of the workload.

```
# kubectl get hpa hpa-policy --watch
NAME          REFERENCE          TARGETS  MINPODS  MAXPODS  REPLICAS  AGE
hpa-policy    Deployment/hpa-example  0%/50%   1         100       1          4m
hpa-policy    Deployment/hpa-example  190%/50%  1         100       1          4m23s
hpa-policy    Deployment/hpa-example  190%/50%  1         100       4          4m31s
hpa-policy    Deployment/hpa-example  200%/50%  1         100       4          5m16s
hpa-policy    Deployment/hpa-example  200%/50%  1         100       4          6m16s
hpa-policy    Deployment/hpa-example  85%/50%   1         100       4          7m16s
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    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
# 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
```

Check the number of nodes. The following output shows that two nodes are added.

```
# kubectl get node
NAME          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 get hpa hpa-policy --watch
NAME          REFERENCE          TARGETS  MINPODS  MAXPODS  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 hpa-
```

```
example-79dd795485 to 3
Normal ScalingReplicaSet 72s deployment-controller Scaled down replica set hpa-
example-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).

----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.

# 11 Namespaces

---

## 11.1 Creating a Namespace

### Scenario

A namespace is a collection of resources and objects. Multiple namespaces can be created inside a cluster and isolated from each other. This enables namespaces to share the same cluster Services without affecting each other.

For example, you can deploy workloads in a development environment into one namespace, and deploy workloads in a testing environment into another namespace.

### Prerequisites

At least one cluster has been created.

### Notes and Constraints

A maximum of 6000 Services can be created in each namespace. The Services mentioned here indicate the Kubernetes Service resources added for workloads.

### Namespace Types

Namespaces can be created automatically or manually.

- Created automatically: When a cluster is up, the **default**, **kube-public**, **kube-system**, and **kube-node-lease** namespaces are created by default.
  - **default**: All objects for which no namespace is specified are allocated to this namespace.
  - **kube-public**: Resources in this namespace can be accessed by all users (including unauthenticated users) so that some resources in the cluster can be readable in the entire cluster. This is a reserved Kubernetes namespace. Its common attributes are only conventions but not requirements.

- **kube-system:** All resources created by Kubernetes are in this namespace.
- **kube-node-lease:** Each node has an associated Lease object in this namespace. The object is periodically updated by the node. Both NodeStatus and NodeLease are considered as heartbeats from a node. In versions earlier than v1.13, only NodeStatus is available. The NodeLease feature is introduced in v1.13. NodeLease is more lightweight than NodeStatus. This feature significantly improves the cluster scalability and performance.
- **Created manually:** You can create namespaces to serve separate purposes. For example, you can create three namespaces, one for a development environment, one for joint debugging environment, and one for test environment. You can also create one namespace for login services and one for game services.

## Creating a Namespace

- Step 1** Log in to the CCE console and click the cluster name to access the cluster console.
- Step 2** Choose **Namespaces** in the navigation pane and click **Create Namespace** in the upper right corner.
- Step 3** Set namespace parameters based on [Table 11-1](#).

**Table 11-1** Parameters for creating a namespace

Parameter	Description
Name	Unique name of the created namespace.
Description	Description about the namespace.
Quota Management	<p>Resource quotas can limit the amount of resources available in namespaces, achieving resource allocation by namespace.</p> <p><b>NOTICE</b>  <b>You are advised to set resource quotas in the namespace as required to prevent cluster or node exceptions caused by resource overload.</b></p> <p>For example, the default number of pods that can be created on each node in a cluster is 110. If you create a cluster with 50 nodes, you can create a maximum of 5,500 pods. Therefore, you can set a resource quota to ensure that the total number of pods in all namespaces does not exceed 5,500.</p> <p>Enter an integer. If the quota of a resource is not specified, no limit is posed on the resource.</p> <p>If you want to limit the CPU or memory quota, you must specify the CPU or memory request value when creating a workload.</p>

- Step 4** After the configuration is complete, click **OK**.

----End

## Using kubectl to Create a Namespace

Define a namespace.

```
apiVersion: v1
kind: Namespace
metadata:
  name: custom-namespace
```

Run the **kubectl** command to create it.

```
$ kubectl create -f custom-namespace.yaml
namespace/custom-namespace created
```

You can also run the **kubectl create namespace** command to create a namespace.

```
$ kubectl create namespace custom-namespace
namespace/custom-namespace created
```

## 11.2 Managing Namespaces

### Using Namespaces

- When creating a workload, you can select a namespace to isolate resources or users.
- When querying workloads, you can select a namespace to view all workloads in the namespace.

### Isolating Namespaces

- **Isolating namespaces by environment**

An application generally goes through the development, joint debugging, and testing stages before it is launched. In this process, the workloads deployed in each environment (stage) are the same, but are logically defined. There are two ways to define them:

- Group them in different clusters for different environments.

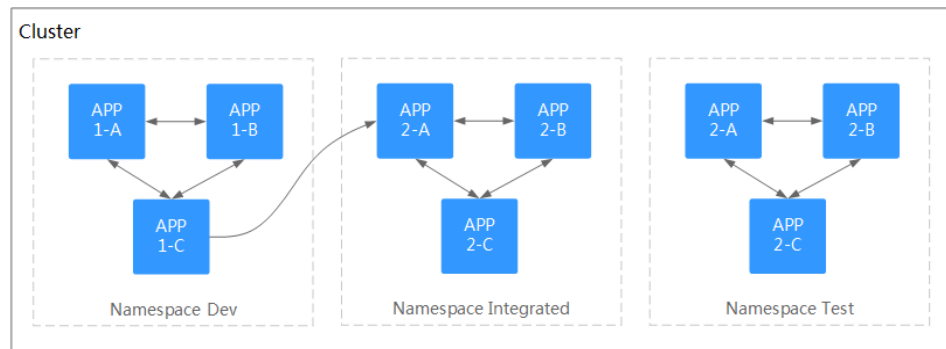
Resources cannot be shared among different clusters. In addition, services in different environments can access each other only through load balancing.

- Group them in different namespaces for different environments.

Workloads in the same namespace can be mutually accessed by using the Service name. Cross-namespace access can be implemented by using the Service name or namespace name.

The following figure shows namespaces created for the development, joint debugging, and testing environments, respectively.

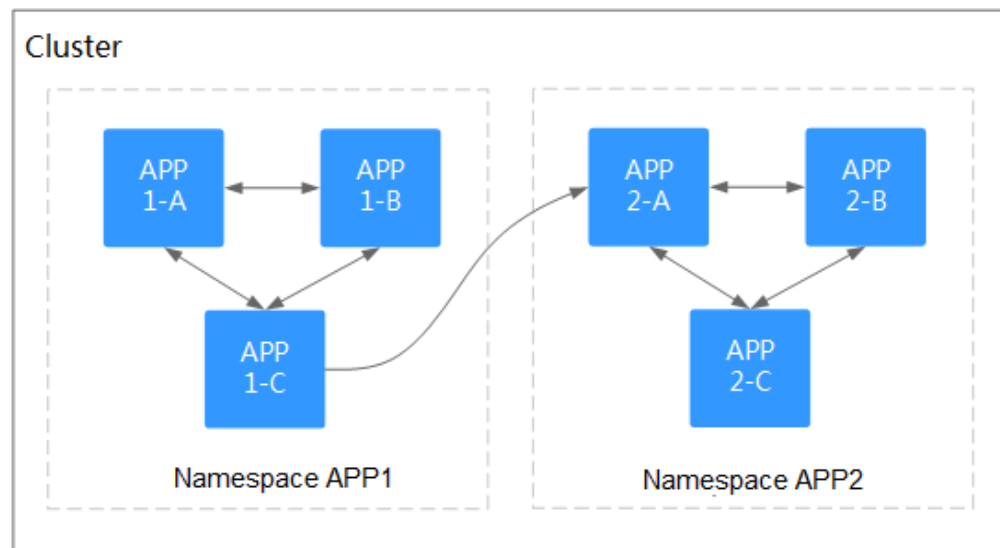
**Figure 11-1** One namespace for one environment



- **Isolating namespaces by application**

You are advised to use this method if a large number of workloads are deployed in the same environment. For example, in the following figure, different namespaces (APP1 and APP2) are created to logically manage workloads as different groups. Workloads in the same namespace access each other using the Service name, and workloads in different namespaces access each other using the Service name or namespace name.

**Figure 11-2** Grouping workloads into different namespaces



## Managing Namespace Labels

- Step 1** Log in to the CCE console and click the cluster name to access the cluster console. In the navigation pane, choose **Namespaces**.
- Step 2** Locate the row containing the target namespace and choose **More > Manage Label** in the **Operation** column.
- Step 3** In the dialog box that is displayed, the existing labels of the namespace are displayed. Modify the labels as needed.
  - Adding a label: Click the add icon, enter the key and value of the label to be added, and click **OK**.  
For example, the key is **project** and the value is **cidc**, indicating that the namespace is used to deploy CICD.

- Deleting a label: Click **Delete** next the label to be deleted and then **OK**.


**Step 4** Switch to the **Manage Label** dialog box again and check the modified labels.

----End

## Enabling Node Affinity in a Namespace

After node affinity is enabled in a namespace, the workloads newly created in the namespace can be scheduled only to nodes with specific labels. For details, see [PodNodeSelector](#).

**Step 1** Log in to the CCE console and click the cluster name to access the cluster console. In the navigation pane, choose **Namespaces**.

**Step 2** Locate the target namespace and click  in the **Node Affinity** column.

**Step 3** In the displayed dialog box, select **Enable** and click **OK**.

After node affinity is enabled, new workloads in the current namespace will be scheduled only to nodes with specified labels. For example, in namespace **test**, the workloads in the namespace can be scheduled only to the node whose label key is **kubelet.kubernetes.io/namespace** and label value is **test**.

**Step 4** You can add specified labels to a node in **Labels and Taints** on the **Nodes** page. For details, see [Managing Node Labels](#).

----End

## Deleting a Namespace

If a namespace is deleted, all resources (such as workloads, jobs, and ConfigMaps) in this namespace will also be deleted. Exercise caution when deleting a namespace.

**Step 1** Log in to the CCE console and click the cluster name to access the cluster console.

**Step 2** Choose **Namespaces** in the navigation pane. On the displayed page, click **More** in the row of the target namespace and choose **Delete**.

Follow the prompts to delete the namespace. The default namespaces cannot be deleted.

----End

## 11.3 Configuring Resource Quotas

Kubernetes provides namespaces for you to group resources in a cluster. Namespaces serve different purposes to meet the needs of multiple users, environments, and applications. To prevent resource abuse and ensure cluster reliability, you can configure quotas for resources like CPUs, memory, and pods for each namespace. For details, see [Resource Quotas](#).

In clusters of v1.21 or later, the default resource quotas will be created when a namespace is created if you have enabled **enable-resource-quota** in [Cluster Configuration Management](#). [Table 11-2](#) lists the resource quotas based on

cluster specifications. You can modify them according to your service requirements.

**Table 11-2** Default resource quotas

Cluster Scale	Pod	Deployment	Secret	ConfigMap	Service
50 nodes	2000	1000	1000	1000	1000
200 nodes	2000	1000	1000	1000	1000
1000 nodes	5000	2000	2000	2000	2000
2000 nodes	5000	2000	2000	2000	2000

## Procedure

- Step 1** Log in to the CCE console and click the cluster name to access the cluster console.
- Step 2** In the navigation pane, click **Namespaces**.
- Step 3** Click **Quota Management** next to the target namespace.  
This operation cannot be performed on system namespaces **kube-system** and **kube-public**.
- Step 4** Configure the resource quotas and click **OK**.



---

**NOTICE**

- After configuring CPU and memory quotas for a namespace, you must specify the request and limit values of CPU and memory resources when creating a workload. Otherwise, the workload cannot be created. If the quota of a resource is set to **0**, the resource usage is not limited.
- Accumulated quota usage includes the resources used by CCE to create default components, such as the Kubernetes Services (which can be viewed using `kubectl`) created under the **default** namespace. Therefore, you are advised to set a resource quota greater than expected to reserve resource for creating default components.
- Kubernetes provides optimistic concurrency control (OCC), also known as optimistic locking, for frequent data updates. You can use optimistic locking by defining the **resourceVersion** field. This field is in the object metadata. This field identifies the internal version number of the object. When the object is modified, this field is modified accordingly. You can use `kube-apiserver` to check whether an object has been modified. When the API server receives an update request containing the **resourceVersion** field, the server compares the requested data with the resource version number of the server. If they are different, the object on the server has been modified when the update is submitted. In this case, the API server returns a conflict error (409). Obtain the server data, modify the data, and submit the data to the server again. Resource quotas limit the total resource usage of each namespace and maintain a record of resource information in the cluster. Enabling resource quotas may result in more resource creation conflicts in high-concurrency scenarios, which can affect the performance of batch resource creation.

---

----End

# 12 ConfigMaps and Secrets

---

## 12.1 Creating a ConfigMap

### Scenario

A ConfigMap is a type of resource that stores configuration information required by a workload. Its content is user-defined. After creating ConfigMaps, you can use them as files or environment variables in a containerized workload.

ConfigMaps allow you to decouple configuration files from container images to enhance the portability of workloads.

Benefits of ConfigMaps:

- Manage configurations of different environments and services.
- Deploy workloads in different environments. Multiple versions are supported for configuration files so that you can update and roll back workloads easily.
- Quickly import configurations in the form of files to containers.

### Notes and Constraints

- The size of a ConfigMap resource file cannot exceed 1 MB.
- ConfigMaps cannot be used in [static pods](#).

### Procedure

- Step 1** Log in to the CCE console and click the cluster name to access the cluster console.
- Step 2** Choose **ConfigMaps and Secrets** in the navigation pane and click **Create ConfigMap** in the upper right corner.
- Step 3** Configure parameters.

**Table 12-1** Parameters for creating a ConfigMap

Parameter	Description
Name	Name of the ConfigMap you create, which must be unique in a namespace.
Namespace	Namespace to which the ConfigMap belongs. If you do not specify this parameter, the value <b>default</b> is used by default.
Description	Description of the ConfigMap.
Data	Data of a ConfigMap, in the key-value pair format. Click <b>+</b> to add data. The value can be in string, JSON, or YAML format.
Label	Label of the ConfigMap. Enter a key-value pair and click <b>Confirm</b> .

**Step 4** Click **OK**.

The new ConfigMap is displayed in the ConfigMap list.

----End

## Creating a ConfigMap Using kubectl

**Step 1** Use kubectl to access the cluster. For details, see [Connecting to a Cluster Using kubectl](#).

**Step 2** Create a file named **cce-configmap.yaml** and edit it.

**vi cce-configmap.yaml**

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: cce-configmap
data:
  SPECIAL_LEVEL: Hello
  SPECIAL_TYPE: CCE
```

**Table 12-2** Key parameters

Parameter	Description
apiVersion	The value is fixed at <b>v1</b> .
kind	The value is fixed at <b>ConfigMap</b> .
metadata.name	ConfigMap name, which can be customized.
data	ConfigMap data. The value must be key-value pairs.

**Step 3** Run the following commands to create a ConfigMap.

**kubectl create -f cce-configmap.yaml**

Run the following commands to view the created ConfigMap:

**kubectl get cm**

```
NAME          DATA      AGE
cce-configmap 3          7m
```

----End

## Related Operations

After creating a ConfigMap, you can update or delete it as described in [Table 12-3](#).

**Table 12-3** Related operations

Operation	Description
Editing a YAML file	Click <b>Edit YAML</b> in the row where the target ConfigMap resides to edit its YAML file.
Updating a ConfigMap	<ol style="list-style-type: none"> <li>1. Select the name of the ConfigMap to be updated and click <b>Update</b>.</li> <li>2. Modify the secret data. For more information, see <a href="#">Table 12-1</a>.</li> <li>3. Click <b>OK</b>.</li> </ol>
Deleting a ConfigMap	Select the configuration you want to delete and click <b>Delete</b> . Follow the prompts to delete the ConfigMap.

## 12.2 Using a ConfigMap

After a ConfigMap is created, it can be used in three workload scenarios: environment variables, command line parameters, and data volumes.

- [Configuring Environment Variables of a Workload](#)
- [Configuring Command Line Parameters](#)
- [Mounting a ConfigMap to the Workload Data Volume](#)

The following example shows how to use a ConfigMap.

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: cce-configmap
data:
  SPECIAL_LEVEL: Hello
  SPECIAL_TYPE: CCE
```

**NOTICE**

- When a ConfigMap is used in a workload, the workload and ConfigMap must be in the same cluster and namespace.
- When a ConfigMap is mounted as a data volume and the ConfigMap is updated, Kubernetes updates the data in the data volume at the same time.  
For a ConfigMap data volume mounted in **subPath** mode, Kubernetes cannot automatically update data in the data volume when the ConfigMap is updated.
- When a ConfigMap is used as an environment variable, data is not automatically updated when the ConfigMap is updated. To update the data, restart the pod.

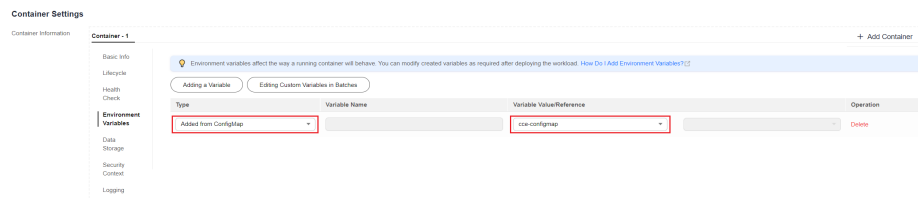
## Configuring Environment Variables of a Workload

### Using the CCE console

- 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 **Workloads**. Then, click **Create Workload** in the upper right corner.

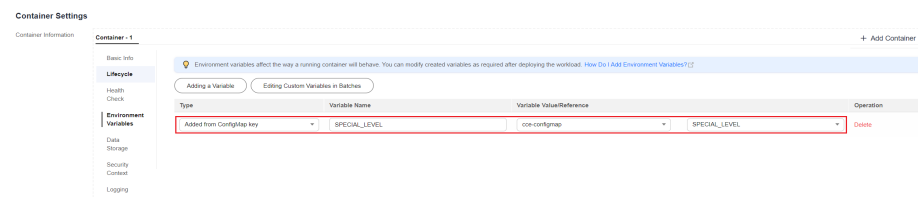
When creating a workload, click **Environment Variables** in the **Container Settings** area, and click **Add Variable**.

- **Added from ConfigMap:** Select a ConfigMap to import all of its keys as environment variables.



- **Added from ConfigMap key:** Import a key in a ConfigMap as the value of an environment variable.
  - **Variable Name:** name of an environment variable in the workload. The name can be customized and is set to the key name selected in the ConfigMap by default.
  - **Variable Value/Reference:** Select a ConfigMap and the key to be imported. The corresponding value is imported as a workload environment variable.

For example, after you import the value **Hello** of **SPECIAL\_LEVEL** in ConfigMap **cce-configmap** as the value of workload environment variable **SPECIAL\_LEVEL**, an environment variable named **SPECIAL\_LEVEL** with its value **Hello** exists in the container.



**Step 3** Configure other workload parameters and click **Create Workload**.

After the workload runs properly, [log in to the container](#) and run the following statement to check whether the ConfigMap has been set as an environment variable of the workload:

```
printenv SPECIAL_LEVEL
```

The example output is as follows:

```
Hello
```

----End

**Using kubectl**

**Step 1** Use kubectl to access the cluster. For details, see [Connecting to a Cluster Using kubectl](#).

**Step 2** Create a file named **nginx-configmap.yaml** and edit it.

**vi nginx-configmap.yaml**

Content of the YAML file:

- **Added from ConfigMap:** To add all data in a ConfigMap to environment variables, use the **envFrom** parameter. The keys in the ConfigMap will become names of environment variables in the workload.

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-configmap
spec:
  replicas: 1
  selector:
    matchLabels:
      app: nginx-configmap
  template:
    metadata:
      labels:
        app: nginx-configmap
    spec:
      containers:
        - name: container-1
          image: nginx:latest
          envFrom:
            # Use envFrom to specify a ConfigMap to be referenced by
            # environment variables.
            - configMapRef:
                name: cce-configmap # Name of the referenced ConfigMap.
          imagePullSecrets:
            - name: default-secret
```

- **Added from ConfigMap key:** When creating a workload, you can use a ConfigMap to set environment variables and use the **valueFrom** parameter to reference the key-value pair in the ConfigMap separately.

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-configmap
spec:
  replicas: 1
  selector:
    matchLabels:
      app: nginx-configmap
  template:
    metadata:
      labels:
```

```

app: nginx-configmap
spec:
  containers:
  - name: container-1
    image: nginx:latest
    env:
      # Set the environment variable in the workload.
      - name: SPECIAL_LEVEL # Name of the environment variable in the workload.
        valueFrom: # Specify a ConfigMap to be referenced by the environment variable.
          configMapKeyRef:
            name: cce-configmap # Name of the referenced ConfigMap.
            key: SPECIAL_LEVEL # Key in the referenced ConfigMap.
    - name: SPECIAL_TYPE # Add multiple environment variables to import them at the
      same time.
      valueFrom:
        configMapKeyRef:
          name: cce-configmap
          key: SPECIAL_TYPE
    imagePullSecrets:
    - name: default-secret

```

**Step 3** Create a workload.

**kubectl apply -f nginx-configmap.yaml**

**Step 4** View the environment variables in the pod.

1. Run the following command to view the created pod:

```
kubectl get pod | grep nginx-configmap
```

Expected output:

```
nginx-configmap-*** 1/1 Running 0 2m18s
```

2. Run the following command to view the environment variables in the pod:

```
kubectl exec nginx-configmap-*** -- printenv SPECIAL_LEVEL SPECIAL_TYPE
```

Expected output:

```
Hello
CCE
```

The ConfigMap has been set as environment variables of the workload.

----End

## Configuring Command Line Parameters

You can use a ConfigMap as an environment variable to set commands or parameter values for a container by using the environment variable substitution syntax `$(VAR_NAME)`.

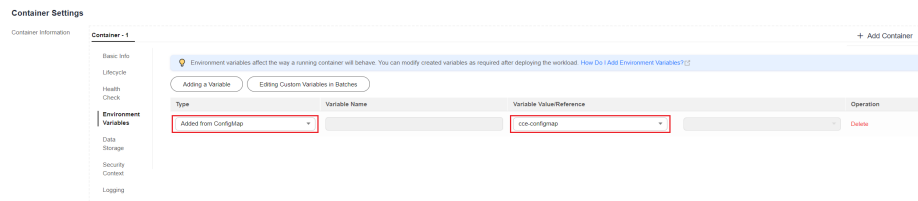
### Using the CCE console

**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 **Workloads**. In the dialog box displayed, click **Create Workload** in the upper right corner.

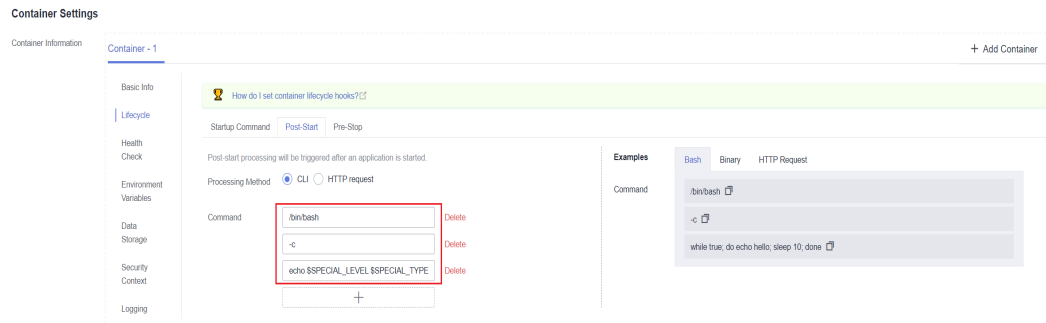
When creating a workload, click **Environment Variables** in the **Container Settings** area, and click **Add Variable**. In this example, select **Added from ConfigMap**.

- **Added from ConfigMap:** Select a ConfigMap to import all of its keys as environment variables.



**Step 3** Click **Lifecycle** in the **Container Settings** area, click the **Post-Start** tab on the right, and set the following parameters:

- **Processing Method: CLI**
- **Command:** Enter the following three command lines. *SPECIAL\_LEVEL* and *SPECIAL\_TYPE* are the environment variable names in the workload, which are key names in the **cce-configmap** ConfigMap.  
/bin/bash  
-c  
echo \$SPECIAL\_LEVEL \$SPECIAL\_TYPE > /usr/share/nginx/html/index.html



**Step 4** Configure other workload parameters and click **Create Workload**.

After the workload runs properly, [log in to the container](#) and run the following statement to check whether the ConfigMap has been set as an environment variable of the workload:

```
cat /usr/share/nginx/html/index.html
```

The example output is as follows:

```
Hello CCE
```

----End

### Using kubectl

**Step 1** Use kubectl to access the cluster. For details, see [Connecting to a Cluster Using kubectl](#).

**Step 2** Create a file named **nginx-configmap.yaml** and edit it.

#### vi nginx-configmap.yaml

In the following example, the **cce-configmap** ConfigMap is imported to the workload. *SPECIAL\_LEVEL* and *SPECIAL\_TYPE* are the environment variable names in the workload, which are key names in the **cce-configmap** ConfigMap.

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-configmap
spec:
```



```
replicas: 1
selector:
  matchLabels:
    app: nginx-configmap
template:
  metadata:
    labels:
      app: nginx-configmap
  spec:
    containers:
      - name: container-1
        image: nginx:latest
        lifecycle:
          postStart:
            exec:
              command: [ "/bin/sh", "-c", "echo $SPECIAL_LEVEL $SPECIAL_TYPE > /usr/share/nginx/html/
index.html" ]
        envFrom:
          # Use envFrom to specify a ConfigMap to be referenced by environment
          variables.
          - configMapRef:
              name: cce-configmap # Name of the referenced ConfigMap.
        imagePullSecrets:
          - name: default-secret
```

**Step 3** Create a workload.

```
kubectl apply -f nginx-configmap.yaml
```

**Step 4** Wait until the workload runs properly. Then, data will be added the `/usr/share/nginx/html/index.html` file in the container.

1. Run the following command to view the created pod:

```
kubectl get pod | grep nginx-configmap
```

Expected output:

```
nginx-configmap-*** 1/1 Running 0 2m18s
```

2. Run the following command to view the environment variables in the pod:

```
kubectl exec nginx-configmap-*** -- cat /usr/share/nginx/html/index.html
```

Expected output:

```
Hello CCE
```

----End

## Mounting a ConfigMap to the Workload Data Volume

The data stored in a ConfigMap can be referenced in a volume of type ConfigMap. You can mount such a volume to a specified container path. The platform supports the separation of workload codes and configuration files. ConfigMap volumes are used to store workload configuration parameters. Before that, create ConfigMaps in advance. For details, see [Creating a ConfigMap](#).

### Using the CCE console

**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 **Workloads**. In the dialog box displayed, click **Create Workload** in the upper right corner.

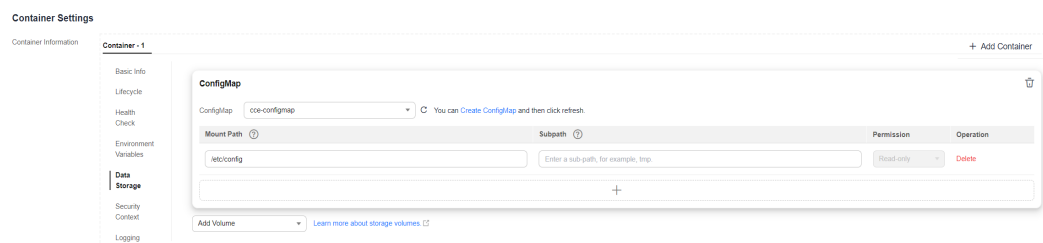
When creating a workload, click **Data Storage** in the **Container Settings** area. Click **Add Volume** and select **ConfigMap** from the drop-down list.

**Step 3** Select parameters for mounting a ConfigMap volume, as shown in [Table 12-4](#).

**Table 12-4** Mounting a ConfigMap volume

Parameter	Description
ConfigMap	Select the desired ConfigMap. A ConfigMap must be created beforehand. For details, see <a href="#">Creating a ConfigMap</a> .
Mount Path	Enter a mount path. After the ConfigMap volume is mounted, a configuration file with the key as the file name and value as the file content is generated in the mount path of the container.  This parameter specifies a container path to which a data volume will be mounted. Do not mount the volume to a system directory such as / or <b>/var/run</b> . This may lead to container errors. Mount the volume to an empty directory. If the directory is not empty, ensure that there are no files that affect container startup. Otherwise, the files will be replaced, leading to container startup failures or workload creation failures.  <b>NOTICE</b> If the container is mounted to a high-risk directory, use an account with minimum permissions to start the container. Otherwise, high-risk files on the host may be damaged.
Subpath	Enter a subpath of the mount path.  <ul style="list-style-type: none"> <li>• A subpath is used to mount a local volume so that the same data volume is used in a single pod. If this parameter is left blank, the root path will be used by default.</li> <li>• The subpath can be the key and value of a ConfigMap or secret. If the subpath is a key-value pair that does not exist, the data import does not take effect.</li> <li>• The data imported by specifying a subpath will not be updated along with the ConfigMap/secret updates.</li> </ul>
Permission	Read-only, indicating that data volume in the path is read-only.

**Figure 12-1** Mounting a ConfigMap to a workload data volume



**Step 4** After the configuration, click **Create Workload**.

After the workload runs properly, the **SPECIAL\_LEVEL** and **SPECIAL\_TYPE** files will be generated in the **/etc/config** directory in this example. The contents of the files are **Hello** and **CCE**, respectively.

[Access the container](#) and run the following statement to view the **SPECIAL\_LEVEL** or **SPECIAL\_TYPE** file in the container:

```
cat /etc/config/SPECIAL_LEVEL
```

Expected output:

```
Hello
```

----End

### Using kubectl

**Step 1** Use kubectl to access the cluster. For details, see [Connecting to a Cluster Using kubectl](#).

**Step 2** Create a file named **nginx-configmap.yaml** and edit it.

#### vi nginx-configmap.yaml

As shown in the following example, after the ConfigMap volume is mounted, a configuration file with the key as the file name and value as the file content is generated in the **/etc/config** directory of the container.

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-configmap
spec:
  replicas: 1
  selector:
    matchLabels:
      app: nginx-configmap
  template:
    metadata:
      labels:
        app: nginx-configmap
    spec:
      containers:
        - name: container-1
          image: nginx:latest
          volumeMounts:
            - name: config-volume
              mountPath: /etc/config # Mount to the /etc/config directory.
              readOnly: true
      volumes:
        - name: config-volume
          configMap:
            name: cce-configmap # Name of the referenced ConfigMap.
```

**Step 3** Create a workload.

#### kubectl apply -f nginx-configmap.yaml

**Step 4** After the workload runs properly, the **SPECIAL\_LEVEL** and **SPECIAL\_TYPE** files will be generated in the **/etc/config** directory. The contents of the files are **Hello** and **CCE**, respectively.

1. Run the following command to view the created pod:

```
kubectl get pod | grep nginx-configmap
```

Expected output:

```
nginx-configmap-*** 1/1 Running 0 2m18s
```

2. Run the following command to view the **SPECIAL\_LEVEL** or **SPECIAL\_TYPE** file in the pod:

```
kubectl exec nginx-configmap-*** -- cat /etc/config/SPECIAL_LEVEL
```

Expected output:

```
Hello
```

----End

## 12.3 Creating a Secret

### Scenario

A secret is a type of resource that holds sensitive data, such as authentication and key information. Its content is user-defined. After creating secrets, you can use them as files or environment variables in a containerized workload.

### Notes and Constraints

Secrets cannot be used in [static pods](#).

### Procedure

- Step 1** Log in to the CCE console and click the cluster name to access the cluster console.
- Step 2** Choose **ConfigMaps and Secrets** in the navigation pane, click the **Secrets** tab, and click **Create Secret** in the upper right corner.
- Step 3** Configure parameters.

**Table 12-5** Parameters for creating a secret

Parameter	Description
Name	Name of the secret you create, which must be unique.
Namespace	Namespace to which the secret belongs. If you do not specify this parameter, the value <b>default</b> is used by default.
Description	Description of a secret.
Type	Type of the secret you create. <ul style="list-style-type: none"> <li>● Opaque: common secret.</li> <li>● <code>kubernetes.io/dockerconfigjson</code>: a secret that stores the authentication information required for pulling images from a private repository.</li> <li>● <code>kubernetes.io/tls</code>: Kubernetes TLS secret, which is used to store the certificate required by layer-7 load balancing Services. For details about examples of the <code>kubernetes.io/tls</code> secret and its description, see <a href="#">TLS secrets</a>.</li> <li>● <code>IngressTLS</code>: TLS secret provided by CCE to store the certificate required by layer-7 load balancing Services.</li> <li>● Other: another type of secret, which is specified manually.</li> </ul>

Parameter	Description
Secret Data	<p>Workload secret data can be used in containers.</p> <ul style="list-style-type: none"> <li>If <b>Secret Type</b> is <b>Opaque</b>, click <b>+</b>. In the dialog box displayed, enter a key-value pair and select <b>Auto Base64 Encoding</b>.</li> <li>If <b>Secret Type</b> is <b>kubernetes.io/dockerconfigjson</b>, enter the account and password for logging in to the private image repository.</li> <li>If <b>Secret Type</b> is <b>kubernetes.io/tls</b> or <b>IngressTLS</b>, upload the certificate file and private key file.</li> </ul> <p><b>NOTE</b></p> <ul style="list-style-type: none"> <li>A certificate is a self-signed or CA-signed credential used for identity authentication.</li> <li>A certificate request is a request for a signature with a private key.</li> </ul>
Secret Label	Label of the secret. Enter a key-value pair and click <b>Confirm</b> .

**Step 4** Click **OK**.

The new secret is displayed in the key list.

----End

## Secret Resource File Configuration Example

This section describes configuration examples of secret resource description files.

- Opaque type

The **secret.yaml** file is defined as shown below. The **data** field is filled in as a key-value pair, and the **value** field must be encoded using Base64. For details about the Base64 encoding method, see [Base64 Encoding](#).

```
apiVersion: v1
kind: Secret
metadata:
  name: mysecret          #Secret name
  namespace: default     #Namespace. The default value is default.
data:
  <your_key>: <your_value> # Enter a key-value pair. The value must be encoded using Base64.
type: Opaque
```

- kubernetes.io/dockerconfigjson type

The **secret.yaml** file is defined as shown below. The value of **.dockerconfigjson** must be encoded using Base64. For details, see [Base64 Encoding](#).

```
apiVersion: v1
kind: Secret
metadata:
  name: mysecret          #Secret name
  namespace: default     #Namespace. The default value is default.
data:
  .dockerconfigjson: eyJh***** # Content encoded using Base64.
type: kubernetes.io/dockerconfigjson
```

To obtain the **.dockerconfigjson** content, perform the following steps:

- a. Obtain the following login information of the image repository.
  - Image repository address: The section uses *address* as an example. Replace it with the actual address.
  - Username: The section uses *username* as an example. Replace it with the actual username.
  - Password: The section uses *password* as an example. Replace it with the actual password.

- b. Use Base64 to encode the key-value pair *username:password* and fill the encoded content in **3**.

```
echo -n "username:password" | base64
```

Command output:

```
dXNlcm5hbWU6cGFzc3dvcmQ=
```

- c. Use Base64 to encode the following JSON content:

```
echo -n '{"auths":{"address":  
{"username":"username","password":"password","auth":"dXNlcm5hbWU6cGFzc3dvcmQ="}}}'  
| base64
```

Command output:

```
eyJhdXRocyl6eyJhZGRyZXNzIjp7InVzZXJhZG90eS99LjoidXNlcm5hbWU6cGFzc3dvcmQ=IiwiaWF0IjoxOTY0aCI6ImR5TmxbTVoYldVNmNHRnpjM2R2Y21RPSJ9fX0=
```

The encoded content is the **.dockerconfigjson** content.

- kubernetes.io/tls type

The value of **tls.crt** and **tls.key** must be encoded using Base64. For details, see [Base64 Encoding](#).

```
kind: Secret
apiVersion: v1
metadata:
  name: mysecret          #Secret name
  namespace: default     #Namespace. The default value is default.
data:
  tls.crt: LS0tLS1CRU*****FURS0tLS0t # Certificate content, which must be encoded using Base64.
  tls.key: LS0tLS1CRU*****VZLS0tLS0= # Private key content, which must be encoded using Base64.
type: kubernetes.io/tls
```

- IngressTLS type

The value of **tls.crt** and **tls.key** must be encoded using Base64. For details, see [Base64 Encoding](#).

```
kind: Secret
apiVersion: v1
metadata:
  name: mysecret          #Secret name
  namespace: default     #Namespace. The default value is default.
data:
  tls.crt: LS0tLS1CRU*****FURS0tLS0t # Certificate content, which must be encoded using Base64.
  tls.key: LS0tLS1CRU*****VZLS0tLS0= # Private key content, which must be encoded using Base64.
type: IngressTLS
```

## Creating a Secret Using kubectl

**Step 1** Use kubectl to access the cluster. For details, see [Connecting to a Cluster Using kubectl](#).

**Step 2** Create and edit the Base64-encoded **cce-secret.yaml** file.

```
# echo -n "content to be encoded" | base64
*****
```

### vi cce-secret.yaml

The following YAML file uses the Opaque type as an example. For details about other types, see [Secret Resource File Configuration Example](#).

```
apiVersion: v1
kind: Secret
metadata:
  name: mysecret
type: Opaque
data:
  <your_key>: <your_value> # Enter a key-value pair. The value must be encoded using Base64.
```

**Step 3** Create a secret.

```
kubectl create -f cce-secret.yaml
```

You can query the secret after creation.

```
kubectl get secret -n default
```

```
----End
```

## Related Operations

After creating a secret, you can update or delete it as described in [Table 12-6](#).

### NOTE

The secret list contains system secret resources that can be queried only. The system secret resources cannot be updated or deleted.

**Table 12-6** Related Operations

Operation	Description
Editing a YAML file	Click <b>Edit YAML</b> in the row where the target secret resides to edit its YAML file.
Updating a secret	<ol style="list-style-type: none"> <li>1. Select the name of the secret to be updated and click <b>Update</b>.</li> <li>2. Modify the secret data. For more information, see <a href="#">Table 12-5</a>.</li> <li>3. Click <b>OK</b>.</li> </ol>
Deleting a secret	Select the secret you want to delete and click <b>Delete</b> . Follow the prompts to delete the secret.
Deleting secrets in batches	<ol style="list-style-type: none"> <li>1. Select the secrets to be deleted.</li> <li>2. Click <b>Delete</b> above the secret list.</li> <li>3. Follow the prompts to delete the secrets.</li> </ol>

## Base64 Encoding

To Base64-encode a string, run the `echo -n content to be encoded | base64` command. The following is an example:

```
root@ubuntu:~# echo -n "content to be encoded" | base64  
*****
```

## 12.4 Using a Secret

After secrets are created, they can be mounted as data volumes or be exposed as environment variables to be used by a container in a pod.

### NOTICE

Do not perform any operation on the following secrets. For details, see [Cluster Secrets](#).

- Do not operate secrets under kube-system.
- Do not operate default-secret and paas.elb in any of the namespaces. The default-secret is used to pull the private image of SWR, and the paas.elb is used to connect the service in the namespace to the ELB service.

- [Configuring Environment Variables of a Workload](#)
- [Configuring the Data Volume of a Workload](#)

The following example shows how to use a secret.

```
apiVersion: v1  
kind: Secret  
metadata:  
  name: mysecret  
type: Opaque  
data:  
  username: ***** #The value must be Base64-encoded.  
  password: ***** #The value must be encoded using Base64.
```

### NOTICE

- When a secret is used in a pod, the pod and secret must be in the same cluster and namespace.
- When a secret is updated, Kubernetes updates the data in the data volume at the same time.

However, when a secret data volume mounted in [subPath](#) mode is updated, Kubernetes cannot automatically update the data in the data volume.

## Configuring Environment Variables of a Workload

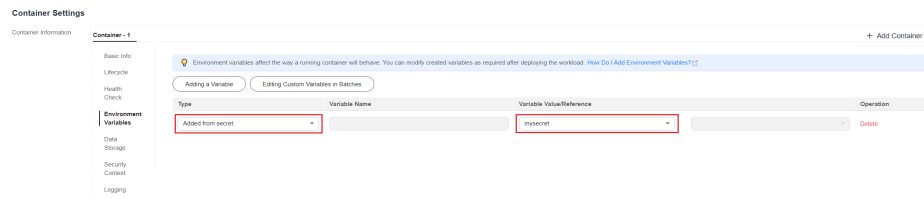
### Using the CCE console

- 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 **Workloads**. In the dialog box displayed, click **Create Workload** in the upper right corner.

When creating a workload, click **Environment Variables** in the **Container Settings** area, and click **Add Variable**.

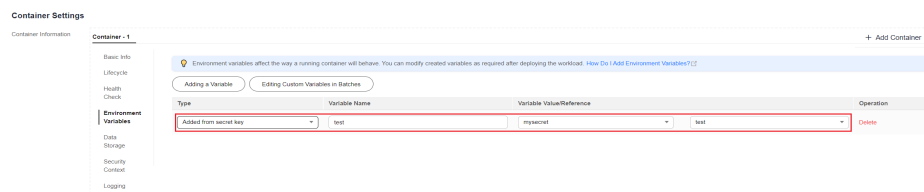


- **Added from secret:** Select a secret and import all keys in the secret as environment variables.



- **Added from secret key:** Import the value of a key in a secret as the value of an environment variable.
  - **Variable Name:** name of an environment variable in the workload. The name can be customized and is set to the key name selected in the secret by default.
  - **Variable Value/Reference:** Select a secret and the key to be imported. The corresponding value is imported as a workload environment variable.

For example, after you import the value of **username** in secret **mysecret** as the value of workload environment variable **username**, an environment variable named **username** exists in the container.



### Step 3 Configure other workload parameters and click **Create Workload**.

After the workload runs properly, **log in to the container** and run the following statement to check whether the secret has been set as an environment variable of the workload:

```
printenv username
```

If the output is the same as the content in the secret, the secret has been set as an environment variable of the workload.

----End

### Using kubectl

**Step 1** Use kubectl to access the cluster. For details, see [Connecting to a Cluster Using kubectl](#).

**Step 2** Create a file named **nginx-secret.yaml** and edit it.

**vi nginx-secret.yaml**

Content of the YAML file:

- **Added from secret:** To add all data in a secret to environment variables, use the **envFrom** parameter. The keys in the secret will become names of environment variables in a workload.

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-secret
```

```
spec:
  replicas: 1
  selector:
    matchLabels:
      app: nginx-secret
  template:
    metadata:
      labels:
        app: nginx-secret
    spec:
      containers:
        - name: container-1
          image: nginx:latest
          envFrom:
            # Use envFrom to specify a secret to be referenced by environment
            variables:
              - secretRef:
                  name: mysecret # Name of the referenced secret.
            imagePullSecrets:
              - name: default-secret
```

- **Added from secret key:** When creating a workload, you can use a secret to set environment variables and use the **valueFrom** parameter to reference the key-value pair in the secret separately.

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-secret
spec:
  replicas: 1
  selector:
    matchLabels:
      app: nginx-secret
  template:
    metadata:
      labels:
        app: nginx-secret
    spec:
      containers:
        - name: container-1
          image: nginx:latest
          env:
            # Set the environment variable in the workload.
            - name: SECRET_USERNAME # Name of the environment variable in the workload.
              valueFrom:
                # Use valueFrom to specify a secret to be referenced by environment
                variables:
                  secretKeyRef:
                    name: mysecret # Name of the referenced secret.
                    key: username # Key in the referenced secret.
            - name: SECRET_PASSWORD # Add multiple environment variables to import them at
              the same time.
          valueFrom:
            secretKeyRef:
              name: mysecret
              key: password
            imagePullSecrets:
              - name: default-secret
```

**Step 3** Create a workload.

**kubectl apply -f nginx-secret.yaml**

**Step 4** View the environment variables in the pod.

1. Run the following command to view the created pod:

```
kubectl get pod | grep nginx-secret
```

Expected output:

```
nginx-secret-*** 1/1 Running 0 2m18s
```

2. Run the following command to view the environment variables in the pod:

```
kubectl exec nginx-secret-*** -- printenv SPECIAL_USERNAME SPECIAL_PASSWORD
```

If the output is the same as the content in the secret, the secret has been set as an environment variable of the workload.

----End

## Configuring the Data Volume of a Workload

You can mount a secret as a volume to the specified container path. Contents in a secret are user-defined. Before that, create a secret. For details, see [Creating a Secret](#).

### Using the CCE console

**Step 1** Log in to the CCE console and click the cluster name to access the cluster console.

**Step 2** Choose **Workloads** in the navigation pane. In the right pane, click the **Deployments** tab. Click **Create Workload** in the upper right corner.

When creating a workload, click **Data Storage** in the **Container Settings** area. Click **Add Volume** and select **Secret** from the drop-down list.

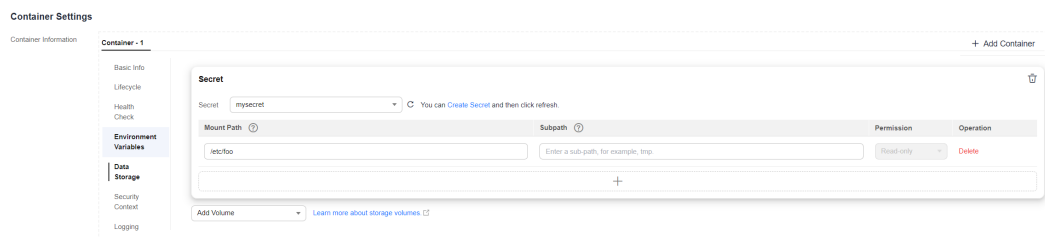
**Step 3** Select parameters for mounting a secret volume, as shown in [Table 12-7](#).

**Table 12-7** Mounting a secret volume

Parameter	Description
Secret	Select the desired secret. A secret must be created beforehand. For details, see <a href="#">Creating a Secret</a> .
Mount Path	Enter a mount path. After the secret volume is mounted, a secret file with the key as the file name and value as the file content is generated in the mount path of the container. This parameter specifies a container path to which a data volume will be mounted. Do not mount the volume to a system directory such as / or <b>/var/run</b> . This may lead to container errors. Mount the volume to an empty directory. If the directory is not empty, ensure that there are no files that affect container startup. Otherwise, the files will be replaced, leading to container startup failures or workload creation failures. <b>NOTICE</b> If the container is mounted to a high-risk directory, use an account with minimum permissions to start the container. Otherwise, high-risk files on the host may be damaged.

Parameter	Description
Subpath	<p>Enter a subpath of the mount path.</p> <ul style="list-style-type: none"> <li>A subpath is used to mount a local volume so that the same data volume is used in a single pod. If this parameter is left blank, the root path will be used by default.</li> <li>The subpath can be the key and value of a ConfigMap or secret. If the subpath is a key-value pair that does not exist, the data import does not take effect.</li> <li>The data imported by specifying a subpath will not be updated along with the ConfigMap/secret updates.</li> </ul>
Permission	Read-only, indicating that data volume in the path is read-only.

**Figure 12-2** Mounting a secret to a workload data volume



**Step 4** After the configuration, click **Create Workload**.

After the workload runs properly, the **username** and **password** files will be generated in the **/etc/foo** directory in this example. The contents of the files are secret values.

**Access the container** and run the following statement to view the **username** or **password** file in the container:

```
cat /etc/foo/username
```

The expected output is the same as the content in the secret.

----End

### Using kubectl

**Step 1** Use kubectl to access the cluster. For details, see [Connecting to a Cluster Using kubectl](#).

**Step 2** Create a file named **nginx-secret.yaml** and edit it.

#### vi nginx-secret.yaml

In the following example, the username and password in the **mysecret** secret are saved in the **/etc/foo** directory as files.

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-secret
```

```
spec:
  replicas: 1
  selector:
    matchLabels:
      app: nginx-secret
  template:
    metadata:
      labels:
        app: nginx-secret
    spec:
      containers:
        - name: container-1
          image: nginx:latest
          volumeMounts:
            - name: foo
              mountPath: /etc/foo      # Mount to the /etc/foo directory.
              readOnly: true
      volumes:
        - name: foo
          secret:
            secretName: mysecret      # Name of the referenced secret.
```

You can also use the **items** field to control the mapping path of secret keys. For example, store username in the **/etc/foo/my-group/my-username** directory in the container.

#### NOTE

- If you use the **items** field to specify the mapping path of the secret keys, the keys that are not specified will not be created as files. For example, if the **password** key in the following example is not specified, the file will not be created.
- If you want to use all keys in a secret, you must list all keys in the **items** field.
- All keys listed in the **items** field must exist in the corresponding secret. Otherwise, the volume is not created.

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-secret
spec:
  replicas: 1
  selector:
    matchLabels:
      app: nginx-secret
  template:
    metadata:
      labels:
        app: nginx-secret
    spec:
      containers:
        - name: container-1
          image: nginx:latest
          volumeMounts:
            - name: foo
              mountPath: /etc/foo      # Mount to the /etc/foo directory.
              readOnly: true
      volumes:
        - name: foo
          secret:
            secretName: mysecret      # Name of the referenced secret.
            items:
              - key: username          # Name of the referenced key.
                path: my-group/my-username # Mapping path of the secret key
```

**Step 3** Create a workload.

```
kubectl apply -f nginx-secret.yaml
```

**Step 4** After the workload runs properly, the **username** and **password** files are generated in the **/etc/foo** directory.

1. Run the following command to view the created pod:

```
kubectl get pod | grep nginx-secret
```

Expected output:

```
nginx-secret-*** 1/1 Running 0 2m18s
```

2. Run the following command to view the **username** or **password** file in the pod:

```
kubectl exec nginx-secret-*** -- cat /etc/foo/username
```

The expected output is the same as the content in the secret.

----End

## 12.5 Cluster Secrets

By default, CCE creates the following secrets in each namespace:

- default-secret
- paas.elb
- default-token-xxxxx (xxxxx is a random number.)

The functions of these secrets are described as follows.

### default-secret

The type of **default-secret** is **kubernetes.io/dockerconfigjson**. The data is the credential for logging in to the SWR image repository and is used to pull images from SWR. To pull an image from SWR when creating a workload on CCE, set **imagePullSecrets** to **default-secret**.

```
apiVersion: v1
kind: Pod
metadata:
  name: nginx
spec:
  containers:
  - image: nginx:alpine
    name: container-0
  resources:
    limits:
      cpu: 100m
      memory: 200Mi
    requests:
      cpu: 100m
      memory: 200Mi
  imagePullSecrets:
  - name: default-secret
```

The data of **default-secret** is updated periodically, and the current data will expire after a certain period of time. You can run the **describe** command to view the expiration time in default-secret.

**NOTICE**

Use `default-secret` directly instead of copying the secret content to create a new one. The credential in the copied secret will expire and the image cannot be pulled.

```
$ kubectl describe secret default-secret
Name:         default-secret
Namespace:    default
Labels:       secret-generated-by=cce
Annotations:  temporary-ak-sk-expires-at: 2021-11-26 20:55:31.380909 +0000 UTC

Type: kubernetes.io/dockerconfigjson

Data
====
.dockerconfigjson: 347 bytes
```

## paas.elb

The **paas.elb** data stores a temporary AK/SK that is used when a node is created or a load balancer is automatically created. The **paas.elb** data is updated periodically and has a specific time limit before it expires.

In practice, you will not directly use **paas.elb**. Do not delete it, as doing so will result in the failure of creating a node or load balancer. will fail.

## default-token-xxxxx

By default, Kubernetes creates a service account named **default** for each namespace. **default-token-xxxxx** is the key of the service account, and **xxxxx** is a random number.

```
$ kubectl get sa
NAME      SECRETS  AGE
default  1         30d
$ kubectl describe sa default
Name:         default
Namespace:    default
Labels:       <none>
Annotations:  <none>
Image pull secrets: <none>
Mountable secrets:  default-token-xxxxx
Tokens:       default-token-xxxxx
Events:       <none>
```

# 13 Add-ons

## 13.1 Overview

CCE provides multiple types of add-ons to extend cluster functions and meet feature requirements. You can install add-ons as required.

### NOTICE

CCE uses Helm charts to deploy add-ons. To modify or upgrade an add-on, perform operations on the **Add-ons** page or use open add-on management APIs. Do not directly modify resources related to add-ons in the background. Otherwise, add-on exceptions or other unexpected problems may occur.

### Scheduling and Elasticity Add-ons

Add-on Name	Description
<b>Volcano Scheduler</b>	This add-on is a scheduler for general-purpose, high-performance computing such as job scheduling, heterogeneous chip management, and job running management, serving end users through computing frameworks for different industries such as AI, big data, gene sequencing, and rendering.
<b>CCE Cluster Autoscaler</b>	This add-on resizes a cluster based on pod scheduling status and resource usage.
<b>CCE Advanced HPA</b>	This add-on is developed by CCE. It can be used to flexibly scale in or out Deployments based on metrics such as CPU usage and memory usage.



## Cloud Native Observability Add-ons

Add-on Name	Description
<a href="#">Cloud Native Cluster Monitoring</a>	This add-on uses Prometheus-operator and Prometheus to provide easy-to-use, end-to-end Kubernetes cluster monitoring.  With this add-on, you can access the monitoring center to view monitoring data and configure alarms.
<a href="#">CCE Node Problem Detector</a>	This add-on monitors abnormal events of cluster nodes and connects to a third-party monitoring platform. It is a daemon running on each node. It collects node issues from different daemons and reports them to the API server. It can run as a DaemonSet or a daemon.
<a href="#">CCE Network Metrics Exporter</a>	This add-on monitors and manages container network traffic. It collects how many IPv4 packets and bytes are received and sent (including those sent to the Internet) and allows you to obtain pod labels. It supports multiple monitoring tasks, allows you to select monitoring metrics, and uses a PodSelector to select monitoring backends. The monitoring information has been adapted to Prometheus. You can call the Prometheus API to view monitoring data.
<a href="#">Kubernetes Metrics Server</a>	This add-on is an aggregator for monitoring data of core cluster resources.
<a href="#">Prometheus</a>	This add-on is an open-source system monitoring and alerting framework. CCE allows you to quickly install Prometheus as an add-on.

## Cloud Native Heterogeneous Computing Add-ons

Add-on Name	Description
<a href="#">CCE AI Suite (NVIDIA GPU)</a>	This add-on supports and manages GPUs in containers. Only NVIDIA drivers are supported.
<a href="#">CCE AI Suite (Ascend NPU)</a>	This add-on supports and manages NPUs in containers.

## Container Network Add-ons

Add-on Name	Description
<a href="#">CoreDNS</a>	CoreDNS is a DNS server that provides domain name resolution for Kubernetes clusters through a chain add-on.

Add-on Name	Description
<a href="#">NGINX Ingress Controller</a>	This add-on forwards application data such as the data of virtual hosts, load balancers, SSL proxy, and HTTP routing for Services that can be directly accessed outside a cluster.
<a href="#">NodeLocal DNSCache</a>	NodeLocal DNSCache improves cluster DNS performance by running DNS cache proxies as DaemonSets on cluster nodes.

## Container Storage Add-on

Add-on Name	Description
<a href="#">CCE Container Storage (Everest)</a>	This add-on is a cloud native container storage system, which enables clusters of Kubernetes v1.15.6 or later to use cloud storage through the Container Storage Interface (CSI).

## Other Add-ons

Add-on Name	Description
<a href="#">Kubernetes Dashboard</a>	This add-on is a general-purpose, web-based UI for Kubernetes clusters and integrates all commands that can be used in the CLI. It allows users to manage applications running in a cluster and troubleshoot faults, as well as manage the cluster itself.

## Add-on Lifecycle

An add-on lifecycle involves all the statuses of the add-on from installation to uninstallation.

**Table 13-1** Add-on statuses

Status	Attribute	Description
Running	Stable state	The add-on is running properly, all add-on instances are deployed properly, and the add-on can be used properly.
Partially ready	Stable state	The add-on is running properly, but some add-on instances are not properly deployed. In this state, the add-on functions may be unavailable.
Unavailable	Stable state	The add-on malfunctions, and all add-on instances are not properly deployed.

Status	Attribute	Description
Installing	Intermediate state	The add-on is being deployed. If all instances cannot be scheduled due to incorrect add-on configuration or insufficient resources, the system sets the add-on status to <b>Unavailable</b> 10 minutes later.
Installation failed	Stable state	Install add-on failed. Uninstall it and try again.
Upgrading	Intermediate state	The add-on is being upgraded.
Upgrade failed	Stable state	Upgrade add-on failed. Upgrade it again, or uninstall it and try again.
Rolling back	Intermediate state	The add-on is rolling back.
Rollback failed	Stable state	The add-on rollback failed. Retry the rollback, or uninstall it and try again.
Deleting	Intermediate state	The add-on is being deleted. If this state stays for a long time, an exception occurred.
Deletion failed	Stable state	Delete add-on failed. Try again.
Unknown	Stable state	No add-on chart found.

 **NOTE**

When an add-on is in an intermediate state such as **Installing** or **Deleting**, you are not allowed to edit or uninstall the add-on.

If the add-on status is unknown and the returned **status.Reason** is "don't install the add-on in this cluster", the secret associated with the Helm release of the add-on in the cluster is typically deleted by mistake. In this case, uninstall the add-on and reinstall it with the same configurations.

## Related Operations

You can perform the operations listed in [Table 13-2](#) on the **Add-ons** page.

**Table 13-2** Related operations

Operation	Description	Procedure
Install	Install a specified add-on.	<ol style="list-style-type: none"> <li>1. Log in to the CCE console and click the cluster name to access the cluster console. In the navigation pane, choose <b>Add-ons</b>.</li> <li>2. Click <b>Install</b> under the target add-on. Each add-on has different configuration parameters. For details, see the corresponding chapter.</li> <li>3. Click <b>OK</b>.</li> </ol>
Upgrade	Upgrade an add-on to the new version.	<ol style="list-style-type: none"> <li>1. Log in to the CCE console and click the cluster name to access the cluster console. In the navigation pane, choose <b>Add-ons</b>.</li> <li>2. If an add-on can be upgraded, the <b>Upgrade</b> button is displayed under it. Click <b>Upgrade</b>. Each add-on has different configuration parameters. For details, see the corresponding chapter.</li> <li>3. Click <b>OK</b>.</li> </ol>
Edit	Edit add-on parameters.	<ol style="list-style-type: none"> <li>1. Log in to the CCE console and click the cluster name to access the cluster console. In the navigation pane, choose <b>Add-ons</b>.</li> <li>2. Click <b>Edit</b> under the target add-on. Each add-on has different configuration parameters. For details, see the corresponding chapter.</li> <li>3. Click <b>OK</b>.</li> </ol>
Uninstall	Uninstall an add-on from the cluster.	<ol style="list-style-type: none"> <li>1. Log in to the CCE console and click the cluster name to access the cluster console. In the navigation pane, choose <b>Add-ons</b>.</li> <li>2. Click <b>Uninstall</b> under the target add-on.</li> <li>3. In the displayed dialog box, click <b>Yes</b>. This operation cannot be undone.</li> </ol>

Operation	Description	Procedure
Roll back	Roll back an add-on to the source version.  <b>NOTE</b> <ul style="list-style-type: none"> <li>• This function is used to roll back an upgraded add-on to the source version, not to undo the editing of add-on parameters.</li> <li>• An add-on cannot be rolled back repeatedly.</li> </ul>	<ol style="list-style-type: none"> <li>1. Log in to the CCE console and click the cluster name to access the cluster console. In the navigation pane, choose <b>Add-ons</b>.</li> <li>2. If an add-on can be rolled back, the <b>Roll Back</b> button is displayed under it. Click <b>Roll Back</b>.</li> <li>3. In the displayed dialog box, click <b>Yes</b>.</li> </ol>

 **NOTE**

Add-on rollback is supported in certain add-on versions.

- CoreDNS: 1.25.11 and later versions
- Everest: 2.1.19 and later versions
- Autoscaler:
  - v1.21 clusters: v1.21.22 and later versions
  - v1.23 clusters: v1.23.24 and later versions
  - v1.25 clusters: v1.25.14 and later versions
- kube-prometheus-stack: v3.7.2 and later versions
- Volcano: 1.11.4 and later versions
- NPD: 1.18.22 and later versions

## 13.2 Scheduling and Elasticity Add-ons

### 13.2.1 Volcano Scheduler

#### Introduction

**Volcano** is a batch processing platform based on Kubernetes. It provides a series of features required by machine learning, deep learning, bioinformatics, genomics, and other big data applications, as a powerful supplement to Kubernetes capabilities.

Volcano provides general computing capabilities such as high-performance job scheduling, heterogeneous chip management, and job running management. It accesses the computing frameworks for various industries such as AI, big data, gene, and rendering and schedules up to 1000 pods per second for end users, greatly improving scheduling efficiency and resource utilization.

Volcano provides job scheduling, job management, and queue management for computing applications. Its main features are as follows:

- Diverse computing frameworks, such as TensorFlow, MPI, and Spark, can run on Kubernetes in containers. Common APIs for batch computing jobs through CRD, various plugins, and advanced job lifecycle management are provided.
- Advanced scheduling capabilities are provided for batch computing and high-performance computing scenarios, including group scheduling, preemptive priority scheduling, packing, resource reservation, and task topology.
- Queues can be effectively managed for scheduling jobs. Complex job scheduling capabilities such as queue priority and multi-level queues are supported.

Volcano has been open-sourced in GitHub at <https://github.com/volcano-sh/volcano>.

Install and configure the Volcano add-on in CCE clusters. For details, see [Volcano Scheduling](#).

#### NOTE

When using Volcano as a scheduler, use it to schedule all workloads in the cluster. This prevents resource scheduling conflicts caused by simultaneous working of multiple schedulers.

## Installing the Add-on

**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 **Volcano Scheduler** on the right, and click **Install**.

**Step 2** On the **Install Add-on** page, configure the specifications as needed.

- If you selected **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 selected **Custom**, you can adjust the number of pods and resource quotas as needed. High availability is not possible with a single pod. If an error occurs on the node where the add-on instance runs, the add-on will fail. The resource quotas of the volcano-admission component are irrelevant to the number of cluster nodes and pods. You can retain the default value. The resource quotas of volcano-controller and volcano-scheduler are related to the number of cluster nodes and pods. The recommended values are as follows:
  - If the number of nodes is less than 100, retain the default configuration. The requested vCPUs are 500m, and the limit is 2000m. The requested memory is 500 MiB, and the limit is 2000 MiB.
  - If the number of nodes is greater than 100, increase the requested vCPUs by 500m and the requested memory by 1000 MiB each time 100 nodes (10,000 pods) are added. Increase the vCPU limit by 1500m and the memory limit by 1000 MiB.

 NOTE

Recommended formula for calculating the requested value:

- Requested vCPUs: Calculate the number of target nodes multiplied by the number of target pods, perform interpolation search based on the number of nodes in the cluster multiplied by the number of target pods in [Table 13-3](#), and round up the request value and limit value that are closest to the specifications.

For example, for 2000 nodes and 20,000 pods, Number of target nodes x Number of target pods = 40 million, which is close to the specification of 700/70,000 (Number of cluster nodes x Number of pods = 49 million). According to the following table, set the requested vCPUs to 4000m and the limit value to 5500m.

- Requested memory: It is recommended that 2.4 GiB memory be allocated to every 1000 nodes and 1 GiB memory be allocated to every 10,000 pods. The requested memory is the sum of these two values. (The obtained value may be different from the recommended value in [Table 13-3](#). You can use either of them.)

Requested memory = Number of target nodes/1000 x 2.4 GiB + Number of target pods/10,000 x 1 GiB

For example, for 2000 nodes and 20,000 pods, the requested memory is 6.8 GiB (2000/1000 x 2.4 GiB + 20,000/10,000 x 1 GiB).

**Table 13-3** Recommended requested resources and resource limits for volcano-controller and volcano-scheduler

Nodes/Pods in a Cluster	CPU Request (m)	CPU Limit (m)	Memory Request (MiB)	Memory Limit (MiB)
50/5000	500	2000	500	2000
100/10000	1000	2500	1500	2500
200/20000	1500	3000	2500	3500
300/30000	2000	3500	3500	4500
400/40000	2500	4000	4500	5500
500/50000	3000	4500	5500	6500
600/60000	3500	5000	6500	7500
700/70000	4000	5500	7500	8500

**Step 3** Configure the extended functions supported by the add-on.

- Descheduling:** After this function is enabled, the volcano-descheduler component is automatically deployed. The scheduler will evict and reschedule pods that do not meet your policy configuration requirements. This helps to balance cluster load and reduce resource fragmentation. For details, see [Descheduling](#).
- Hybrid Service Deployment:** After this function is enabled, the volcano-agent component is automatically deployed in the node pool with the hybrid

deployment capability enabled, which improves resource utilization by ensuring node QoS, enabling CPU bursts, and allowing for dynamic resource oversubscription. This helps to reduce resource usage costs.

- **NUMA Topology Scheduling:** After this function is enabled, the resource-exporter component is automatically deployed. The scheduler will schedule pods in NUMA affinity mode, which enhances the performance of high-performance training jobs. For details, see [NUMA Affinity Scheduling](#).

**Step 4** Configure deployment policies for the add-on pods.

 **NOTE**

- Scheduling policies do not take effect on add-on instances of the DaemonSet type.
- When configuring multi-AZ deployment or node affinity, ensure that there are nodes meeting the scheduling policy and that resources are sufficient in the cluster. Otherwise, the add-on cannot run.

**Table 13-4** Configurations for add-on scheduling

Parameter	Description
Multi AZ	<ul style="list-style-type: none"> <li>• <b>Preferred:</b> Deployment pods of the add-on will be preferentially scheduled to nodes in different AZs. If all the nodes in the cluster are deployed in the same AZ, the pods will be scheduled to different nodes in that AZ.</li> <li>• <b>Required:</b> Deployment pods of the add-on are forcibly scheduled to nodes in different AZs. There can be at most one pod in each AZ. If nodes in a cluster are not in different AZs, some add-on pods cannot run properly. If a node is faulty, add-on pods on it may fail to be migrated.</li> </ul>
Node Affinity	<ul style="list-style-type: none"> <li>• <b>Not configured:</b> Node affinity is disabled for the add-on.</li> <li>• <b>Node Affinity:</b> Specify the nodes where the add-on is deployed. If you do not specify the nodes, the add-on will be randomly scheduled based on the default cluster scheduling policy.</li> <li>• <b>Specified Node Pool Scheduling:</b> Specify the node pool where the add-on is deployed. If you do not specify the node pool, the add-on will be randomly scheduled based on the default cluster scheduling policy.</li> <li>• <b>Custom Policies:</b> Enter the labels of the nodes where the add-on is to be deployed for more flexible scheduling policies. If you do not specify node labels, the add-on will be randomly scheduled based on the default cluster scheduling policy. If multiple custom affinity policies are configured, ensure that there are nodes that meet all the affinity policies in the cluster. Otherwise, the add-on cannot run.</li> </ul>



Parameter	Description
Toleration	<p>Using both taints and tolerations allows (not forcibly) the add-on Deployment to be scheduled to a node with the matching taints, and controls the Deployment eviction policies after the node where the Deployment is located is tainted.</p> <p>The add-on adds the default tolerance policy for the <b>node.kubernetes.io/not-ready</b> and <b>node.kubernetes.io/unreachable</b> taints, respectively. The tolerance time window is 60s.</p> <p>For details, see <a href="#">Configuring Tolerance Policies</a>.</p>

### Step 5 Click **Install**.

After the add-on is installed, you can choose **Settings** in the navigation pane, switch to the **Scheduling** tab, select **Volcano scheduler**, and find the corresponding expert mode. You can customize advanced scheduling policies based on actual service scenarios. The following is an example:

```

colocation_enable: "
default_scheduler_conf:
  actions: 'allocate, backfill, preempt'
  tiers:
    - plugins:
      - name: 'priority'
      - name: 'gang'
      - name: 'conformance'
      - name: 'lifecycle'
      arguments:
        lifecycle.MaxGrade: 10
        lifecycle.MaxScore: 200.0
        lifecycle.SaturatedTresh: 1.0
        lifecycle.WindowSize: 10
    - plugins:
      - name: 'drf'
      - name: 'predicates'
      - name: 'nodeorder'
    - plugins:
      - name: 'cce-gpu-topology-predicate'
      - name: 'cce-gpu-topology-priority'
      - name: 'cce-gpu'
    - plugins:
      - name: 'nodelocalvolume'
      - name: 'nodeemptydirvolume'
      - name: 'nodeCSIScheduling'
      - name: 'networkresource'
tolerations:
  - effect: NoExecute
    key: node.kubernetes.io/not-ready
    operator: Exists
    tolerationSeconds: 60
  - effect: NoExecute
    key: node.kubernetes.io/unreachable
    operator: Exists
    tolerationSeconds: 60

```

**Table 13-5** Advanced Volcano configuration parameters

Plugin	Function	Description	Demonstration
colocation_enable	Whether to enable hybrid deployment.	Value: <ul style="list-style-type: none"> <li>• <b>true</b>: hybrid enabled</li> <li>• <b>false</b>: hybrid disabled</li> </ul>	None
default_scheduler_conf	Used to schedule pods. It consists of a series of actions and plugins and features high scalability. You can specify and implement actions and plugins based on your requirements.	It consists of actions and tiers. <ul style="list-style-type: none"> <li>• <b>actions</b>: defines the types and sequence of actions to be executed by the scheduler.</li> <li>• <b>tiers</b>: configures the plugin list.</li> </ul>	None

Plugin	Function	Description	Demonstration
actions	<p>Actions to be executed in each scheduling phase. The configured action sequence is the scheduler execution sequence. For details, see <a href="#">Actions</a>.</p> <p>The scheduler traverses all jobs to be scheduled and performs actions such as enqueue, allocate, preempt, and backfill in the configured sequence to find the most appropriate node for each job.</p>	<p>The following options are supported:</p> <ul style="list-style-type: none"> <li>• <b>enqueue</b>: uses a series of filtering algorithms to filter out tasks to be scheduled and sends them to the queue to wait for scheduling. After this action, the task status changes from <b>pending</b> to <b>inqueue</b>.</li> <li>• <b>allocate</b>: selects the most suitable node based on a series of pre-selection and selection algorithms.</li> <li>• <b>preempt</b>: performs preemption scheduling for tasks with higher priorities in the same queue based on priority rules.</li> <li>• <b>backfill</b>: schedules pending tasks as much as possible to maximize the utilization of node resources.</li> </ul>	<p>actions: 'allocate, backfill, preempt'</p> <p><b>NOTE</b> When configuring <b>actions</b>, use either <b>preempt</b> or <b>enqueue</b>.</p>
plugins	<p>Implementation details of algorithms in actions based on different scenarios. For details, see <a href="#">Plugins</a>.</p>	<p>For details, see <a href="#">Table 13-6</a>.</p>	<p>None</p>
tolerations	<p>Tolerance of the add-on to node taints.</p>	<p>By default, the add-on can run on nodes with the <b>node.kubernetes.io/not-ready</b> or <b>node.kubernetes.io/unreachable</b> taint and the taint effect value is <b>NoExecute</b>, but it'll be evicted in 60 seconds.</p>	<p>tolerations:</p> <ul style="list-style-type: none"> <li>- effect: NoExecute key: node.kubernetes.io/not-ready operator: Exists tolerationSeconds: 60</li> <li>- effect: NoExecute key: node.kubernetes.io/unreachable operator: Exists tolerationSeconds: 60</li> </ul>

**Table 13-6** Supported plugins

Plugin	Function	Description	Demonstration
binpack	Schedule pods to nodes with high resource usage (not allocating pods to light-loaded nodes) to reduce resource fragments.	<p><b>arguments:</b></p> <ul style="list-style-type: none"> <li>• <b>binpack.weight:</b> weight of the binpack plugin.</li> <li>• <b>binpack.cpu:</b> ratio of CPUs to all resources. The parameter value defaults to 1.</li> <li>• <b>binpack.memory:</b> ratio of memory resources to all resources. The parameter value defaults to 1.</li> <li>• <b>binpack.resources:</b> other custom resource types requested by the pod, for example, <b>nvidia.com/gpu</b>. Multiple types can be configured and be separated by commas (,).</li> <li>• <b>binpack.resources.&lt;your_resource&gt;:</b> weight of your custom resource in all resources. Multiple types of resources can be added. <i>&lt;your_resource&gt;</i> indicates the resource type defined in <b>binpack.resources</b>, for example, <b>binpack.resources.nvidia.com/gpu</b>.</li> </ul>	<pre>- plugins: - name: binpack   arguments:     binpack.weight: 10     binpack.cpu: 1     binpack.memory: 1     binpack.resources:       nvidia.com/gpu,       example.com/foo  binpack.resources.nvidia.com/ gpu: 2  binpack.resources.example.co m/foo: 3</pre>
conformance	Prevent key pods, such as the pods in the <b>kube-system</b> namespace from being preempted.	None	<pre>- plugins: - name: 'priority' - name: 'gang'   enablePreemptable: false - name: 'conformance'</pre>

Plugin	Function	Description	Demonstration
lifecycle	<p>By collecting statistics on service scaling rules, pods with similar lifecycles are preferentially scheduled to the same node. With the horizontal scaling capability of the Autoscaler, resources can be quickly scaled in and released, reducing costs and improving resource utilization.</p> <ol style="list-style-type: none"> <li>1. Collects statistics on the lifecycle of pods in the service load and schedules pods with similar lifecycles to the same node.</li> <li>2. For a cluster configured with an automatic scaling policy, adjust the scale-in annotation of the node to preferentially scale in the node with low usage.</li> </ol>	<p><b>arguments:</b></p> <ul style="list-style-type: none"> <li>• <b>lifecycle.WindowSize</b> : The value is an integer greater than or equal to 1 and defaults to <b>10</b>. Record the number of times that the number of replicas changes. If the load changes regularly and periodically, decrease the value. If the load changes irregularly and the number of replicas changes frequently, increase the value. If the value is too large, the learning period is prolonged and too many events are recorded.</li> <li>• <b>lifecycle.MaxGrade</b>: The value is an integer greater than or equal to 3 and defaults to <b>3</b>. It indicates levels of replicas. For example, if the value is set to <b>3</b>, the replicas are classified into three levels. If the load changes regularly and periodically, decrease the value. If the load changes irregularly, increase the value. Setting an excessively small value may result in inaccurate lifecycle forecasts.</li> <li>• <b>lifecycle.MaxScore</b>: float64 floating point number. The value must be greater than or equal to 50.0. The default value is <b>200.0</b>.</li> </ul>	<pre>- plugins: - name: priority - name: gang   enablePreemptable: false - name: conformance - <b>name: lifecycle</b>   arguments:     lifecycle.MaxGrade: 3     lifecycle.MaxScore: 200.0     lifecycle.SaturatedTresh: 0.8     lifecycle.WindowSize: 10</pre> <p><b>NOTE</b></p> <ul style="list-style-type: none"> <li>• For nodes that do not want to be scaled in, manually mark them as long-period nodes and add the annotation <b>volcano.sh/long-lifecycle-node: true</b> to them. For an unmarked node, the lifecycle plugin automatically marks the node based on the lifecycle of the load on the node.</li> <li>• The default value of <b>MaxScore</b> is <b>200.0</b>, which is twice the weight of other plugins. When the lifecycle plugin does not have obvious effect or conflicts with other plugins, disable other plugins or increase the value of <b>MaxScore</b>.</li> <li>• After the scheduler is restarted, the lifecycle plugin needs to re-record the load change. The optimal scheduling effect can be achieved only after several periods of statistics are collected.</li> </ul>

Plugin	Function	Description	Demonstration
		<p>Maximum score (equivalent to the weight) of the lifecycle plugin.</p> <ul style="list-style-type: none"> <li> <b>lifecycle.SaturatedThreshold</b>: float64 floating point number. If the value is less than 0.5, use <b>0.5</b>. If the value is greater than 1, use <b>1</b>. The default value is <b>0.8</b>. Threshold for determining whether the node usage is too high. If the node usage exceeds the threshold, the scheduler preferentially schedules jobs to other nodes.                     </li> </ul>	

Plugin	Function	Description	Demonstration
Gang	<p>Consider a group of pods as a whole for resource allocation. This plugin 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.</p> <p><b>NOTE</b> If a gang scheduling policy is used, if the remaining resources in the cluster are greater than or equal to half of the minimum number of resources for running a job but less than the minimum of resources for running the job, Autoscaler scale-outs will not be triggered.</p>	<ul style="list-style-type: none"> <li>• <b>enablePreemptable:</b> <ul style="list-style-type: none"> <li>- <b>true:</b> Preemption enabled</li> <li>- <b>false:</b> Preemption not enabled</li> </ul> </li> <li>• <b>enableJobStarving:</b> <ul style="list-style-type: none"> <li>- <b>true:</b> Resources are preempted based on the <b>minAvailable</b> setting of jobs.</li> <li>- <b>false:</b> Resources are preempted based on job replicas.</li> </ul> </li> </ul> <p><b>NOTE</b></p> <ul style="list-style-type: none"> <li>- The default value of <b>minAvailable</b> for Kubernetes-native workloads (such as Deployments) is 1. It is a good practice to set <b>enableJobStarving</b> to <b>false</b>.</li> <li>- In AI and big data scenarios, you can specify the <b>minAvailable</b> value when creating a vcjob. It is a good practice to set <b>enableJobStarving</b> to <b>true</b>.</li> <li>- In Volcano versions earlier than v1.11.5, <b>enableJobStarving</b> is set to <b>true</b> by default. In Volcano versions later than v1.11.5, <b>enableJobStarving</b> is set to <b>false</b> by default.</li> </ul>	<ul style="list-style-type: none"> <li>- plugins: <ul style="list-style-type: none"> <li>- name: priority</li> <li>- name: gang</li> <li>- enablePreemptable: false</li> <li>- enableJobStarving: false</li> <li>- name: conformance</li> </ul> </li> </ul>
priority	Schedule based on custom load priorities.	None	<ul style="list-style-type: none"> <li>- plugins: <ul style="list-style-type: none"> <li>- name: priority</li> <li>- name: gang</li> <li>- enablePreemptable: false</li> <li>- name: conformance</li> </ul> </li> </ul>

Plugin	Function	Description	Demonstration
overcommit	<p>Resources in a cluster are scheduled after being accumulated in a certain multiple to improve the workload enqueueing efficiency. If all workloads are Deployments, remove this plugin or set the raising factor to <b>2.0</b>.</p> <p><b>NOTE</b> This plugin is supported in Volcano 1.6.5 and later versions.</p>	<p><b>arguments:</b></p> <ul style="list-style-type: none"> <li>• <b>overcommit-factor:</b> inflation factor, which defaults to <b>1.2</b>.</li> </ul>	<pre>- plugins: - name: overcommit   arguments:     overcommit-factor: 2.0</pre>
drf	<p>The Dominant Resource Fairness (DRF) scheduling algorithm, which schedules jobs based on their dominant resource share. Jobs with a smaller resource share will be scheduled with a higher priority.</p>	None	<pre>- plugins: - name: 'drf' - name: 'predicates' - name: 'nodeorder'</pre>



Plugin	Function	Description	Demonstration
predicates	Determine whether a task is bound to a node by using a series of evaluation algorithms, such as node/pod affinity, taint tolerance, node repetition, volume limits, and volume zone matching.	None	<pre data-bbox="1123 297 1426 398">- plugins: - name: 'drf' - name: 'predicates' - name: 'nodeorder'</pre>

Plugin	Function	Description	Demonstration
nodeorder	A common algorithm for selecting nodes. Nodes are scored in simulated resource allocation to find the most suitable node for the current job.	<p>Scoring parameters:</p> <ul style="list-style-type: none"> <li>● <b>nodeaffinity.weight:</b> Pods are scheduled based on node affinity. This parameter defaults to <b>2</b>.</li> <li>● <b>podaffinity.weight:</b> Pods are scheduled based on pod affinity. This parameter defaults to <b>2</b>.</li> <li>● <b>leastrequested.weight:</b> Pods are scheduled to the node with the least requested resources. This parameter defaults to <b>1</b>.</li> <li>● <b>balancedresource.weight:</b> Pods are scheduled to the node with balanced resource allocation. This parameter defaults to <b>1</b>.</li> <li>● <b>mostrequested.weight:</b> Pods are scheduled to the node with the most requested resources. This parameter defaults to <b>0</b>.</li> <li>● <b>tainttoleration.weight:</b> Pods are scheduled to the node with a high taint tolerance. This parameter defaults to <b>3</b>.</li> <li>● <b>imagelocality.weight:</b> : Pods are scheduled to the node where the required images exist. This parameter defaults to <b>1</b>.</li> <li>● <b>podtopologyspread.weight:</b> Pods are</li> </ul>	<pre>- plugins: - name: nodeorder arguments:   leastrequested.weight: 1   mostrequested.weight: 0   nodeaffinity.weight: 2   podaffinity.weight: 2   balancedresource.weight: 1   1   tainttoleration.weight: 3   imagelocality.weight: 1   podtopologyspread.weight: 2</pre>

Plugin	Function	Description	Demonstration
		scheduled based on the pod topology. This parameter defaults to <b>2</b> .	
cce-gpu-topology-predicate	GPU-topology scheduling preselection algorithm	None	<ul style="list-style-type: none"> <li>- plugins:</li> <li>- <b>name: 'cce-gpu-topology-predicate'</b></li> <li>- name: 'cce-gpu-topology-priority'</li> <li>- name: 'cce-gpu'</li> </ul>
cce-gpu-topology-priority	GPU-topology scheduling priority algorithm	None	<ul style="list-style-type: none"> <li>- plugins:</li> <li>- name: 'cce-gpu-topology-predicate'</li> <li>- <b>name: 'cce-gpu-topology-priority'</b></li> <li>- name: 'cce-gpu'</li> </ul>

Plugin	Function	Description	Demonstration
cce-gpu	<p>GPU resource allocation that supports decimal GPU configurations by working with the gpu add-on.</p> <p><b>NOTE</b></p> <ul style="list-style-type: none"> <li>The plugin of version 1.10.5 or later does not support this add-on. Use xGPU instead.</li> <li>The prerequisite for configuring decimal GPUs is that the GPU nodes in the cluster are in shared mode. For details about how to check whether GPU sharing is disabled in the cluster, see the <b>enable-gpu-share</b> parameter in <a href="#">Modifying Cluster Configurations</a>.</li> </ul>	None	<pre>- plugins: - name: 'cce-gpu-topology-predicate' - name: 'cce-gpu-topology-priority' - name: 'cce-gpu'</pre>
numa-aware	<p>NUMA affinity scheduling. For details, see <a href="#">NUMA Affinity Scheduling</a>.</p>	<p><b>arguments:</b></p> <ul style="list-style-type: none"> <li><b>weight:</b> weight of the numa-aware plugin</li> </ul>	<pre>- plugins: - name: 'nodelocalvolume' - name: 'nodeemptydirvolume' - name: 'nodeCSIScheduling' - name: 'networkresource' arguments: NetworkType: vpc-router - name: <b>numa-aware</b> arguments: weight: 10</pre>

Plugin	Function	Description	Demonstration
network resource	The ENI requirement node can be preselected and filtered. The parameters are transferred by CCE and do not need to be manually configured.	<b>arguments:</b> <ul style="list-style-type: none"> <li>• <b>NetworkType:</b> network type (<b>eni</b> or <b>vpc-router</b>)</li> </ul>	<pre>- plugins: - name: 'nodelocalvolume' - name: 'nodeemptydirvolume' - name: 'nodeCSIscheduling' - name: 'networkresource' arguments: NetworkType: vpc-router</pre>
nodelocalvolume	Filter out nodes that do not meet local volume requirements.	None	<pre>- plugins: - name: 'nodelocalvolume' - name: 'nodeemptydirvolume' - name: 'nodeCSIscheduling' - name: 'networkresource'</pre>
nodeemptydirvolume	Filter out nodes that do not meet the emptyDir requirements.	None	<pre>- plugins: - name: 'nodelocalvolume' - name: 'nodeemptydirvolume' - name: 'nodeCSIscheduling' - name: 'networkresource'</pre>
nodeCSIscheduling	Filter out nodes with malfunctional Everest.	None	<pre>- plugins: - name: 'nodelocalvolume' - name: 'nodeemptydirvolume' - name: 'nodeCSIscheduling' - name: 'networkresource'</pre>

----End

## Components

Table 13-7 Add-on components

Component	Description	Resource Type
volcano-scheduler	Schedule pods.	Deployment
volcano-controller	Synchronize CRDs.	Deployment
volcano-admission	Webhook server, which verifies and modifies resources such as pods and jobs	Deployment
volcano-agent	Cloud native hybrid agent, which is used for node QoS assurance, CPU burst, and dynamic resource oversubscription	DaemonSet

Component	Description	Resource Type
resource-exporter	Report the NUMA topology information of nodes.	DaemonSet
volcano-scheduler	Reschedule pods in a cluster. After the rescheduling capability is enabled, pods will be automatically deployed on nodes.	Deployment
volcano-recommender	Generate recommendations for CPU and memory requests based on the historical CPU and memory usage of a container.	Deployment
volcano-recommender-prometheus-adapter	Collect historical CPU and memory metrics of containers from Prometheus.	Deployment

## Modifying the volcano-scheduler Configurations Using the Console

volcano-scheduler is the component responsible for pod scheduling. It consists of a series of actions and plugins. Actions should be executed in every step. Plugins provide the action algorithm details in different scenarios. volcano-scheduler is highly scalable. You can specify and implement actions and plugins based on your requirements.

After the add-on is installed, you can choose **Settings** in the navigation pane, switch to the **Scheduling** tab, and configure the basic scheduling capabilities. You can also use the expert mode of the Volcano scheduler to customize advanced scheduling policies based on service scenarios.


This section describes how to configure volcano-scheduler.

### NOTE

Only Volcano of v1.7.1 and later support this function.



Log in to the CCE console and click the cluster name to access the cluster console. In the navigation pane, choose **Settings** and click the **Scheduling** tab. In the **Select Cluster Scheduler** area, select **Volcano scheduler**, find the expert mode, and click **Refresh**.

#### Select Cluster Scheduler

Default cluster scheduler (default-scheduler) 

Kube-scheduler scheduler  Volcano scheduler

Volcano is compatible with kube-scheduler scheduling capabilities and provides incremental scheduling capabilities.

 Expert mode: allows you to create custom scheduling policies by using a configuration file if the options available on the console do not meet your service needs. [Learn More](#)  Refresh

- Using **resource\_exporter**:

```
...
  "default_scheduler_conf": {
```

```

"actions": "allocate, backfill, preempt",
"tiers": [
  {
    "plugins": [
      {
        "name": "priority"
      },
      {
        "name": "gang"
      },
      {
        "name": "conformance"
      }
    ]
  },
  {
    "plugins": [
      {
        "name": "drf"
      },
      {
        "name": "predicates"
      },
      {
        "name": "nodeorder"
      }
    ]
  },
  {
    "plugins": [
      {
        "name": "cce-gpu-topology-predicate"
      },
      {
        "name": "cce-gpu-topology-priority"
      },
      {
        "name": "cce-gpu"
      },
      {
        "name": "numa-aware" # add this also enable resource_exporter
      }
    ]
  },
  {
    "plugins": [
      {
        "name": "nodelocalvolume"
      },
      {
        "name": "nodeemptydirvolume"
      },
      {
        "name": "nodeCSIscheduling"
      },
      {
        "name": "networkresource"
      }
    ]
  }
]
},
...

```

After this function is enabled, you can use the functions of both numa-aware and resource\_exporter.

## Collecting Prometheus Metrics

volcano-scheduler exposes Prometheus metrics through port 8080. You can build a Prometheus collector to identify and obtain volcano-scheduler scheduling metrics from `http://{{volcano-schedulerPodIP}}:{{volcano-schedulerPodPort}}/metrics`.

### NOTE

Prometheus metrics can be exposed only by the Volcano add-on of version 1.8.5 or later.

**Table 13-8** Key metrics

Metric	Type	Description	Label
e2e_scheduling_latency_milliseconds	Histogram	E2E scheduling latency (ms) (scheduling algorithm + binding)	None
e2e_job_scheduling_latency_milliseconds	Histogram	E2E job scheduling latency (ms)	None
e2e_job_scheduling_duration	Gauge	E2E job scheduling duration	labels=["job_name", "queue", "job_namespace"]
plugin_scheduling_latency_microseconds	Histogram	Add-on scheduling latency ( $\mu$ s)	labels=["plugin", "OnSession"]
action_scheduling_latency_microseconds	Histogram	Action scheduling latency ( $\mu$ s)	labels=["action"]
task_scheduling_latency_milliseconds	Histogram	Task scheduling latency (ms)	None
schedule_attempts_total	Counter	Number of pod scheduling attempts. <b>unschedulable</b> indicates that the pods cannot be scheduled, and <b>error</b> indicates that the internal scheduler is faulty.	labels=["result"]
pod_preemption_victims	Gauge	Number of selected preemption victims	None
total_preemption_attempts	Counter	Total number of preemption attempts in a cluster	None
unschedule_task_count	Gauge	Number of unschedulable tasks	labels=["job_id"]
unschedule_job_count	Gauge	Number of unschedulable jobs	None



Metric	Type	Description	Label
job_retry_counts	Counter	Number of job retries	labels=["job_id"]

## Uninstalling the Volcano Add-on

After the add-on is uninstalled, all custom Volcano resources ([Table 13-9](#)) will be deleted, including the created resources. Reinstalling the add-on will not inherit or restore the tasks before the uninstallation. It is a good practice to uninstall the Volcano add-on only when no custom Volcano resources are being used in the cluster.

**Table 13-9** Custom Volcano resources

Item	API Group	API Version	Resource Level
Command	bus.volcano.sh	v1alpha1	Namespaced
Job	batch.volcano.sh	v1alpha1	Namespaced
Numatopology	nodeinfo.volcano.sh	v1alpha1	Cluster
PodGroup	scheduling.volcano.sh	v1beta1	Namespaced
Queue	scheduling.volcano.sh	v1beta1	Cluster
BalancerPolicyTemplate	autoscaling.volcano.sh	v1alpha1	Cluster
Balancer	autoscaling.volcano.sh	v1alpha1	Cluster

### NOTE

BalancerPolicyTemplate and Balancer resources are created only after the application scaling priority policies are enabled. For details, see [Application Scaling Priority Policies](#).

## Change History

### NOTICE

It is a good practice to upgrade Volcano to the latest version that is supported by the cluster.

**Table 13-10** Release history

Add-on Version	Supported Cluster Version	New Feature
1.13.3	v1.21 v1.23 v1.25 v1.27 v1.28 v1.29	<ul style="list-style-type: none"> <li>• Supported scale-in of customized resources based on node priorities.</li> <li>• Optimized the association between preemption and node scale-out.</li> </ul>
1.13.1	v1.21 v1.23 v1.25 v1.27 v1.28 v1.29	Optimized scheduler memory usage.
1.12.18	v1.21 v1.23 v1.25 v1.27 v1.28 v1.29	<ul style="list-style-type: none"> <li>• CCE clusters 1.29 are supported.</li> <li>• The preemption function is enabled by default.</li> </ul>
1.11.21	v1.19.16 v1.21 v1.23 v1.25 v1.27 v1.28	<ul style="list-style-type: none"> <li>• Supported Kubernetes 1.28.</li> <li>• Supported load-aware scheduling.</li> <li>• Updated image OS to HCE 2.0.</li> <li>• Optimized CSI resource preemption.</li> <li>• Optimized load-aware rescheduling.</li> <li>• Optimized preemption in hybrid deployment scenarios.</li> </ul>
1.11.6	v1.19.16 v1.21 v1.23 v1.25 v1.27	<ul style="list-style-type: none"> <li>• Supported Kubernetes 1.27.</li> <li>• Supported rescheduling.</li> <li>• Supported affinity scheduling of nodes in the node pool.</li> <li>• Optimized the scheduling performance.</li> </ul>
1.10.7	v1.19.16 v1.21 v1.23 v1.25	Fixes the issue that the local PV add-on fails to calculate the number of pods pre-bound to the node.

Add-on Version	Supported Cluster Version	New Feature
1.10.5	v1.19.16 v1.21 v1.23 v1.25	<ul style="list-style-type: none"> <li>• The volcano agent supports resource oversubscription.</li> <li>• Adds the verification admission for GPUs. The value of <b>nvidia.com/gpu</b> must be less than 1 or a positive integer, and the value of <b>volcano.sh/gpu-core.percentage</b> must be less than 100 and a multiple of 5.</li> <li>• Fixes the issue that pod scheduling is slow after PVC binding fails.</li> <li>• Fixes the issue that newly added pods cannot run when there are terminating pods on a node for a long time.</li> <li>• Fixes the issue that volcano restarts when creating or mounting PVCs to pods.</li> </ul>
1.9.1	v1.19.16 v1.21 v1.23 v1.25	<ul style="list-style-type: none"> <li>• Fixes the issue that the counting pipeline pod of the networkresource add-on occupies supplementary network interfaces (Sub-ENI).</li> <li>• Fixes the issue where the binpack add-on scores nodes with insufficient resources.</li> <li>• Fixes the issue of processing resources in the pod with unknown end status.</li> <li>• Optimizes event output.</li> <li>• Supports HA deployment by default.</li> </ul>
1.7.2	v1.19.16 v1.21 v1.23 v1.25	<ul style="list-style-type: none"> <li>• Adapts to clusters 1.25.</li> <li>• Improves scheduling performance of volcano.</li> </ul>
1.7.1	v1.19.16 v1.21 v1.23 v1.25	Adapts to clusters 1.25.
1.4.7	v1.15 v1.17 v1.19 v1.21	Deletes the pod status Undetermined to adapt to cluster Autoscaler.

Add-on Version	Supported Cluster Version	New Feature
1.4.5	v1.17 v1.19 v1.21	Changes the deployment mode of volcano-scheduler from <b>statefulset</b> to <b>deployment</b> , and fixes the issue that pods cannot be automatically migrated when the node is abnormal.
1.4.2	v1.15 v1.17 v1.19 v1.21	<ul style="list-style-type: none"> <li>Resolves the issue that cross-GPU allocation fails.</li> <li>Supports the updated EAS API.</li> </ul>
1.3.7	v1.15 v1.17 v1.19 v1.21	<ul style="list-style-type: none"> <li>Supports hybrid deployment of online and offline jobs and resource oversubscription.</li> <li>Optimizes the scheduling throughput for clusters.</li> <li>Fixes the issue where the scheduler panics in certain scenarios.</li> <li>Fixes the issue that the volumes.secret verification of the volcano job in the CCE clusters 1.15 fails.</li> <li>Fixes the issue that jobs fail to be scheduled when volumes are mounted.</li> </ul>
1.3.3	v1.15 v1.17 v1.19 v1.21	Fixes the scheduler crash caused by GPU exceptions and the privileged init container admission failure.
1.3.1	v1.15 v1.17 v1.19	<ul style="list-style-type: none"> <li>Upgrades the volcano framework to the latest version.</li> <li>Supported Kubernetes 1.19.</li> <li>Adds the numa-aware add-on.</li> <li>Fixes the deployment scaling issue in the multi-queue scenario.</li> <li>Adjusts the algorithm add-on enabled by default.</li> </ul>

Add-on Version	Supported Cluster Version	New Feature
1.2.5	v1.15 v1.17 v1.19	<ul style="list-style-type: none"> <li>• Fixes the OutOfcpu issue in some scenarios.</li> <li>• Fixes the issue that pods cannot be scheduled when some capabilities are set for a queue.</li> <li>• Makes the log time of the volcano component consistent with the system time.</li> <li>• Fixes the issue of preemption between multiple queues.</li> <li>• Fixes the issue that the result of the ioaware add-on does not meet the expectation in some extreme scenarios.</li> <li>• Supports hybrid clusters.</li> </ul>
1.2.3	v1.15 v1.17 v1.19	<ul style="list-style-type: none"> <li>• Fixes the training task OOM issue caused by insufficient precision.</li> <li>• Fixes the GPU scheduling issue in CCE v1.15 and later versions. Rolling upgrade of CCE versions during task distribution is not supported.</li> <li>• Fixes the issue where the queue status is unknown in certain scenarios.</li> <li>• Fixes the issue where a panic occurs when a PVC is mounted to a job in a specific scenario.</li> <li>• Fixes the issue that decimals cannot be configured for GPU jobs.</li> <li>• Adds the ioaware add-on.</li> <li>• Adds the ring controller.</li> </ul>

## 13.2.2 CCE Cluster Autoscaler

### Introduction

Autoscaler is an important Kubernetes controller. It supports microservice scaling and is key to serverless design.

When the CPU or memory usage of a microservice is too high, horizontal pod autoscaling is triggered to add pods to reduce the load. These pods can be automatically reduced when the load is low, allowing the microservice to run as efficiently as possible.

CCE simplifies the creation, upgrade, and manual scaling of Kubernetes clusters, in which traffic loads change over time. To balance resource usage and workload performance of nodes, Kubernetes introduces the Autoscaler add-on to automatically adjust the number of nodes a cluster based on the resource usage required for workloads deployed in the cluster. For details, see [Creating a Node Scaling Policy](#).

Open source community: <https://github.com/kubernetes/autoscaler>

## How the Add-on Works

Autoscaler controls auto scale-out and scale-in.

- **Auto scale-out**

You can choose either of the following methods:

- If a pod cannot be scheduled due to insufficient resources of worker nodes, CCE will add more nodes to the cluster. The new nodes have the same resource quotas as those configured for the node pools that the new nodes are in.

Auto scale-out will be performed when:

- Node resources are insufficient.
- No node affinity policy is set in the scheduling configurations of the pod. If the pod is configured affinity for a node, the system will not automatically add more nodes in the cluster. For details about how to configure node affinity policies, see [Configuring Node Affinity Scheduling \(nodeAffinity\)](#).
- When the cluster meets the node scaling policy, cluster scale-out is also triggered. For details, see [Creating a Node Scaling Policy](#).

### NOTE

The add-on follows the "No Less, No More" policy. For example, if three cores are required for creating a pod and the system supports four-core and eight-core nodes, Autoscaler will preferentially create a four-core node.

- **Auto scale-in**

When a cluster node is idle for a period of time (10 minutes by default), cluster scale-in is triggered, and the node is automatically deleted. However, a node cannot be deleted from a cluster if the following pods exist:

- Pods that do not meet specific requirements set in Pod Disruption Budgets ([PodDisruptionBudget](#))
- Pods that cannot be scheduled to other nodes due to constraints such as affinity and anti-affinity policies
- Pods that have the **cluster-autoscaler.kubernetes.io/safe-to-evict: 'false'** annotation
- Pods (except those created by DaemonSets in the kube-system namespace) that exist in the kube-system namespace on the node
- Pods that are not created by the controller (Deployment/ReplicaSet/job/StatefulSet)

 NOTE

When a node meets the scale-in conditions, Autoscaler adds the **DeletionCandidateOfClusterAutoscaler** taint to the node in advance to prevent pods from being scheduled to the node. After the Autoscaler add-on is uninstalled, if the taint still exists on the node, manually delete it.

## Notes and Constraints

- Ensure that there are sufficient resources for installing the add-on.
- The default node pool does not support auto scaling. For details, see [Description of DefaultPool](#).
- When Autoscaler is used, some taints or annotations may affect auto scaling. Therefore, do not use the following taints or annotations in clusters:
  - **ignore-taint.cluster-autoscaler.kubernetes.io**: The taint works on nodes. Kubernetes-native Autoscaler supports protection against abnormal scale outs and periodically evaluates the proportion of available nodes in the cluster. When the proportion of non-ready nodes exceeds 45%, protection will be triggered. In this case, all nodes with the **ignore-taint.cluster-autoscaler.kubernetes.io** taint in the cluster are filtered out from the Autoscaler template and recorded as non-ready nodes, which affects cluster scaling.
  - **cluster-autoscaler.kubernetes.io/enable-ds-eviction**: The annotation works on pods, which determines whether DaemonSet pods can be evicted by Autoscaler. For details, see [Well-Known Labels, Annotations and Taints](#).

## Installing the Add-on

**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 **CCE Cluster Autoscaler** on the right, and click **Install**.

**Step 2** On the **Install Add-on** page, configure the specifications as needed.

There are three types of **preset specifications** based on the cluster scale. You can select one as required. The system will configure the number of pods and resource quotas for the add-on based on the selected preset specifications. You can see the configurations on the console.

**Step 3** Configure deployment policies for the add-on pods.

 NOTE

- Scheduling policies do not take effect on add-on instances of the DaemonSet type.
- When configuring multi-AZ deployment or node affinity, ensure that there are nodes meeting the scheduling policy and that resources are sufficient in the cluster. Otherwise, the add-on cannot run.

**Table 13-11** Configurations for add-on scheduling

Parameter	Description
Multi AZ	<ul style="list-style-type: none"> <li>● <b>Preferred:</b> Deployment pods of the add-on will be preferentially scheduled to nodes in different AZs. If all the nodes in the cluster are deployed in the same AZ, the pods will be scheduled to different nodes in that AZ.</li> <li>● <b>Equivalent mode:</b> Deployment pods of the add-on are evenly scheduled to the nodes in the cluster in each AZ. If a new AZ is added, you are advised to increase add-on pods for cross-AZ HA deployment. With the Equivalent multi-AZ deployment, the difference between the number of add-on pods in different AZs will be less than or equal to 1. If resources in one of the AZs are insufficient, pods cannot be scheduled to that AZ.</li> <li>● <b>Required:</b> Deployment pods of the add-on are forcibly scheduled to nodes in different AZs. There can be at most one pod in each AZ. If nodes in a cluster are not in different AZs, some add-on pods cannot run properly. If a node is faulty, add-on pods on it may fail to be migrated.</li> </ul>
Node Affinity	<ul style="list-style-type: none"> <li>● <b>Not configured:</b> Node affinity is disabled for the add-on.</li> <li>● <b>Node Affinity:</b> Specify the nodes where the add-on is deployed. If you do not specify the nodes, the add-on will be randomly scheduled based on the default cluster scheduling policy.</li> <li>● <b>Specified Node Pool Scheduling:</b> Specify the node pool where the add-on is deployed. If you do not specify the node pool, the add-on will be randomly scheduled based on the default cluster scheduling policy.</li> <li>● <b>Custom Policies:</b> Enter the labels of the nodes where the add-on is to be deployed for more flexible scheduling policies. If you do not specify node labels, the add-on will be randomly scheduled based on the default cluster scheduling policy. If multiple custom affinity policies are configured, ensure that there are nodes that meet all the affinity policies in the cluster. Otherwise, the add-on cannot run.</li> </ul>
Toleration	<p>Using both taints and tolerations allows (not forcibly) the add-on Deployment to be scheduled to a node with the matching taints, and controls the Deployment eviction policies after the node where the Deployment is located is tainted.</p> <p>The add-on adds the default tolerance policy for the <b>node.kubernetes.io/not-ready</b> and <b>node.kubernetes.io/unreachable</b> taints, respectively. The tolerance time window is 60s.</p> <p>For details, see <a href="#">Configuring Tolerance Policies</a>.</p>



**Step 4** After the configuration is complete, click **Install**.

----End

## Components

**Table 13-12** Add-on components

Component	Description	Resource Type
Autoscaler	Auto scaling for Kubernetes clusters	Deployment

## Change History

**Table 13-13** Release history for add-on adapted to clusters 1.29

Add-on Version	Supported Cluster Version	New Feature	Community Version
1.29.17	v1.29	Optimized events.	<a href="#">1.29.1</a>
1.29.13	v1.29	Clusters 1.29 are supported.	<a href="#">1.29.1</a>

**Table 13-14** Release history for add-on adapted to clusters 1.28

Add-on Version	Supported Cluster Version	New Feature	Community Version
1.28.55	v1.28	Optimized events.	<a href="#">1.28.1</a>
1.28.51	v1.28	Optimized the logic for generating alarms when resources in a node pool are sold out.	<a href="#">1.28.1</a>
1.28.22	v1.28	Fixed some issues.	<a href="#">1.28.1</a>
1.28.20	v1.28	Fixed some issues.	<a href="#">1.28.1</a>
1.28.17	v1.28	Fixed the issue that scale-in cannot be performed when there are custom pod controllers in a cluster.	<a href="#">1.28.1</a>

**Table 13-15** Release history for add-on adapted to clusters 1.27

Add-on Version	Supported Cluster Version	New Feature	Community Version
1.27.88	v1.27	Optimized events.	<a href="#">1.27.1</a>
1.27.84	v1.27	Optimized the logic for generating alarms when resources in a node pool are sold out.	<a href="#">1.27.1</a>
1.27.55	v1.27	Fixed some issues.	<a href="#">1.27.1</a>
1.27.51	v1.27	Fixed some issues.	<a href="#">1.27.1</a>
1.27.14	v1.27	Fixed the scale-in failure of nodes of different specifications in the same node pool and unexpected <b>PreferNoSchedule</b> taint issues.	<a href="#">1.27.1</a>

**Table 13-16** Release history for add-on adapted to clusters 1.25

Add-on Version	Supported Cluster Version	New Feature	Community Version
1.25.120	v1.25	Optimized events.	<a href="#">1.25.0</a>
1.25.116	v1.25	Optimized the logic for generating alarms when resources in a node pool are sold out.	<a href="#">1.25.0</a>
1.25.88	v1.25	Fixed some issues.	<a href="#">1.25.0</a>
1.25.84	v1.25	Fixed some issues.	<a href="#">1.25.0</a>
1.25.34	v1.25	<ul style="list-style-type: none"> <li>Optimized the method of identifying GPUs and NPUs.</li> <li>Used the remaining node quota of a cluster for the extra nodes that are beyond the cluster scale.</li> </ul>	<a href="#">1.25.0</a>

Add-on Version	Supported Cluster Version	New Feature	Community Version
1.25.21	v1.25	<ul style="list-style-type: none"> <li>● Fixed the issue that the autoscaler's least-waste is disabled by default.</li> <li>● Fixed the issue that the node pool cannot be switched to another pool for scaling out after a scale-out failure and the add-on has to restart.</li> <li>● The default taint tolerance duration is changed to 60s.</li> <li>● Fixed the issue that scale-out is still triggered after the scale-out rule is disabled.</li> </ul>	<a href="#">1.25.0</a>
1.25.11	v1.25	<ul style="list-style-type: none"> <li>● Supported anti-affinity scheduling of add-on pods on nodes in different AZs.</li> <li>● Added the tolerance time during which the pods with temporary storage volumes cannot be scheduled.</li> <li>● Fixed the issue that the number of node pools cannot be restored when AS group resources are insufficient.</li> </ul>	<a href="#">1.25.0</a>
1.25.7	v1.25	<ul style="list-style-type: none"> <li>● CCE clusters 1.25 are supported.</li> <li>● Modified the memory request and limit of a customized flavor.</li> <li>● Enabled to report an event indicating that scaling cannot be performed in a node pool with auto scaling disabled.</li> <li>● Fixed the bug that NPU node scale-out is triggered again during scale-out.</li> </ul>	<a href="#">1.25.0</a>

**Table 13-17** Release history for add-on adapted to clusters 1.23

Add-on Version	Supported Cluster Version	New Feature	Community Version
1.23.125	v1.23	Optimized events.	<a href="#">1.23.0</a>
1.23.121	1.23	Optimized the logic for generating alarms when resources in a node pool are sold out.	<a href="#">1.23.0</a>
1.23.95	v1.23	Fixed some issues.	<a href="#">1.23.0</a>
1.23.91	v1.23	Fixed some issues.	<a href="#">1.23.0</a>
1.23.54	v1.23	Fixed the scale-in failure of nodes of different specifications in the same node pool and unexpected <b>PreferNoSchedule</b> taint issues.	<a href="#">1.23.0</a>
1.23.44	v1.23	<ul style="list-style-type: none"> <li>Optimized the method of identifying GPUs and NPUs.</li> <li>Used the remaining node quota of a cluster for the extra nodes that are beyond the cluster scale.</li> </ul>	<a href="#">1.23.0</a>
1.23.31	v1.23	<ul style="list-style-type: none"> <li>Fixed the issue that the autoscaler's least-waste is disabled by default.</li> <li>Fixed the issue that the node pool cannot be switched to another pool for scaling out after a scale-out failure and the add-on has to restart.</li> <li>The default taint tolerance duration is changed to 60s.</li> <li>Fixed the issue that scale-out is still triggered after the scale-out rule is disabled.</li> </ul>	<a href="#">1.23.0</a>
1.23.21	v1.23	<ul style="list-style-type: none"> <li>Supported anti-affinity scheduling of add-on pods on nodes in different AZs.</li> <li>Added the tolerance time during which the pods with temporary storage volumes cannot be scheduled.</li> <li>Fixed the issue that the number of node pools cannot be restored when AS group resources are insufficient.</li> </ul>	<a href="#">1.23.0</a>

Add-on Version	Supported Cluster Version	New Feature	Community Version
1.23.17	v1.23	<ul style="list-style-type: none"> <li>Supported NPUs and security containers.</li> <li>Supported node scaling policies without a step.</li> <li>Fixed a bug so that deleted node pools are automatically removed.</li> <li>Supported scheduling by priority.</li> <li>Supported the emptyDir scheduling policy.</li> <li>Fixed a bug so that scale-in can be triggered on the nodes whose capacity is lower than the scale-in threshold when the node scaling policy is disabled.</li> <li>Modified the memory request and limit of a customized flavor.</li> <li>Enabled to report an event indicating that scaling cannot be performed in a node pool with auto scaling disabled.</li> <li>Fixed the bug that NPU node scale-out is triggered again during scale-out.</li> </ul>	<a href="#">1.23.0</a>
1.23.10	v1.23	<ul style="list-style-type: none"> <li>Optimized logging.</li> <li>Supported scale-in waiting so that operations such as data dump can be performed before a node is deleted.</li> </ul>	<a href="#">1.23.0</a>
1.23.9	v1.23	Added the <b>nodenetworkconfigs.crd.yangtse.cni</b> resource object permission.	<a href="#">1.23.0</a>
1.23.8	v1.23	Fixed the issue that scale-out fails when the number of nodes to be added at a time exceeds the upper limit in periodic scale-outs.	<a href="#">1.23.0</a>
1.23.7	v1.23		<a href="#">1.23.0</a>
1.23.3	v1.23	CCE clusters 1.23 are supported.	<a href="#">1.23.0</a>

**Table 13-18** Release history for add-on adapted to clusters 1.21

Add-on Version	Supported Cluster Version	New Feature	Community Version
1.21.114	v1.21	Optimized the logic for generating alarms when resources in a node pool are sold out.	<a href="#">1.21.0</a>
1.21.89	v1.21	Fixed some issues.	<a href="#">1.21.0</a>
1.21.86	v1.21	Fixed the issue that the node pool auto scaling cannot meet expectations after AZ topology constraints are configured for nodes.	<a href="#">1.21.0</a>
1.21.51	v1.21	Fixed the scale-in failure of nodes of different specifications in the same node pool and unexpected <b>PreferNoSchedule</b> taint issues.	<a href="#">1.21.0</a>
1.21.43	v1.21	<ul style="list-style-type: none"> <li>Optimized the method of identifying GPUs and NPUs.</li> <li>Used the remaining node quota of a cluster for the extra nodes that are beyond the cluster scale.</li> </ul>	<a href="#">1.21.0</a>
1.21.29	v1.21	<ul style="list-style-type: none"> <li>Supported anti-affinity scheduling of add-on pods on nodes in different AZs.</li> <li>Added the tolerance time during which the pods with temporary storage volumes cannot be scheduled.</li> <li>Fixed the issue that the number of node pools cannot be restored when scaling group resources are insufficient.</li> <li>Fixed the issue that the node pool cannot be switched to another pool for scaling out after a scale-out failure and the add-on has to restart.</li> <li>The default taint tolerance duration is changed to 60s.</li> <li>Fixed the issue that scale-out is still triggered after the scale-out rule is disabled.</li> </ul>	<a href="#">1.21.0</a>

Add-on Version	Supported Cluster Version	New Feature	Community Version
1.21.20	v1.21	<ul style="list-style-type: none"> <li>Supported anti-affinity scheduling of add-on pods on nodes in different AZs.</li> <li>Added the tolerance time during which the pods with temporary storage volumes cannot be scheduled.</li> <li>Fixed the issue that the number of node pools cannot be restored when scaling group resources are insufficient.</li> </ul>	<a href="#">1.21.0</a>
1.21.16	v1.21	<ul style="list-style-type: none"> <li>Supported NPUs and security containers.</li> <li>Supported node scaling policies without a step.</li> <li>Fixed a bug so that deleted node pools are automatically removed.</li> <li>Supported scheduling by priority.</li> <li>Supported the emptyDir scheduling policy.</li> <li>Fixed a bug so that scale-in can be triggered on the nodes whose capacity is lower than the scale-in threshold when the node scaling policy is disabled.</li> <li>Modified the memory request and limit of a customized flavor.</li> <li>Enabled to report an event indicating that scaling cannot be performed in a node pool with auto scaling disabled.</li> <li>Fixed the bug that NPU node scale-out is triggered again during scale-out.</li> </ul>	<a href="#">1.21.0</a>
1.21.9	v1.21	<ul style="list-style-type: none"> <li>Optimized logging.</li> <li>Supported scale-in waiting so that operations such as data dump can be performed before a node is deleted.</li> </ul>	<a href="#">1.21.0</a>

Add-on Version	Supported Cluster Version	New Feature	Community Version
1.21.8	v1.21	Added the <b>nodenetworkconfigs.crd.yangtse.cni</b> resource object permission.	<a href="#">1.21.0</a>
1.21.6	v1.21	Fixed the issue that authentication fails due to incorrect signature in the add-on request retries.	<a href="#">1.21.0</a>
1.21.4	v1.21	Fixed the issue that authentication fails due to incorrect signature in the add-on request retries.	<a href="#">1.21.0</a>
1.21.2	v1.21	Fixed the issue that auto scaling may be blocked due to a failure in deleting an unregistered node.	<a href="#">1.21.0</a>
1.21.1	v1.21	Fixed the issue that the node pool modification in the existing periodic auto scaling rule does not take effect.	<a href="#">1.21.0</a>

**Table 13-19** Release history for add-on adapted to clusters 1.19

Add-on Version	Supported Cluster Version	New Feature	Community Version
1.19.56	v1.19	Fixed the scale-in failure of nodes of different specifications in the same node pool and unexpected <b>PreferNoSchedule</b> taint issues.	<a href="#">1.19.0</a>
1.19.48	v1.19	<ul style="list-style-type: none"> <li>Optimized the method of identifying GPUs and NPUs.</li> <li>Used the remaining node quota of a cluster for the extra nodes that are beyond the cluster scale.</li> </ul>	<a href="#">1.19.0</a>



Add-on Version	Supported Cluster Version	New Feature	Community Version
1.19.35	v1.19	<ul style="list-style-type: none"> <li>● Supported anti-affinity scheduling of add-on pods on nodes in different AZs.</li> <li>● Added the tolerance time during which the pods with temporary storage volumes cannot be scheduled.</li> <li>● Fixed the issue that the number of node pools cannot be restored when scaling group resources are insufficient.</li> <li>● Fixed the issue that the node pool cannot be switched to another pool for scaling out after a scale-out failure and the add-on has to restart.</li> <li>● The default taint tolerance duration is changed to 60s.</li> <li>● Fixed the issue that scale-out is still triggered after the scale-out rule is disabled.</li> </ul>	<a href="#">1.19.0</a>
1.19.27	v1.19	<ul style="list-style-type: none"> <li>● Supported anti-affinity scheduling of add-on pods on nodes in different AZs.</li> <li>● Added the tolerance time during which the pods with temporary storage volumes cannot be scheduled.</li> <li>● Fixed the issue that the number of node pools cannot be restored when scaling group resources are insufficient.</li> </ul>	<a href="#">1.19.0</a>

Add-on Version	Supported Cluster Version	New Feature	Community Version
1.19.22	v1.19	<ul style="list-style-type: none"> <li>• Supported NPUs and security containers.</li> <li>• Supported node scaling policies without a step.</li> <li>• Fixed a bug so that deleted node pools are automatically removed.</li> <li>• Supported scheduling by priority.</li> <li>• Supported the emptyDir scheduling policy.</li> <li>• Fixed a bug so that scale-in can be triggered on the nodes whose capacity is lower than the scale-in threshold when the node scaling policy is disabled.</li> <li>• Modified the memory request and limit of a customized flavor.</li> <li>• Enabled to report an event indicating that scaling cannot be performed in a node pool with auto scaling disabled.</li> <li>• Fixed the bug that NPU node scale-out is triggered again during scale-out.</li> </ul>	<a href="#">1.19.0</a>
1.19.14	v1.19	<ul style="list-style-type: none"> <li>• Optimized logging.</li> <li>• Supported scale-in waiting so that operations such as data dump can be performed before a node is deleted.</li> </ul>	<a href="#">1.19.0</a>
1.19.13	v1.19	Fixed the issue that scale-out fails when the number of nodes to be added at a time exceeds the upper limit in periodic scale-outs.	<a href="#">1.19.0</a>
1.19.12	v1.19	Fixed the issue that authentication fails due to incorrect signature in the add-on request retries.	<a href="#">1.19.0</a>
1.19.11	v1.19	Fixed the issue that authentication fails due to incorrect signature in the add-on request retries.	<a href="#">1.19.0</a>

Add-on Version	Supported Cluster Version	New Feature	Community Version
1.19.9	v1.19	Fixed the issue that auto scaling may be blocked due to a failure in deleting an unregistered node.	<a href="#">1.19.0</a>
1.19.8	v1.19	Fixed the issue that the node pool modification in the existing periodic auto scaling rule does not take effect.	<a href="#">1.19.0</a>
1.19.7	v1.19	Regular upgrade of add-on dependencies	<a href="#">1.19.0</a>
1.19.6	v1.19	Fixed the issue that repeated scale-out is triggered when taints are asynchronously updated.	<a href="#">1.19.0</a>
1.19.3	v1.19	Supports scheduled scaling policies based on the total number of nodes, CPU limit, and memory limit. Fixes other functional defects.	<a href="#">1.19.0</a>

**Table 13-20** Release history for add-on adapted to clusters 1.17

Add-on Version	Supported Cluster Version	New Feature	Community Version
1.17.27	v1.17	<ul style="list-style-type: none"> <li>● Optimized logging.</li> <li>● Fixed a bug so that deleted node pools are automatically removed.</li> <li>● Supported scheduling by priority.</li> <li>● Fixed the issue that taints on newly added nodes are overwritten.</li> <li>● Fixed a bug so that scale-in can be triggered on the nodes whose capacity is lower than the scale-in threshold when the node scaling policy is disabled.</li> <li>● Modified the memory request and limit of a customized flavor.</li> <li>● Enabled to report an event indicating that scaling cannot be performed in a node pool with auto scaling disabled.</li> </ul>	<a href="#">1.17.0</a>
1.17.22	v1.17	Optimized logging.	<a href="#">1.17.0</a>
1.17.21	v1.17	Fixed the issue that scale-out fails when the number of nodes to be added at a time exceeds the upper limit in periodic scale-outs.	<a href="#">1.17.0</a>
1.17.19	v1.17	Fixed the issue that authentication fails due to incorrect signature in the add-on request retries.	<a href="#">1.17.0</a>
1.17.17	v1.17	Fixed the issue that auto scaling may be blocked due to a failure in deleting an unregistered node.	<a href="#">1.17.0</a>
1.17.16	v1.17	Fixed the issue that the node pool modification in the existing periodic auto scaling rule does not take effect.	<a href="#">1.17.0</a>
1.17.15	v1.17	Unified resource specification configuration unit.	<a href="#">1.17.0</a>

Add-on Version	Supported Cluster Version	New Feature	Community Version
1.17.14	v1.17	Fixed the issue that repeated scale-out is triggered when taints are asynchronously updated.	<a href="#">1.17.0</a>
1.17.8	v1.17	Fixed bugs.	<a href="#">1.17.0</a>
1.17.7	v1.17	Added log content and fixed bugs.	<a href="#">1.17.0</a>
1.17.5	v1.17	Supported clusters 1.17 and allowed scaling events to be displayed on the CCE console.	<a href="#">1.17.0</a>
1.17.2	v1.17	Clusters 1.17 are supported.	<a href="#">1.17.0</a>

### 13.2.3 CCE Advanced HPA

CCE Advanced HPA (cce-hpa-controller) is a CCE-developed add-on, which can be used to flexibly scale in or out Deployments based on metrics such as CPU usage and memory usage.

#### Main Functions

- Scaling can be performed based on the percentage of the current number of pods.
- The minimum scaling step can be set.
- Different scaling operations can be performed based on the actual metric values.

#### Notes and Constraints

- If the add-on version is earlier than 1.2.11, the [Prometheus](#) add-on must be installed. If the add-on version is 1.2.11 or later, the add-ons that can provide metrics API must be installed. You can select one of the following add-ons based on your cluster version and requirements.
  - [Kubernetes Metrics Server](#): provides basic resource usage metrics, such as container CPU and memory usage. It is supported by all cluster versions.
  - [Cloud Native Cluster Monitoring](#): available only in clusters of v1.17 or later.
    - Auto scaling based on basic resource metrics: Prometheus needs to be registered as a metrics API. For details, see [Providing Resource Metrics Through the Metrics API](#).
    - Auto scaling based on custom metrics: Custom metrics need to be aggregated to the Kubernetes API server. For details, see [Creating an HPA Policy Using Custom Metrics](#).

- **Prometheus** : Prometheus needs to be registered as a metrics API. For details, see [Providing Resource Metrics Through the Metrics API](#). This add-on supports only clusters of v1.21 or earlier.

## Installing the Add-on

**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 **CCE Advanced HPA** on the right, and click **Install**.

**Step 2** On the **Install Add-on** page, configure the specifications as needed.

- If you selected **Preset**, the add-on specifications will be automatically configured based on the recommended values by CCE. These values are suitable for most scenarios and can be viewed on the console.
- If you selected **Custom**, you can modify the number of replicas, CPUs, and memory of each add-on component as required.

### NOTE

**Replicas:** HA is not possible with just one replica, so one replica is used only for verification. In commercial scenarios, you can configure multiple replicas based on the cluster specifications.

**CPU Quota and Memory Quota:** The resource quotas of a component are affected by how many containers and scaling policies in a cluster. For typical situations, it is recommended that you configure 500m CPU cores and 1,000 MiB of memory for every 5,000 containers in a cluster. As for scaling policies, 100m CPU cores and 500 MiB of memory should be configured for every 1,000 of them.

**Step 3** Configure deployment policies for the add-on pods.

### NOTE

- Scheduling policies do not take effect on add-on instances of the DaemonSet type.
- When configuring multi-AZ deployment or node affinity, ensure that there are nodes meeting the scheduling policy and that resources are sufficient in the cluster. Otherwise, the add-on cannot run.

**Table 13-21** Configurations for add-on scheduling

Parameter	Description
Multi AZ	<ul style="list-style-type: none"> <li>● <b>Preferred:</b> Deployment pods of the add-on will be preferentially scheduled to nodes in different AZs. If all the nodes in the cluster are deployed in the same AZ, the pods will be scheduled to different nodes in that AZ.</li> <li>● <b>Equivalent mode:</b> Deployment pods of the add-on are evenly scheduled to the nodes in the cluster in each AZ. If a new AZ is added, you are advised to increase add-on pods for cross-AZ HA deployment. With the Equivalent multi-AZ deployment, the difference between the number of add-on pods in different AZs will be less than or equal to 1. If resources in one of the AZs are insufficient, pods cannot be scheduled to that AZ.</li> <li>● <b>Required:</b> Deployment pods of the add-on are forcibly scheduled to nodes in different AZs. There can be at most one pod in each AZ. If nodes in a cluster are not in different AZs, some add-on pods cannot run properly. If a node is faulty, add-on pods on it may fail to be migrated.</li> </ul>
Node Affinity	<ul style="list-style-type: none"> <li>● <b>Not configured:</b> Node affinity is disabled for the add-on.</li> <li>● <b>Node Affinity:</b> Specify the nodes where the add-on is deployed. If you do not specify the nodes, the add-on will be randomly scheduled based on the default cluster scheduling policy.</li> <li>● <b>Specified Node Pool Scheduling:</b> Specify the node pool where the add-on is deployed. If you do not specify the node pool, the add-on will be randomly scheduled based on the default cluster scheduling policy.</li> <li>● <b>Custom Policies:</b> Enter the labels of the nodes where the add-on is to be deployed for more flexible scheduling policies. If you do not specify node labels, the add-on will be randomly scheduled based on the default cluster scheduling policy. If multiple custom affinity policies are configured, ensure that there are nodes that meet all the affinity policies in the cluster. Otherwise, the add-on cannot run.</li> </ul>
Toleration	<p>Using both taints and tolerations allows (not forcibly) the add-on Deployment to be scheduled to a node with the matching taints, and controls the Deployment eviction policies after the node where the Deployment is located is tainted.</p> <p>The add-on adds the default tolerance policy for the <b>node.kubernetes.io/not-ready</b> and <b>node.kubernetes.io/unreachable</b> taints, respectively. The tolerance time window is 60s.</p> <p>For details, see <a href="#">Configuring Tolerance Policies</a>.</p>

**Step 4** Click **Install**.

----End

## Components

**Table 13-22** Add-on components

Component	Description	Resource Type
customedhpa-controller	CCE auto scaling component, which scales in or out Deployments based on metrics such as CPU usage and memory usage	Deployment

## Change History

**Table 13-23** Release history

Add-on Version	Supported Cluster Version	New Feature
1.4.3	v1.21 v1.23 v1.25 v1.27 v1.28 v1.29	Fixed some issues.
1.4.2	v1.21 v1.23 v1.25 v1.27 v1.28 v1.29	CCE clusters 1.29 are supported.
1.3.42	v1.21 v1.23 v1.25 v1.27 v1.28	CCE clusters 1.28 are supported.



Add-on Version	Supported Cluster Version	New Feature
1.3.14	v1.19 v1.21 v1.23 v1.25 v1.27	CCE clusters 1.27 are supported.
1.3.10	v1.19 v1.21 v1.23 v1.25	Periodic scaling is not affected by the cooldown period.
1.3.7	v1.19 v1.21 v1.23 v1.25	Supported anti-affinity scheduling of add-on pods on nodes in different AZs.
1.3.3	v1.19 v1.21 v1.23 v1.25	<ul style="list-style-type: none"> <li>• CCE clusters 1.25 are supported.</li> <li>• Allowed CronHPA to adjust the number of Deployment pods with the skip scenario supported.</li> </ul>
1.3.1	v1.19 v1.21 v1.23	CCE clusters 1.23 are supported.
1.2.12	v1.15 v1.17 v1.19 v1.21	Optimizes the add-on performance to reduce resource consumption.
1.2.11	v1.15 v1.17 v1.19 v1.21	<ul style="list-style-type: none"> <li>• Enables the Kubernetes metrics API to obtain resource metrics.</li> <li>• Takes not-ready pods into consideration when calculating resource usage.</li> </ul>
1.2.10	v1.15 v1.17 v1.19 v1.21	CCE clusters 1.21 are supported.

Add-on Version	Supported Cluster Version	New Feature
1.2.4	v1.15 v1.17 v1.19	<ul style="list-style-type: none"><li>• Regular upgrade of add-on dependencies</li><li>• Allows custom add-on resource specifications.</li></ul>
1.2.3	v1.15 v1.17 v1.19	Supports ARM64 nodes.
1.2.2	v1.15 v1.17 v1.19	Enhances the health check function.
1.2.1	v1.15 v1.17 v1.19	<ul style="list-style-type: none"><li>• CCE clusters 1.19 are supported.</li><li>• Updates the add-on to a stable version.</li></ul>
1.1.3	v1.15 v1.17	Supports periodic scaling rules.

## 13.3 Cloud Native Observability Add-ons

### 13.3.1 Cloud Native Cluster Monitoring

#### Introduction

Cloud Native Cluster Monitoring (kube-prometheus-stack) uses Prometheus-operator and Prometheus to provide easy-to-use, end-to-end Kubernetes cluster monitoring.

This add-on allows the monitoring data to be interconnected with Container Intelligent Analysis (CIA) so that you can view monitoring data and configure alarms on the CIA console.

Open source community: <https://github.com/prometheus/prometheus>

#### Notes and Constraints

- By default, the kube-state-metrics component of the add-on does not collect labels and annotations of Kubernetes resources. To collect these labels and annotations, manually enable the collection function in the startup parameters and check whether the corresponding metrics are added to the collection trustlist of ServiceMonitor named **kube-state-metrics**. For details, see [Collecting All Labels and Annotations of a Pod](#).

- In 3.8.0 and later versions, component metrics in the kube-system and monitoring namespaces are not collected by default. If you have workloads in the two namespaces, use **Pod Monitor** or **Service Monitor** to collect these metrics.
- In 3.8.0 and later versions, etcd-server, kube-controller, kube-scheduler, autoscaler, fluent-bit, volcano-agent, volcano-scheduler and otel-collector metrics are not collected by default. Enable the collection as required.

To enable this function, on the **ConfigMaps and Secrets** page, expand the dropdown list of **Namespace**, and select **monitoring**. Locate the row that contains the configuration item named **persistent-user-config**, and click **Edit YAML** in the operation column. Remove the **serviceMonitorDisable** or **podMonitorDisable** configuration in the **customSettings** field as required or set the configuration to an empty array.

```
...
customSettings:
  podMonitorDisable: []
  serviceMonitorDisable: []
```

## Permissions

The node-exporter component of the Cloud Native Cluster Monitoring add-on needs to read the Docker info data from the **/var/run/docker.sock** directory on the host for monitoring the Docker disk space.

The following permission is required for running node-exporter:

- **cap\_dac\_override**: reads the Docker info data.

## Installing the Add-on

### NOTE

The Cloud Native Cluster Monitoring add-on automatically selects a deployment mode based on **Data Storage Configuration**. This is supported by Cloud Native Cluster Monitoring 3.7.1 or later.

- Original agent mode: Disable **Local Data Storage** and enable at least one of **Report Monitoring Data to AOM** and **Report Monitoring Data to a Third-Party Platform**.
- Original server mode: Enable **Local Data Storage** and **Report Monitoring Data to AOM** or **Report Monitoring Data to a Third-Party Platform**.

**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 **Cloud Native Cluster Monitoring** on the right, and click **Install**.

**Step 2** On the **Install Add-on** page, enable at least one item in the **Data Storage Configuration** area.

- **Reporting Monitoring Data to AOM**: Report Prometheus data to AOM. After this function is enabled, you can select the corresponding AOM instance. The collected basic metrics are free of charge. Custom metrics are charged by AOM. To interconnect with AOM, you must have certain permissions. Only **Huawei Cloud accounts, HUAWEI IDs, and users in the admin user group** can perform this operation.
- **Reporting Monitoring Data to a Third-Party Monitoring Platform**: To report Prometheus data to a third-party monitoring system, you need to enter

the address and token of the third-party monitoring system and determine whether to skip certificate authentication.

- **Local data storage:** Select the type and size of a disk for storing monitoring data to store Prometheus data in PVCs in the cluster. **Storage volumes are not deleted along with the add-on.** If local data storage is enabled, all components will be deployed. For details, see [Components](#).

 **NOTE**

An available PVC named **pvc-prometheus-server** exists in namespace **monitoring** and will be used as the storage source.

**Step 3** Configure the add-on **specifications** as needed.

- **Add-on Specifications:**
  - If you selected **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 selected **Custom**, you can adjust the number of pods and resource quotas as needed. High availability is not possible with a single pod. If an error occurs on the node where the add-on instance runs, the add-on will fail.
- **Prometheus HA:** The Prometheus-server, Prometheus-operator, thanos-query, custom-metrics-apiserver, alertmanager, and kube-state-metrics components are deployed in multi-instance mode in the cluster.
- **Number of collected shards** (available after **Local data storage** is disabled): When there is a lot of Prometheus data, you can configure this parameter to spread the data across a specific number of Prometheus instances. This will help with storage and querying. Increasing the number of shards reduces the data volume carried by each shard. This can increase the upper limit of the metric collection throughput and cause more resources to be consumed. You are advised to increase shards when the cluster scale is large to improve the collection performance. In addition, you need to consider the impact of resource usage and optimize it based on specific scenarios.
- **Install Grafana:** Use Grafana to visualize monitoring data. Grafana creates a 5 GiB storage volume by default. Uninstalling the add-on **will not delete this volume**. The default username and password for the first login are **admin**. You will be asked to change the password immediately after login.

**Step 4** Configure the add-on parameters.

- **Custom Metric Collection:** Application metrics are automatically collected in the form of service discovery. After this function is enabled, you need to add related configurations to the target application. For details, see [Monitoring Custom Metrics Using Cloud Native Cluster Monitoring](#).
- **Collection Interval:** Configure the collection interval.
- **Data Retention** (available only after **Local data storage** is enabled): Configure how long the monitoring data can be kept.
- **Listening port of node-exporter:** This port uses the host network to listen to and expose metrics on the node for Prometheus collection. The default port number is 9100, but it can be modified if there is a conflict with an existing port.

**Step 5 Click Install.**

After the add-on is installed, you may need to perform the following operations:

- To use custom metrics to create an auto scaling policy, ensure that local data storage is enabled for the add-on and then take the following steps:
  - a. Collect custom metrics reported by applications to Prometheus. For details, see [Monitoring Custom Metrics Using Cloud Native Cluster Monitoring](#).
  - b. Aggregate these custom metrics collected by Prometheus to the API server for the HPA policy to use. For details, see [Creating an HPA Policy Using Custom Metrics](#).
- To provide system resource metrics (such as CPU and memory usage) for workload auto scaling using this add-on, ensure that local data storage is enabled for the add-on and then enable the Metrics API. For details, see [Providing Resource Metrics Through the Metrics API](#). After the configuration, you can use Prometheus to collect system resource metrics. (This operation is not recommended because it may conflict with the Kubernetes Metric Server add-on.)

----End

**Components**

All Kubernetes resources created during Cloud Native Cluster Monitoring add-on installation are created in the namespace named **monitoring**.

**Table 13-24** Add-on components

Component	Description	Supported Deployment Mode	Resource Type
prometheusOperator (workload name: prometheus-operator)	Deploys and manages the Prometheus Server based on CustomResourceDefinitions (CRDs), and monitors and processes the events related to these CRDs. It is the control center of the entire system.	All	Deployment
prometheus (workload name: prometheus-server)	A Prometheus Server cluster deployed by the operator based on the Prometheus CRDs that can be regarded as StatefulSets.	All	StatefulSet
alertmanager (workload name: alertmanager-alertmanager)	Alarm center of the add-on. It receives alarms sent by Prometheus and manages alarm information by deduplicating, grouping, and distributing.	Local data storage enabled	StatefulSet

Component	Description	Supported Deployment Mode	Resource Type
thanosSidecar	Available only in HA mode. Runs with prometheus-server in the same pod to implement persistent storage of Prometheus metric data.	Local data storage enabled	Container
thanosQuery	Available only in HA mode. Entry for PromQL query when Prometheus is in HA scenarios. It can delete duplicate metrics from Store or Prometheus.	Local data storage enabled	Deployment
adapter (workload name: custom-metrics-apiserver)	Aggregates custom metrics to the native Kubernetes API Server.	Local data storage enabled	Deployment
kubeStateMetrics (workload name: kube-state-metrics)	Converts the Prometheus metric data into a format that can be identified by Kubernetes APIs. By default, the kube-state-metrics component does not collect all labels and annotations of Kubernetes resources. To collect all labels and annotations, see <a href="#">Collecting All Labels and Annotations of a Pod</a> . <b>NOTE</b> If the components run in multiple pods, only one pod provides metrics.	All	Deployment
nodeExporter (workload name: node-exporter)	Deployed on each node to collect node monitoring data.	All	DaemonSet
grafana (workload name: grafana)	Visualizes monitoring data. Grafana creates a 5 GiB storage volume by default. Uninstalling the add-on will not delete this volume.	All	Deployment
clusterProblemDetector (workload name: cluster-problem-detector)	Monitors cluster exceptions.	Local data storage enabled	Deployment

## Providing Resource Metrics Through the Metrics API

### NOTE

Resource metrics can only be provided through Metrics API if local data storage is enabled for the Cloud Native Cluster Monitoring add-on.

Resource metrics of containers and nodes, such as CPU and memory usage, can be obtained through the Kubernetes Metrics API. Resource metrics can be directly accessed, for example, by using the **kubectl top** command, or used by HPA or CustomedHPA policies for auto scaling.

The add-on can provide the Kubernetes Metrics API that is disabled by default. To enable the Metrics API, create the following APIService object:

```
apiVersion: apiregistration.k8s.io/v1
kind: APIService
metadata:
  labels:
    app: custom-metrics-apiserver
    release: cceaddon-prometheus
    name: v1beta1.metrics.k8s.io
spec:
  group: metrics.k8s.io
  groupPriorityMinimum: 100
  insecureSkipTLSVerify: true
  service:
    name: custom-metrics-apiserver
    namespace: monitoring
    port: 443
  version: v1beta1
  versionPriority: 100
```

You can save the object as a file, name it **metrics-apiservice.yaml**, and run the following command:

```
kubectl create -f metrics-apiservice.yaml
```

Run the **kubectl top pod -n monitoring** command. If information similar to the following is displayed, the Metrics API can be accessed:

```
# kubectl top pod -n monitoring
NAME                                CPU(cores)  MEMORY(bytes)
.....
custom-metrics-apiserver-d4f556ff9-l2j2m    38m        44Mi
.....
```

### NOTICE

To uninstall the add-on, run the following **kubectl** command and delete the APIService object. Otherwise, residual APIService resources may prevent the installation of the Kubernetes Metrics Server add-on.

```
kubectl delete APIService v1beta1.metrics.k8s.io
```

## Creating an HPA Policy Using Custom Metrics

HPA policies can only be used when Cloud Native Cluster Monitoring is deployed with local data storage enabled. You can configure custom metrics required by HPA policies in the **user-adapter-config** ConfigMap.

**NOTICE**

To use Prometheus to monitor custom metrics, the application needs to provide a metric monitoring API. For details, see [Prometheus Monitoring Data Collection](#).

In this section, the nginx metric (nginx\_connections\_accepted) in [Monitoring Custom Metrics Using Cloud Native Cluster Monitoring](#) 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 **ConfigMaps and Secrets**.
- Step 2** Click the **ConfigMaps** tab, select the **monitoring** namespace, locate the row containing **user-adapter-config** (or **adapter-config**), and click **Update**.

**Figure 13-1** Updating a ConfigMap



- Step 3** In **Data**, click **Edit** for the **config.yaml** file to add a custom metric collection rule under the **rules** field. Click **OK**.

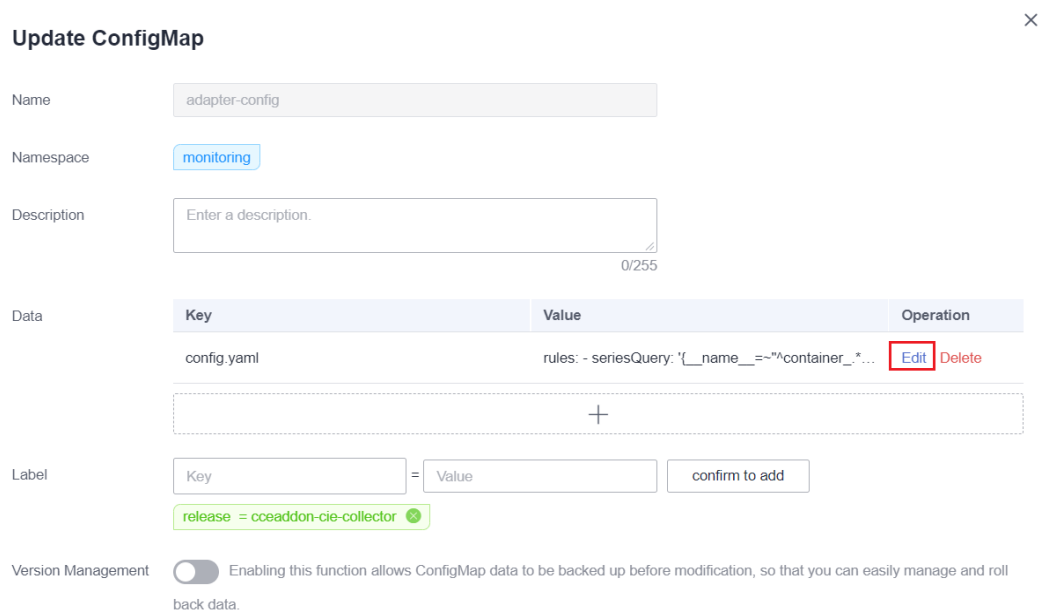
You can add multiple collection rules by adding multiple configurations under the **rules** field. For details, see [Metrics Discovery and Presentation Configuration](#).

Example custom metric rule:

```
rules:
# Match the metric whose name is nginx_connections_accepted. The metric name must be confirmed.
Otherwise, the HPA controller cannot get the metric.
- seriesQuery: '{_name =~ "nginx_connections_accepted",container!="POD",namespace!="",pod!=""}'
  resources:
    # Specify pod and namespace resources.
  overrides:
    namespace:
      resource: namespace
    pod:
      resource: pod
  name:
    #Use nginx_connections_accepted"
  matches: "nginx_connections_accepted"
    #Use nginx_connections_accepted_per_second to represent the metric. The name is the custom metric
name in a custom HPA policy.
  as: "nginx_connections_accepted_per_second"
    # Calculate rate(nginx_connections_accepted[2m]) to specify the number of requests received per second.
  metricsQuery: 'rate(<<.Series>>{<<.LabelMatchers>>,container!="POD"}[2m])'
```

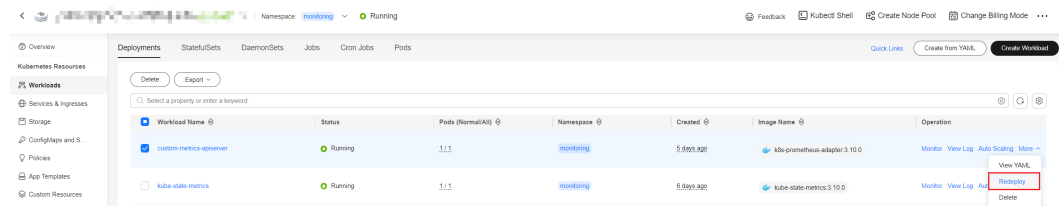


**Figure 13-2** Modifying ConfigMap data



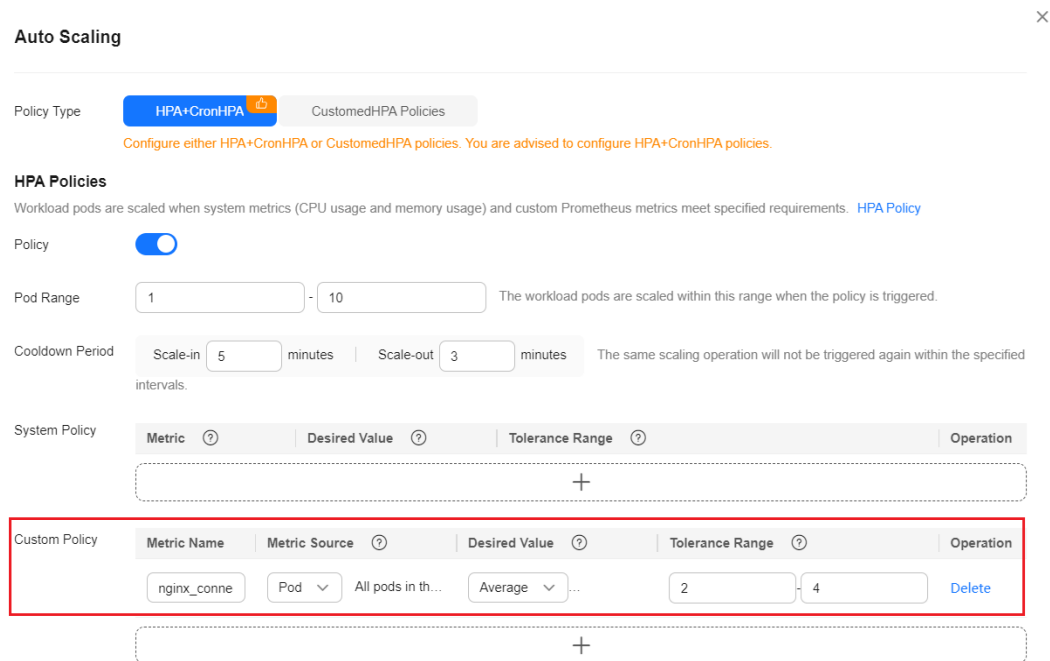
**Step 4** Redeploy the **custom-metrics-apiserver** workload in the **monitoring** namespace.

**Figure 13-3** Redeploying custom-metrics-apiserver



**Step 5** In the navigation pane, choose **Workloads**. Locate the workload for which you want to create an HPA policy and choose **More > Auto Scaling**. In the **Custom Policy** area, you can select the preceding parameters to create an auto scaling policy.

**Figure 13-4** Creating an HPA policy

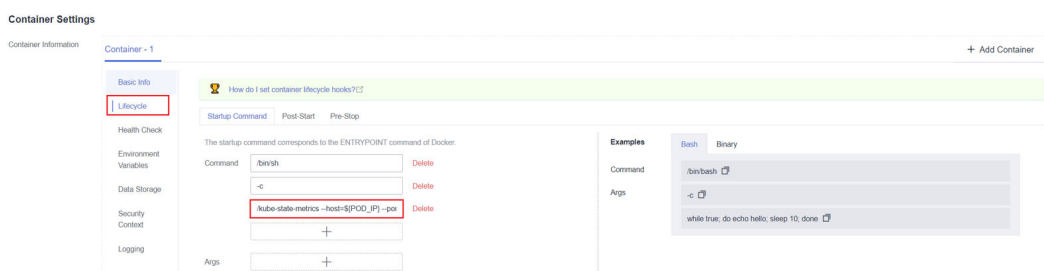


----End

## Collecting All Labels and Annotations of a Pod

- 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** Switch to the **monitoring** namespace, click the **Deployments** tab, and click the name of the **kube-state-metrics** workload. On the page displayed, click the **Containers** tab and click **Edit** on the right.
- Step 3** In the **Lifecycle** area of the container settings, edit the startup command.

**Figure 13-5** Editing the startup command



To collect labels, add the following information to the end of the original **kube-state-metrics** startup parameter:

```
--metric-labels-allowlist=pods=[*],nodes=[node,failure-domain.beta.kubernetes.io/  
zone,topology.kubernetes.io/zone]
```

To collect annotations, add parameters in the startup parameters in the same way.

```
--metric-annotations-allowlist=pods=[*],nodes=[node,failure-domain.beta.kubernetes.io/  
zone,topology.kubernetes.io/zone]
```

**NOTICE**

When editing the startup command, do not modify other original startup parameters. Otherwise, the component may be abnormal.

**Step 4 kube-state-metrics** starts to collect the labels/annotations of pods and nodes and checks whether **kube\_pod\_labels/kube\_pod\_annotations** is in the collection task of CloudScope.

```
kubectcl get servicemonitor kube-state-metrics -nmonitoring -oyaml | grep kube_pod_labels
```

----End

For more kube-state-metrics startup parameters, see [kube-state-metrics/cli-arguments](#).

## Change History

**Table 13-25** Release history

Add-on Version	Supported Cluster Version	New Feature	Community Version
3.11.0	v1.21 v1.23 v1.25 v1.27 v1.28 v1.29 v1.30	CCE clusters 1.30 are supported.	<a href="#">2.37.8</a>
3.7.3	v1.17 v1.19 v1.21 v1.23 v1.25	None	<a href="#">2.35.0</a>
3.7.2	v1.17 v1.19 v1.21 v1.23 v1.25	Supported collection of Virtual Kubelet pod metrics.	<a href="#">2.35.0</a>

Add-on Version	Supported Cluster Version	New Feature	Community Version
3.7.1	v1.17 v1.19 v1.21 v1.23 v1.25	Supports PrometheusAgent.	<a href="#">2.35.0</a>
3.6.6	v1.17 v1.19 v1.21 v1.23 v1.25	<ul style="list-style-type: none"> <li>Grafana is upgraded to 7.5.17.</li> <li>Supported containerd nodes.</li> </ul>	<a href="#">2.35.0</a>
3.5.1	v1.17 v1.19 v1.21 v1.23	None	<a href="#">2.35.0</a>
3.5.0	v1.17 v1.19 v1.21 v1.23	Updated the add-on to its community version v2.35.0.	<a href="#">2.35.0</a>

## 13.3.2 CCE Node Problem Detector

### Introduction

CCE Node Problem Detector (NPD) is an add-on that monitors abnormal events of cluster nodes and connects to a third-party monitoring platform. It is a daemon running on each node. It collects node issues from different daemons and reports them to the API server. This add-on can run as a DaemonSet or a daemon.

For more information, see [node-problem-detector](#).

### Notes and Constraints

- When using this add-on, do not format or partition node disks.
- Each NPD process occupies 30 m CPU and 100 MiB of memory.
- If the NPD version is 1.18.45 or later, the EulerOS version of the host machine must be 2.5 or later.

## Permissions

To monitor kernel logs, the NPD add-on needs to read the host `/dev/kmsg`. Therefore, the privileged mode must be enabled. For details, see [privileged](#).

In addition, CCE mitigates risks according to the least privilege principle. Only the following privileges are available for NPD running:

- `cap_dac_read_search`: permission to access `/run/log/journal`.
- `cap_sys_admin`: permission to access `/dev/kmsg`.

## Installing the Add-on

**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 **CCE Node Problem Detector** on the right, and click **Install**.

**Step 2** On the **Install Add-on** page, configure the specifications as needed.

You can adjust the number of add-on instances and resource quotas as required. High availability is not possible with a single pod. If an error occurs on the node where the add-on instance runs, the add-on will fail.

**Step 3** Configure the add-on parameters.

**Maximum Number of Isolated Nodes in a Fault:** specifies the maximum number of nodes that can be isolated to prevent avalanches in case of a fault occurring on multiple nodes. You can configure this parameter either by percentage or quantity.

**Step 4** Configure deployment policies for the add-on pods.

### NOTE

- Scheduling policies do not take effect on add-on instances of the DaemonSet type.
- When configuring multi-AZ deployment or node affinity, ensure that there are nodes meeting the scheduling policy and that resources are sufficient in the cluster. Otherwise, the add-on cannot run.

**Table 13-26** Configurations for add-on scheduling

Parameter	Description
Multi AZ	<ul style="list-style-type: none"> <li>• <b>Preferred:</b> Deployment pods of the add-on will be preferentially scheduled to nodes in different AZs. If all the nodes in the cluster are deployed in the same AZ, the pods will be scheduled to different nodes in that AZ.</li> <li>• <b>Equivalent mode:</b> Deployment pods of the add-on are evenly scheduled to the nodes in the cluster in each AZ. If a new AZ is added, you are advised to increase add-on pods for cross-AZ HA deployment. With the Equivalent multi-AZ deployment, the difference between the number of add-on pods in different AZs will be less than or equal to 1. If resources in one of the AZs are insufficient, pods cannot be scheduled to that AZ.</li> <li>• <b>Required:</b> Deployment pods of the add-on are forcibly scheduled to nodes in different AZs. There can be at most one pod in each AZ. If nodes in a cluster are not in different AZs, some add-on pods cannot run properly. If a node is faulty, add-on pods on it may fail to be migrated.</li> </ul>
Node Affinity	<ul style="list-style-type: none"> <li>• <b>Not configured:</b> Node affinity is disabled for the add-on.</li> <li>• <b>Node Affinity:</b> Specify the nodes where the add-on is deployed. If you do not specify the nodes, the add-on will be randomly scheduled based on the default cluster scheduling policy.</li> <li>• <b>Specified Node Pool Scheduling:</b> Specify the node pool where the add-on is deployed. If you do not specify the node pool, the add-on will be randomly scheduled based on the default cluster scheduling policy.</li> <li>• <b>Custom Policies:</b> Enter the labels of the nodes where the add-on is to be deployed for more flexible scheduling policies. If you do not specify node labels, the add-on will be randomly scheduled based on the default cluster scheduling policy. If multiple custom affinity policies are configured, ensure that there are nodes that meet all the affinity policies in the cluster. Otherwise, the add-on cannot run.</li> </ul>
Toleration	<p>Using both taints and tolerations allows (not forcibly) the add-on Deployment to be scheduled to a node with the matching taints, and controls the Deployment eviction policies after the node where the Deployment is located is tainted.</p> <p>The add-on adds the default tolerance policy for the <b>node.kubernetes.io/not-ready</b> and <b>node.kubernetes.io/unreachable</b> taints, respectively. The tolerance time window is 60s.</p> <p>For details, see <a href="#">Configuring Tolerance Policies</a>.</p>

**Step 5** Click **Install**.

----End

## Components

**Table 13-27** Add-on components

Component	Description	Resource Type
node-problem-controller	Isolate faults basically based on fault detection results.	Deployment
node-problem-detector	Detect node faults.	DaemonSet

## NPD Check Items

 **NOTE**

Check items are supported only in 1.16.0 and later versions.

Check items cover events and statuses.

- Event-related

For event-related check items, when a problem occurs, NPD reports an event to the API server. The event type can be **Normal** (normal event) or **Warning** (abnormal event).

**Table 13-28** Event-related check items

Check Item	Function	Description
OOMKilling	Listen to the kernel logs and check whether OOM events occur and are reported.  Typical scenario: When the memory usage of a process in a container exceeds the limit, OOM is triggered and the process is terminated.	Warning event Listening object: <b>/dev/kmsg</b>  Matching rule: "Killed process \\d+ (.+) total-vm:\\d+kB, anon-rss:\\d+kB, file-rss:\\d+kB.*"
TaskHung	Listen to the kernel logs and check whether taskHung events occur and are reported.  Typical scenario: Disk I/O suspension causes process suspension.	Warning event Listening object: <b>/dev/kmsg</b>  Matching rule: "task \\S+:\\w+ blocked for more than \\w+ seconds\\."

Check Item	Function	Description
Readonly Filesystem	<p>Check whether the <b>Remount root filesystem read-only</b> error occurs in the system kernel by listening to the kernel logs.</p> <p>Typical scenario: A user detaches a data disk from a node by mistake on the ECS, and applications continuously write data to the mount point of the data disk. As a result, an I/O error occurs in the kernel and the disk is remounted as a read-only disk.</p> <p><b>NOTE</b> If the rootfs of node pods is of the device mapper type, an error will occur in the thin pool if a data disk is detached. This will affect NPD and NPD will not be able to detect node faults.</p>	<p>Warning event</p> <p>Listening object: <b>/dev/kmsg</b></p> <p>Matching rule: <b>Remounting filesystem read-only</b></p>

- Status-related

For status-related check items, when a problem occurs, NPD reports an event to the API server and changes the node status synchronously. This function can be used together with [Node-problem-controller fault isolation](#) to isolate nodes.

**If the check period is not specified in the following check items, the default period is 30 seconds.**

**Table 13-29** Checking system components

Check Item	Function	Description
Container network component error CNIPProblem	Check the status of the CNI components (container network components).	None
Container runtime component error CRIPProblem	Check the status of Docker and containerd of the CRI components (container runtime components).	Check object: Docker or containerd



Check Item	Function	Description
Frequent restarts of Kubelet FrequentKubeletRestart	Periodically backtrack system logs to check whether the key component Kubelet restarts frequently.	<ul style="list-style-type: none"> <li>Default threshold: 10 restarts within 10 minutes</li> <li>If Kubelet restarts for 10 times within 10 minutes, it indicates that the system restarts frequently and a fault alarm is generated.</li> <li>Listening object: logs in the <b>/run/log/journal</b> directory</li> </ul> <p><b>NOTE</b> The Ubuntu and HCE 2.0 OSs do not support the preceding check items due to incompatible log formats.</p>
Frequent restarts of Docker FrequentDockerRestart	Periodically backtrack system logs to check whether the container runtime Docker restarts frequently.	
Frequent restarts of containerd FrequentContainerdRestart	Periodically backtrack system logs to check whether the container runtime containerd restarts frequently.	
kubelet error KubeletProblem	Check the status of the key component Kubelet.	None
kube-proxy error KubeProxyProblem	Check the status of the key component kube-proxy.	None

**Table 13-30** Checking system metrics

Check Item	Function	Description
Conntrack table full ConntrackFullProblem	Check whether the conntrack table is full.	<ul style="list-style-type: none"> <li>Default threshold: 90%</li> <li>Usage: <b>nf_conntrack_count</b></li> <li>Maximum value: <b>nf_conntrack_max</b></li> </ul>
Insufficient disk resources DiskProblem	Check the usage of the system disk and CCE data disks (including the CRI logical disk and kubelet logical disk) on the node.	<ul style="list-style-type: none"> <li>Default threshold: 90%</li> <li>Source: <code>df -h</code></li> </ul> <p>Currently, additional data disks are not supported.</p>

Check Item	Function	Description
Insufficient file handles FDProblem	Check if the FD file handles are used up.	<ul style="list-style-type: none"> <li>• Default threshold: 90%</li> <li>• Usage: the first value in <b>/proc/sys/fs/file-nr</b></li> <li>• Maximum value: the third value in <b>/proc/sys/fs/file-nr</b></li> </ul>
Insufficient node memory MemoryProblem	Check whether memory is used up.	<ul style="list-style-type: none"> <li>• Default threshold: 80%</li> <li>• Usage: <b>MemTotal-MemAvailable</b> in <b>/proc/meminfo</b></li> <li>• Maximum value: <b>MemTotal</b> in <b>/proc/meminfo</b></li> </ul>
Insufficient process resources PIDProblem	Check whether PID process resources are exhausted.	<ul style="list-style-type: none"> <li>• Default threshold: 90%</li> <li>• Usage: <b>nr_threads</b> in <b>/proc/loadavg</b></li> <li>• Maximum value: smaller value between <b>/proc/sys/kernel/pid_max</b> and <b>/proc/sys/kernel/threads-max</b>.</li> </ul>

**Table 13-31** Checking the storage

Check Item	Function	Description
Disk read-only DiskReadOnly	Periodically perform write tests on the system disk and CCE data disks (including the CRI logical disk and Kubelet logical disk) of the node to check the availability of key disks.	<p>Detection paths:</p> <ul style="list-style-type: none"> <li>• /mnt/paas/kubernetes/kubelet/</li> <li>• /var/lib/docker/</li> <li>• /var/lib/containerd/</li> <li>• /var/paas/sys/log/cceaddon-npd/</li> </ul> <p>The temporary file <b>npd-disk-write-ping</b> is generated in the detection path.</p> <p>Currently, additional data disks are not supported.</p>

Check Item	Function	Description
<p>emptyDir storage pool error</p> <p>EmptyDirVolumeGroupStatusError</p>	<p>Check whether the ephemeral volume group on the node is normal.</p> <p>Impact: Pods that depend on the storage pool cannot write data to the temporary volume. The temporary volume is remounted as a read-only file system by the kernel due to an I/O error.</p> <p>Typical scenario: When creating a node, a user configures two data disks as an ephemeral volume storage pool. Some data disks are deleted by mistake. As a result, the storage pool becomes abnormal.</p>	<ul style="list-style-type: none"> <li>● Detection period: 30s</li> <li>● Source: vgs -o vg_name, vg_attr</li> <li>● Principle: Check whether the VG (storage pool) is in the P state. If yes, some PVs (data disks) are lost.</li> <li>● Joint scheduling: The scheduler can automatically identify a PV storage pool error and prevent pods that depend on the storage pool from being scheduled to the node.</li> <li>● Exceptional scenario: The NPD add-on cannot detect the loss of all PVs (data disks), resulting in the loss of VGs (storage pools). In this case, kubelet automatically isolates the node, detects the loss of VGs (storage pools), and updates the corresponding resources in <b>nodestatus.allocatable</b> to <b>0</b>. This prevents pods that depend on the storage pool from being scheduled to the node. The damage of a single PV cannot be detected by this check item, but by the <b>ReadonlyFilesystem</b> check item.</li> </ul>
<p>PV storage pool error</p> <p>LocalPvVolumeGroupStatusError</p>	<p>Check the PV group on the node.</p> <p>Impact: Pods that depend on the storage pool cannot write data to the persistent volume. The persistent volume is remounted as a read-only file system by the kernel due to an I/O error.</p> <p>Typical scenario: When creating a node, a user configures two data disks as a persistent volume storage pool. Some data disks are deleted by mistake.</p>	

Check Item	Function	Description
<p>Mount point error</p> <p>MountPointProblem</p>	<p>Check the mount point on the node.</p> <p>Exceptional definition: You cannot access the mount point by running the <b>cd</b> command.</p> <p>Typical scenario: Network File System (NFS), for example, obsfs and s3fs is mounted to a node. When the connection is abnormal due to network or peer NFS server exceptions, all processes that access the mount point are suspended. For example, during a cluster upgrade, a kubelet is restarted, and all mount points are scanned. If the abnormal mount point is detected, the upgrade fails.</p>	<p>Alternatively, you can run the following command:</p> <pre data-bbox="1086 360 1430 465">for dir in `df -h   grep -v "Mounted on"   awk '{print \\\$NF}'`;do cd \$dir; done &amp;&amp; echo "ok"</pre>

Check Item	Function	Description
<p>Suspended disk I/O DiskHung</p>	<p>Check whether I/O suspension occurs on all disks on the node, that is, whether I/O read and write operations are not responded.</p> <p>Definition of I/O suspension: The system does not respond to disk I/O requests, and some processes are in the D state.</p> <p>Typical scenario: Disks cannot respond due to abnormal OS hard disk drivers or severe faults on the underlying network.</p>	<ul style="list-style-type: none"> <li>• Check object: all data disks</li> <li>• Source: /proc/diskstat Alternatively, you can run the following command: iostat -xmt 1</li> <li>• Thresholds: (All following conditions must be met). <ul style="list-style-type: none"> <li>- Average usage (<b>ioutil</b>) <math>\geq 0.99</math></li> <li>- Average I/O queue length (<b>avgqu-sz</b>) <math>\geq 1</math></li> <li>- Average I/O transfer volume <math>\leq 1</math> Average I/O transfer volume = Number of writes completed per second (<b>iops</b>, unit: w/s) + Amount of data written per second (<b>ioth</b>, unit: wMB/s)</li> </ul> </li> </ul> <p><b>NOTE</b> In some OSs, no data changes during I/O. In this case, calculate the CPU I/O time usage. The value of <b>iowait</b> should be greater than 0.8.</p>

Check Item	Function	Description
Slow disk I/O DiskSlow	<p>Check whether all disks on the node have slow I/Os, that is, whether I/Os respond slowly.</p> <p>Typical scenario: EVS disks have slow I/Os due to network fluctuation.</p>	<ul style="list-style-type: none"> <li>Check object: all data disks</li> <li>Source: /proc/diskstat Alternatively, you can run the following command: iostat -xmt 1</li> <li>Default threshold: Average I/O latency (<b>await</b>) ≥ 5000 ms</li> </ul> <p><b>NOTE</b> If I/O requests are not responded and the <b>await</b> data is not updated, this check item is invalid.</p>

**Table 13-32** Other check items

Check Item	Function	Description
Abnormal NTP NTPProblem	Check whether the node clock synchronization service ntpd or chronyd is running properly and whether a system time drift is caused.	Default clock offset threshold: 8000 ms
Process D error ProcessD	Check whether there is a process D on the node.	Default threshold: 10 abnormal processes detected for three consecutive times Source: <ul style="list-style-type: none"> <li>/proc/{PID}/stat</li> <li>Alternately, you can run the <b>ps aux</b> command.</li> </ul> Exceptional scenario: The ProcessD check item ignores the resident D processes (heartbeat and update) on which the SDI driver on the BMS node depends.
Process Z error ProcessZ	Check whether the node has processes in Z state.	

Check Item	Function	Description
ResolvConf error ResolvConfFileProblem	Check whether the ResolvConf file is lost. Check whether the ResolvConf file is normal. Exceptional definition: No upstream domain name resolution server (nameserver) is included.	Object: <b>/etc/resolv.conf</b>
Existing scheduled event ScheduledEvent	Check whether scheduled live migration events exist on the node. A live migration plan event is usually triggered by a hardware fault and is an automatic fault rectification method at the IaaS layer. Typical scenario: The host is faulty. For example, the fan is damaged or the disk has bad sectors. As a result, live migration is triggered for VMs.	Source: <ul style="list-style-type: none"> <li>• <a href="http://169.254.169.254/metadata/latest/events/scheduled">http://169.254.169.254/metadata/latest/events/scheduled</a></li> </ul> This check item is an Alpha feature and is disabled by default.

The kubelet component has the following default check items, which have bugs or defects. You can fix them by upgrading the cluster or using NPDP.

**Table 13-33** Default kubelet check items

Check Item	Function	Description
Insufficient PID resources PIDPressure	Check whether PIDs are sufficient.	<ul style="list-style-type: none"> <li>• Interval: 10 seconds</li> <li>• Threshold: 90%</li> <li>• Defect: In community version 1.23.1 and earlier versions, this check item becomes invalid when over 65535 PIDs are used. For details, see <a href="#">issue 107107</a>. In community version 1.24 and earlier versions, thread-max is not considered in this check item.</li> </ul>

Check Item	Function	Description
Insufficient memory MemoryPressure	Check whether the allocable memory for the containers is sufficient.	<ul style="list-style-type: none"> <li>Interval: 10 seconds</li> <li>Threshold: max. 100 MiB</li> <li>Allocable = Total memory of a node – Reserved memory of a node</li> <li>Defect: This check item checks only the memory consumed by containers, and does not consider that consumed by other elements on the node.</li> </ul>
Insufficient disk resources DiskPressure	Check the disk usage and inodes usage of the kubelet and Docker disks.	<ul style="list-style-type: none"> <li>Interval: 10 seconds</li> <li>Threshold: 90%</li> </ul>

## Node-problem-controller Fault Isolation

### NOTE

Fault isolation is supported only by add-ons of 1.16.0 and later versions.

By default, if multiple nodes become faulty, NPC adds taints to up to 10% of the nodes. You can set **npc.maxTaintedNode** to increase the threshold.

The open source NPD plugin provides fault detection but not fault isolation. CCE enhances the node-problem-controller (NPC) based on the open source NPD. This component is implemented based on the Kubernetes [node controller](#). For faults reported by NPD, NPC automatically adds taints to nodes for node fault isolation.

**Table 13-34** Parameters

Parameter	Description	Default
npc.enable	Whether to enable NPC This parameter is not supported in 1.18.0 or later versions.	true



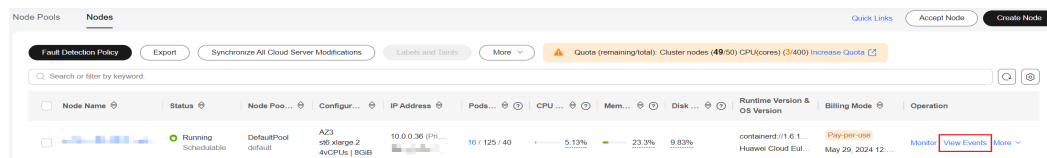
Parameter	Description	Default
npc.maxTaintedNode	The maximum number of nodes that NPC can add taints to when an individual fault occurs on multiple nodes for minimizing impact.  The value can be in int or percentage format.	10%  Value range: <ul style="list-style-type: none"> <li>The value is in int format and ranges from 1 to infinity.</li> <li>The value ranges from 1% to 100%, in percentage. The minimum value of this parameter multiplied by the number of cluster nodes is 1.</li> </ul>
npc.nodeAffinity	Node affinity of the controller	N/A

## Viewing NPD Events

Events reported by the NPD add-on can be queried on the **Nodes** page.

- 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.
- Step 3** Locate the row that contains the target node, and click **View Events**.

**Figure 13-6** Viewing node events



----End

## Collecting Prometheus Metrics

The NPD daemon pod exposes Prometheus metric data on port 19901. By default, the NPD pod is added with the annotation **metrics.alpha.kubernetes.io/custom-endpoints: '[{"api": "prometheus", "path": "/metrics", "port": "19901", "names": ""}]'**. You can build a Prometheus collector to identify and obtain NPD metrics from **http://{{NpdPodIP}}:{{NpdPodPort}}/metrics**.

### NOTE

If the NPD add-on version is earlier than 1.16.5, the exposed port of Prometheus metrics is **20257**.

Currently, the metric data includes **problem\_counter** and **problem\_gauge**, as shown below.

```
# HELP problem_counter Number of times a specific type of problem have occurred.
# TYPE problem_counter counter
problem_counter{reason="DockerHung"} 0
problem_counter{reason="DockerStart"} 0
problem_counter{reason="EmptyDirVolumeGroupStatusError"} 0
...
# HELP problem_gauge Whether a specific type of problem is affecting the node or not.
# TYPE problem_gauge gauge
problem_gauge{reason="CNIsDown",type="CNIPProblem"} 0
problem_gauge{reason="CNIsUp",type="CNIPProblem"} 0
problem_gauge{reason="CRIIsDown",type="CRIPProblem"} 0
problem_gauge{reason="CRIIsUp",type="CRIPProblem"} 0
..
```

## Change History

**Table 13-35** Release history

Add-on Version	Supported Cluster Version	New Feature	Community Version
1.19.1	v1.21 v1.23 v1.25 v1.27 v1.28 v1.29	Fixed some issues.	<a href="#">0.8.10</a>
1.18.46	v1.21 v1.23 v1.25 v1.27 v1.28	CCE clusters 1.28 are supported.	<a href="#">0.8.10</a>
1.18.22	v1.19 v1.21 v1.23 v1.25 v1.27	None	<a href="#">0.8.10</a>
1.18.14	v1.19 v1.21 v1.23 v1.25	<ul style="list-style-type: none"> <li>Supported anti-affinity scheduling of add-on pods on nodes in different AZs.</li> <li>Allows adding a taint to a node before the release of a spot ECS for the node to repel a set of pods.</li> <li>Synchronized time zones used by the add-on and the node.</li> </ul>	<a href="#">0.8.10</a>

Add-on Version	Supported Cluster Version	New Feature	Community Version
1.18.10	v1.19 v1.21 v1.23 v1.25	<ul style="list-style-type: none"> <li>Optimizes the configuration page.</li> <li>Adds threshold configuration to the DiskSlow check item.</li> <li>Added threshold configuration to the NTPProblem check item.</li> <li>Supported anti-affinity scheduling of add-on pods on nodes in different AZs.</li> <li>Supports interruption detection for spot EC2s and evicts pods on nodes before the interruption.</li> </ul>	<a href="#">0.8.10</a>
1.17.4	v1.17 v1.19 v1.21 v1.23 v1.25	Optimizes DiskHung check item.	<a href="#">0.8.10</a>
1.17.3	v1.17 v1.19 v1.21 v1.23 v1.25	<ul style="list-style-type: none"> <li>The maximum number of taint nodes that can be added to the NPC can be configured by percentage.</li> <li>Added the ProcessZ check item.</li> <li>Added the time deviation detection to the NTPProblem check item.</li> <li>Fixed the processes consistently in the D state (exist in the BMS node).</li> </ul>	<a href="#">0.8.10</a>

Add-on Version	Supported Cluster Version	New Feature	Community Version
1.17.2	v1.17 v1.19 v1.21 v1.23 v1.25	<ul style="list-style-type: none"> <li>Added the DiskHung check item for disk I/O.</li> <li>Added the DiskSlow check item for disk I/O.</li> <li>Added the ProcessD check item.</li> <li>Added MountPointProblem to check the health of mount points.</li> <li>To avoid conflicts with the service port range, the default health check listening port is changed to <b>19900</b>, and the default Prometheus metric exposure port is changed to <b>19901</b>.</li> <li>Supports clusters 1.25.</li> </ul>	<b>0.8.10</b>
1.16.4	v1.17 v1.19 v1.21 v1.23	<ul style="list-style-type: none"> <li>Adds the beta check item ScheduledEvent to detect cold and live VM migration events caused by host machine exceptions using the metadata API. This check item is disabled by default.</li> </ul>	<b>0.8.10</b>
1.16.3	v1.17 v1.19 v1.21 v1.23	Adds the function of checking the ResolvConf configuration file.	<b>0.8.10</b>
1.16.1	v1.17 v1.19 v1.21 v1.23	<ul style="list-style-type: none"> <li>Adds node-problem-controller. Supports basic fault isolation.</li> <li>Adds the PID, FD, disk, memory, temporary volume pool, and PV pool check items.</li> </ul>	<b>0.8.10</b>

Add-on Version	Supported Cluster Version	New Feature	Community Version
1.15.0	v1.17 v1.19 v1.21 v1.23	<ul style="list-style-type: none"> <li>• Hardens check items comprehensively to avoid false positives.</li> <li>• Supports kernel check. Supports reporting of OOMKilled and TaskHung events.</li> </ul>	<a href="#">0.8.10</a>
1.14.11	v1.17 v1.19 v1.21	CCE clusters 1.21 are supported.	<a href="#">0.7.1</a>
1.14.5	v1.17 v1.19	Fixes the issue that monitoring metrics cannot be obtained.	<a href="#">0.7.1</a>
1.14.4	v1.17 v1.19	<ul style="list-style-type: none"> <li>• Supported containerd nodes.</li> </ul>	<a href="#">0.7.1</a>
1.14.2	v1.17 v1.19	<ul style="list-style-type: none"> <li>• CCE clusters 1.19 are supported.</li> <li>• Supported Ubuntu OS and Kata containers.</li> </ul>	<a href="#">0.7.1</a>
1.13.8	v1.15.11 v1.17	<ul style="list-style-type: none"> <li>• Fixes the CNI health check issue on the container tunnel network.</li> <li>• Adjusts resource quotas.</li> </ul>	<a href="#">0.7.1</a>
1.13.6	v1.15.11 v1.17	Fixes the issue that zombie processes are not reclaimed.	<a href="#">0.7.1</a>
1.13.5	v1.15.11 v1.17	Added taint tolerance configuration.	<a href="#">0.7.1</a>
1.13.2	v1.15.11 v1.17	Added resource limits and enhanced the detection capability of the cni add-on.	<a href="#">0.7.1</a>

### 13.3.3 CCE Network Metrics Exporter

#### Introduction

CCE Network Metrics Exporter enhances the observability of container networks by providing monitoring data related to container networks in CCE Turbo clusters, so you can observe network traffic and detect and locate network problems faster.

The add-on pods can only be deployed on HCE 2.0 nodes running x86/Arm, or EulerOS nodes running x86.

The IP and TCP traffic can be monitored by pod or flow. You can use podSelector to select the monitoring backend. Multiple monitoring tasks can be created and monitoring metrics can be selected as required. The label information of pods can also be obtained. The monitoring information has been adapted to the Prometheus format. You can call the Prometheus API to view monitoring data.

## Notes and Constraints

- This add-on can be installed only in CCE Turbo clusters 1.19 or later. The add-on pods can run only on nodes running EulerOS on x86.
- This add-on can be installed on nodes that use the containerd or Docker container engine. In containerd nodes, it can trace pod updates in real time. In Docker nodes, it can query pod updates in polling mode.
- Only traffic statistics of secure containers using the Kata container runtime and common containers (using the runC container runtime) in a CCE Turbo cluster can be collected.
- By default, traffic monitoring is not carried out after the installation of the add-on. To set up a traffic monitoring task, you must create a MonitorPolicy in the kube-system namespace.
- Pods using the host network mode cannot be monitored.
- Ensure that there are enough resources on a node for installing the add-on.
- The source of monitoring labels and user labels must be already available before a pod is created.
- You can specify a maximum of five labels (a maximum of 10 labels in versions later than 1.3.4). You cannot specify the labels used by the system. Labels used by the system include **pod**, **task**, **ipfamily**, **srcip**, **dstip**, **srcport**, **dstport**, and **protocol**.

## Installing the Add-on

**Step 1** Log in to the CCE console and click the CCE Turbo cluster name to access the cluster. In the navigation pane, choose **Add-ons**, locate **CCE Network Metrics Exporter** on the right, and click **Install**.

**Step 2** On the Install Add-on page, check the add-on configuration.

No parameter can be configured for the current add-on.

**Step 3** Click **Install**.

After the add-on is installed, select the cluster and choose **Add-ons** in the navigation pane. On the displayed page, view the add-on in the **Add-ons Installed** area.

----End

## Components

**Table 13-36** Add-on components

Component	Description	Resource Type
dolphin	Used to monitor the container network traffic of CCE Turbo clusters	Daemon Set

## Monitoring Metrics of dolphin

You can deliver a monitoring task by creating a MonitorPolicy. A MonitorPolicy can be created by calling an API or using the **kubectl apply** command after logging in to a worker node. A MonitorPolicy represents a monitoring task and provides optional parameters such as **selector** and **podLabel**. The following table describes the supported monitoring metrics.

**Table 13-37** Supported monitoring metrics

Monitoring Metric	Monitoring Item	Granularity	Supported Runtime	Supported Cluster Version	Supported Add-on Version	Supported OS
Number of IPv4 packets sent to the Internet	dolphin_ip4_send_pkt_internet	Pod	runC/ Kata	v1.19 or later	1.1.2	EulerOS 2.9 on x86 EulerOS 2.10 on x86
Number of IPv4 bytes sent to the Internet	dolphin_ip4_send_byte_internet	Pod	runC/ Kata	v1.19 or later	1.1.2	EulerOS 2.9 on x86 EulerOS 2.10 on x86
Number of received IPv4 packets	dolphin_ip4_rcv_pkt	Pod	runC/ Kata	v1.19 or later	1.1.2	EulerOS 2.9 on x86 EulerOS 2.10 on x86

Monitoring Metric	Monitoring Item	Granularity	Supported Runtime	Supported Cluster Version	Supported Add-on Version	Supported OS
Number of received IPv4 bytes	dolphin_ip4_rcv_byte	Pod	runC/ Kata	v1.19 or later	1.1.2	EulerOS 2.9 on x86 EulerOS 2.10 on x86
Number of sent IPv4 packets	dolphin_ip4_send_pkt	Pod	runc/ kata	v1.19 or later	1.1.2	EulerOS 2.9 on x86 EulerOS 2.10 on x86
Number of sent IPv4 bytes	dolphin_ip4_send_byte	Pod	runC/ Kata	v1.19 or later	1.1.2	EulerOS 2.9 on x86 EulerOS 2.10 on x86
Health status of the latest health check	dolphin_health_check_statuses	Pod	runc/ kata	v1.19 or later	1.2.2	EulerOS 2.9 on x86 EulerOS 2.10 on x86
Total number of successful health checks	dolphin_health_check_successful_counter	Pod	runC/ Kata	v1.19 or later	1.2.2	EulerOS 2.9 on x86 EulerOS 2.10 on x86
Total number of failed health checks	dolphin_health_check_failed_counter	Pod	runC/ Kata	v1.19 or later	1.2.2	EulerOS 2.9 on x86 EulerOS 2.10 on x86



Monitoring Metric	Monitoring Item	Granularity	Supported Runtime	Supported Cluster Version	Supported Add-on Version	Supported OS
Number of received IP packets	dolphin_ip_receive_pkt	Pod	runC	v1.23 or later	1.3.5	EulerOS 2.9 on x86 EulerOS 2.10 on x86
Number of received IP bytes	dolphin_ip_receive_byte	Pod	runC	v1.23 or later	1.3.5	EulerOS 2.9 on x86 EulerOS 2.10 on x86
Number of sent IP packets	dolphin_ip_send_pkt	Pod	runC	v1.23 or later	1.3.5	EulerOS 2.9 on x86 EulerOS 2.10 on x86
Number of sent IP bytes	dolphin_ip_send_byte	Pod	runC	v1.23 or later	1.3.5	EulerOS 2.9 on x86 EulerOS 2.10 on x86
Number of received TCP packets	dolphin_tcp_receive_pkt	Pod	runC	v1.23 or later	1.3.5	EulerOS 2.9 on x86 EulerOS 2.10 on x86
Number of received TCP bytes	dolphin_tcp_receive_byte	Pod	runC	v1.23 or later	1.3.5	EulerOS 2.9 on x86 EulerOS 2.10 on x86

Monitoring Metric	Monitoring Item	Granularity	Supported Runtime	Supported Cluster Version	Supported Add-on Version	Supported OS
Number of sent TCP packets	dolphin_tcp_send_pkt	Pod	runC	v1.23 or later	1.3.5	EulerOS 2.9 on x86 EulerOS 2.10 on x86
Number of sent TCP bytes	dolphin_tcp_send_byte	Pod	runC	v1.23 or later	1.3.5	EulerOS 2.9 on x86 EulerOS 2.10 on x86
Number of retransmitted TCP packets	dolphin_tcp_retrans	Pod	runC	v1.23 or later	1.3.5	EulerOS 2.9 on x86 EulerOS 2.10 on x86
Number of new TCP connections	dolphin_tcp_connections	Pod	runC	v1.23 or later	1.3.5	EulerOS 2.9 on x86 EulerOS 2.10 on x86
Number of received IP packets	dolphin_flow_ip_receive_pkt	Flow	runC	v1.23 or later	1.3.5	EulerOS 2.9 on x86 EulerOS 2.10 on x86
Number of received IP bytes	dolphin_flow_ip_receive_byte	Flow	runC	v1.23 or later	1.3.5	EulerOS 2.9 on x86 EulerOS 2.10 on x86

Monitoring Metric	Monitoring Item	Granularity	Supported Runtime	Supported Cluster Version	Supported Add-on Version	Supported OS
Number of sent IP packets	dolphin_flow_ip_send_pkt	Flow	runC	v1.23 or later	1.3.5	EulerOS 2.9 on x86 EulerOS 2.10 on x86
Number of sent IP bytes	dolphin_flow_ip_send_byte	Flow	runC	v1.23 or later	1.3.5	EulerOS 2.9 on x86 EulerOS 2.10 on x86
Number of received TCP packets	dolphin_flow_tcp_receive_pkt	Flow	runC	v1.23 or later	1.3.5	EulerOS 2.9 on x86 EulerOS 2.10 on x86
Number of received TCP bytes	dolphin_flow_tcp_receive_byte	Flow	runC	v1.23 or later	1.3.5	EulerOS 2.9 on x86 EulerOS 2.10 on x86
Number of sent TCP packets	dolphin_flow_tcp_send_pkt	Flow	runC	v1.23 or later	1.3.5	EulerOS 2.9 on x86 EulerOS 2.10 on x86
Number of sent TCP bytes	dolphin_flow_tcp_send_byte	Flow	runC	v1.23 or later	1.3.5	EulerOS 2.9 on x86 EulerOS 2.10 on x86

Monitoring Metric	Monitoring Item	Granularity	Supported Runtime	Supported Cluster Version	Supported Add-on Version	Supported OS
Number of retransmitted TCP packets	dolphin_flow_tcp_retrans	Flow	runC	v1.23 or later	1.3.5	EulerOS 2.9 on x86 EulerOS 2.10 on x86
TCP smoothed round trip	dolphin_flow_tcp_srtt	Flow	runC	v1.23 or later	1.3.5	EulerOS 2.9 on x86 EulerOS 2.10 on x86

## Delivering a Monitoring Task

The template for creating a MonitorPolicy is as follows:

```

apiVersion: crd.dolphin.io/v1
kind: MonitorPolicy
metadata:
  name: example-task          # Monitoring task name.
  namespace: kube-system     # The value must be kube-system. This field is mandatory.
spec:
  selector:                   # (Optional) Backend monitored by the dolphin add-on, for example,
  labelSelector: By default, all containers on the node are monitored.
  matchLabels:
    app: nginx
  matchExpressions:
    - key: app
      operator: In
      values:
        - nginx
  podLabel: [app]            # (Optional) Pod label.
  ip4Tx:                     # (Optional) Indicates whether to collect statistics about the number of sent IPv4
  packets and the number of sent IPv4 bytes. This function is disabled by default.
    enable: true
  ip4Rx:                     # (Optional) Indicates whether to collect statistics about the number of received
  IPv4 packets and the number of received IPv4 bytes. This function is disabled by default.
    enable: true
  ip4TxInternet:             # (Optional) Indicates whether to collect statistics about the number of sent
  IPv4 packets and the number of sent IPv4 bytes. This function is disabled by default.
    enable: true
  healthCheck:               # (Optional) Whether to collect statistics about whether the latest health check
  result is healthy and the total number of healthy times and unhealthy times in the pod health checks of the
  local node. This function is disabled by default.
    enable: true              # true false
    failureThreshold: 3      # (Optional) Number of failures that determine the health check is unhealthy.
  One check failure is considered as unhealthy by default.
    periodSeconds: 5         # (Optional) Interval between health checks, in seconds. The default value is 60.
    command: ""              # (Optional) Health check command. The value can be ping (default), arping,
  or curl.
  ipFamilies: ["" ]         # (Optional) Health check IP address family. The value is IPv4 by default.
  port: 80                   # (Optional) Port number, which is mandatory when curl is used.
  path: ""                   # (Optional) HTTP API path, which is mandatory when curl is used.

```

```

monitor:
  ip:
    ipReceive:
      aggregateType: flow # (Optional). The value can be pod (monitored by pod) or flow (monitored
by flow).
    ipSend:
      aggregateType: flow # (Optional). The value can be pod (monitored by pod) or flow (monitored
by flow).
  tcp:
    tcpReceive:
      aggregateType: flow # (Optional). The value can be pod (monitored by pod) or flow (monitored
by flow).
    tcpSend:
      aggregateType: flow # (Optional). The value can be pod (monitored by pod) or flow (monitored
by flow).
    tcpRetrans:
      aggregateType: flow # (Optional). The value can be pod (monitored by pod) or flow (monitored
by flow).
    tcpRtt:
      aggregateType: flow # (Optional). The value can be flow (monitored by flow). The unit is  $\mu$ s.
    tcpNewConnection:
      aggregateType: pod # (Optional). The value can be pod (monitored by pod).

```

**PodLabel:** You can enter the labels of multiple pods and separate them with commas (,), for example, [app, version].

Labels must comply with the following rules. The corresponding regular expression is  $(^[a-zA-Z_])|(^([a-zA-Z][a-zA-Z0-9_][a-zA-Z0-9])([a-zA-Z0-9_]){0,254})$$ .

- A maximum of five labels can be entered (a maximum of 10 labels in versions later than 1.3.4). A label can contain a maximum of 256 characters.
- The value cannot start with a digit or double underscores (``\_``).
- The format of a single label must comply with A-Za-z\_0-9.

You can create, modify, and delete monitoring tasks in the preceding format. A maximum of 10 monitoring tasks can be created. When multiple monitoring tasks match the same monitoring backend, each monitoring backend generates the monitoring metrics specific to the number of monitoring tasks.

#### NOTE

- If you modify or delete a monitoring task, monitoring data collected by the monitoring task will be lost. Therefore, exercise caution when performing this operation.
- If the add-on is uninstalled, the MonitorPolicy of the monitoring task will be removed together with the add-on.

Example application scenarios:

1. The example below monitors all pods with label **app=nginx** selected by the **labelselector** on a node and generates the three health check metrics. By default, the **ping** command is used to detect local pods. If the monitored container contains the **test** and **app** labels, the key-value information of the corresponding label is carried in the monitoring metrics. Otherwise, the value of the corresponding label is **not found**.

```

apiVersion: crd.dolphin.io/v1
kind: MonitorPolicy
metadata:
  name: example-task
  namespace: kube-system
spec:
  selector:
    matchLabels:
      app: nginx

```

```
podLabel: [test, app]
healthCheck:
  enable: true
  failureThreshold: 3
  periodSeconds: 5
```

2. The example below monitors all pods with label **app=nginx** selected by the **labelselector** on a node and generates the three health check metrics. Customized **curl** command is used, which considers only the network connectivity. That is, no matter what the HTTP code is returned by the program, the pod is considered healthy as long as the network is connected. If the monitored container contains the **test** and **app** labels, the key-value information of the corresponding label is carried in the monitoring metrics. Otherwise, the value of the corresponding label is **not found**.

```
apiVersion: crd.dolphin.io/v1
kind: MonitorPolicy
metadata:
  name: example-task
  namespace: kube-system
spec:
  selector:
    matchLabels:
      app: nginx
  podLabel: [test, app]
  healthCheck:
    enable: true
    failureThreshold: 3
    periodSeconds: 5
    command: "curl"
    port: 80
    path: "healthz"
```

3. The example below monitors all pods with label **app=nginx** selected by the **labelselector** on a node and generates monitoring data by pod, including the number of sent IP packets, received IP packets, sent IP bytes, received IP bytes, sent TCP packets, received TCP packets, sent TCP bytes, received TCP bytes, retransmitted TCP packets, and new TCP connections. If the monitored container contains the **test** and **app** labels, the key-value information of the corresponding label is carried in the monitoring metrics. Otherwise, the value of the corresponding label is **not found**.

```
apiVersion: crd.dolphin.io/v1
kind: MonitorPolicy
metadata:
  name: example-task
  namespace: kube-system
spec:
  selector:
    matchLabels:
      app: nginx
  podLabel: [test, app]
  monitor:
    ip:
      ipReceive:
        aggregateType: pod
      ipSend:
        aggregateType: pod
    tcp:
      tcpReceive:
        aggregateType: pod
      tcpSend:
        aggregateType: pod
      tcpRetrans:
        aggregateType: pod
      tcpNewConnection:
        aggregateType: pod
```

- The example below monitors all pods with label **app=nginx** selected by the **labelselector** on a node and generates monitoring data by flow, including the number of sent IP packets, received IP packets, sent IP bytes, received IP bytes, sent TCP packets, received TCP packets, sent TCP bytes, received TCP bytes, retransmitted TCP packets, and TCP round-trip time ( $\mu$ s). If the monitored container contains the **test** and **app** labels, the key-value information of the corresponding label is carried in the monitoring metrics. Otherwise, the value of the corresponding label is **not found**. Flow-based monitoring helps you learn about detailed container traffic information. It generates a large amount of data that occupies more CPU and memory resources. Use flow-based monitoring based on your needs.

A flow-based IP monitoring task (one or more IP monitoring items enabled in a MonitorPolicy) occupies 2.6 MiB of kernel memory. A flow-based TCP monitoring task (one or more TCP monitoring items enabled in a MonitorPolicy) occupies 14 MiB of kernel memory.

```
apiVersion: crd.dolphin.io/v1
kind: MonitorPolicy
metadata:
  name: example-task
  namespace: kube-system
spec:
  selector:
    matchLabels:
      app: nginx
  podLabel: [test, app]
  monitor:
    ip:
      ipReceive:
        aggregateType: flow
      ipSend:
        aggregateType: flow
    tcp:
      tcpReceive:
        aggregateType: flow
      tcpSend:
        aggregateType: flow
      tcpRetrans:
        aggregateType: flow
      tcpRtt:
        aggregateType: flow
```

#### NOTE

If the data generated by flow-based monitoring exceeds a certain limit, excess flow statistics will be lost. The restrictions are as follows:

- A maximum of 50,000 TCP flows (per monitoring task) can be collected in kernel mode within 10 seconds.
  - A maximum of 10,000 IP flows (per monitoring task) can be collected in kernel mode within 10 seconds.
  - A maximum of 60,000 flow statistical records (all monitoring tasks) can be cached at the interval between two CloudScope data fetches.
  - If CloudScope does not obtain monitoring data for a long time, only the monitoring data generated within the latest hour will be cached.
- The example below monitors all pods on a node and generates the number of sent IPv4 packets and the number of sent IPv4 bytes. If the monitored container contains the **app** label, the key-value information of the corresponding label is carried in the monitoring metrics. Otherwise, the value of the corresponding label is **not found**.

```
apiVersion: crd.dolphin.io/v1
kind: MonitorPolicy
```

```

metadata:
  name: example-task
  namespace: kube-system
spec:
  podLabel: [app]
  ip4Tx:
    enable: true

```

6. The example below monitors all pods with label **app=nginx** selected by the **labelselector** on a node and generates the number of sent IPv4 packets, received IPv4 packets, sent IPv4 bytes, received IPv4 bytes, IPv4 packets sent to the public network, and IPv4 bytes sent to the public network. If the monitored container contains the **test** and **app** labels, the key-value information of the corresponding label is carried in the monitoring metrics. Otherwise, the value of the corresponding label is **not found**.

```

apiVersion: crd.dolphin.io/v1
kind: MonitorPolicy
metadata:
  name: example-task
  namespace: kube-system
spec:
  selector:
    matchLabels:
      app: nginx
    podLabel: [test, app]
  ip4Tx:
    enable: true
  ip4Rx:
    enable: true
  ip4TxInternet:
    enable: true

```

## Checking Traffic Statistics

The monitoring data collected by this add-on is exported in Prometheus exporter format, which can be obtained in the following ways:

- Directly access service port 10001 provided by the dolphin add-on, for example, `http://{POD_IP}:10001/metrics`.

Note that if you access the dolphin service port on a node, allow access from the security group of the node and pod.

Examples of the monitored information:

- Example 1 (number of IPv4 packets sent to the Internet):  
`dolphin_ip4_send_pkt_internet{app="nginx",pod="default/nginx-66c9c65dbf-zjg24",task="kube-system/example-task "} 241`

In the preceding example, the namespace of the pod is **default**, the pod name is **nginx-66c9c65dbf-zjg24**, the label is **app**, and the value is **nginx**. This metric is created by monitoring task **example-task**, and the number of IPv4 packets sent by the pod to the public network is **241**.

- Example 2 (number of IPv4 bytes sent to the Internet):  
`dolphin_ip4_send_byte_internet{app="nginx",pod="default/nginx-66c9c65dbf-zjg24",task="kube-system/example-task "} 23618`

In the preceding example, the namespace of the pod is **default**, the pod name is **nginx-66c9c65dbf-zjg24**, the label is **app**, and the value is **nginx**. This metric is created by monitoring task **example-task**, and the number of IPv4 bytes sent by the pod to the public network is **23618**.

- Example 3 (number of sent IPv4 packets):  
`dolphin_ip4_send_pkt{app="nginx",pod="default/nginx-66c9c65dbf-zjg24",task="kube-system/example-task "} 379`



In the preceding example, the namespace of the pod is **default**, the pod name is **nginx-66c9c65dbf-zjg24**, the label is **app**, and the value is **nginx**. This metric is created by monitoring task **example-task**, and the number of IPv4 packets sent by the pod is **379**.

- Example 4 (number of sent IPv4 bytes):

```
dolphin_ip4_send_byte{app="nginx",pod="default/nginx-66c9c65dbf-zjg24",task="kube-system/example-task "} 33129
```

In the preceding example, the namespace of the pod is **default**, the pod name is **nginx-66c9c65dbf-zjg24**, the label is **app**, and the value is **nginx**. This metric is created by monitoring task **example-task**, and the number of IPv4 bytes sent by the pod is **33129**.

- Example 5 (number of received IPv4 packets):

```
dolphin_ip4_rcv_pkt{app="nginx",pod="default/nginx-66c9c65dbf-zjg24",task="kube-system/example-task "} 464
```

In the preceding example, the namespace of the pod is **default**, the pod name is **nginx-66c9c65dbf-zjg24**, the label is **app**, and the value is **nginx**. This metric is created by monitoring task **example-task**, and the number of IPv4 packets received by the pod is **464**.

- Example 6 (number of received IPv4 bytes):

```
dolphin_ip4_rcv_byte{app="nginx",pod="default/nginx-66c9c65dbf-zjg24",task="kube-system/example-task "} 34654
```

In the preceding example, the namespace of the pod is **default**, the pod name is **nginx-66c9c65dbf-zjg24**, the label is **app**, and the value is **nginx**. This metric is created by monitoring task **example-task**, and the number of IPv4 bytes received by the pod is **34654**.

- Example 7 (health check status)

```
dolphin_health_check_status{app="nginx",pod="default/nginx-b74766f5f-7582p",task="kube-system/example-task"} 0
```

In the preceding example, the namespace of the pod is **kube-system**, the pod name is **default/nginx-deployment-b74766f5f-7582p**, the label is **app**, and the value is **nginx**. This metric is created by monitoring task **example-task**, and the network health status of the pod is **0** (healthy). If the network status is unhealthy, the value will be **1**.

- Example 8 (number of successful health checks)

```
dolphin_health_check_successful_counter{app="nginx",pod="default/nginx-b74766f5f-7582p",task="kube-system/example-task"} 5
```

In the preceding example, the namespace of the pod is **kube-system**, the pod name is **default/nginx-deployment-b74766f5f-7582p**, the label is **app**, and the value is **nginx**. This metric is created by monitoring task **example-task**, and the number of successful network health checks for the pod is **5**.

- Example 9 (number of failed health check failures)

```
dolphin_health_check_failed_counter{app="nginx",pod="default/nginx-b74766f5f-7582p",task="kube-system/example-task"} 0
```

In the preceding example, the namespace of the pod is **kube-system**, the pod name is **default/nginx-deployment-b74766f5f-7582p**, the label is **app**, and the value is **nginx**. This metric is created by monitoring task **example-task**, and the number of failed network health checks for the pod is **0**.

- Example 10 (flow-based monitoring result):

```
dolphin_flow_tcp_send_byte{app="nginx",dstip="192.168.0.89",dstport="80",ipfamily="ipv4",pod="kube-system/nginx-b74766f5f-7582p",srcip="192.168.1.67",srcport="12973",task="kube-system/example-task"} 1725 1700538280914
```

In the preceding example, the namespace of the pod is **kube-system**, the pod name is **nginx-b74766f5f-7582p**, the label is **app**, and the value is **nginx**.

This metric is created by monitoring task **example-task**, and the number of IPv4 TCP bytes sent from **192.168.1.67:12973** to **192.168.0.89:80** is **1725**. The timestamp is **1700538280914**.

- Example 11 (pod-based monitoring result):  

```
dolphin_tcp_send_pkt{app="nginx",ipfamily="ipv4",pod="kube-system/nginx-b74766f5f-7582p",task="kube-system/example-task"} 14
dolphin_tcp_send_pkt{app="nginx",ipfamily="ipv6",pod="kube-system/nginx-b74766f5f-7582p",task="kube-system/example-task"} 0
```

In the preceding example, the namespace of the pod is **kube-system**, the pod name is **nginx-b74766f5f-7582p**, the label is **app**, and the value is **nginx**. This metric is created by monitoring task **example-task**, and the number of IPv4 packets sent by the pod is **14**. **0** IPv6 packets were sent by this pod.

 **NOTE**

If the container does not contain the specified label, the label value in the response body is **not found**. The format is as follows:

```
dolphin_ip4_send_byte_internet{test="not found", pod="default/nginx-66c9c65dbf-zjg24",task="default" } 23618
```

## Change History

**Table 13-38** Release history

Add-on Version	Supported Cluster Version	New Feature
1.4.7	v1.23 v1.25 v1.27 v1.28 v1.29	Fixed some issues.
1.4.5	v1.23 v1.25 v1.27 v1.28 v1.29	<ul style="list-style-type: none"> <li>• Supported pod-based UDP, TCP drop, and TCP connect fail monitoring for common containers.</li> <li>• Supported flow-based UDP and TCP drop monitoring for common containers.</li> <li>• Supported Huawei Cloud EulerOS 2.0 on x86 or Arm.</li> <li>• CCE clusters 1.29 are supported.</li> </ul>

Add-on Version	Supported Cluster Version	New Feature
1.3.8	v1.23 v1.25 v1.27 v1.28	<ul style="list-style-type: none"> <li>• Supported pod-based IP address and TCP monitoring of containers.</li> <li>• Supported flow-based IP address and TCP monitoring of containers.</li> <li>• CCE clusters 1.27 are supported.</li> <li>• CCE clusters 1.28 are supported.</li> </ul>
1.2.27	v1.19 v1.21 v1.23 v1.25	None
1.2.4	v1.19 v1.21 v1.23 v1.25	<ul style="list-style-type: none"> <li>• Added the description that only EulerOS is supported.</li> </ul>
1.2.2	v1.19 v1.21 v1.23 v1.25	<ul style="list-style-type: none"> <li>• Supported VPC network health checks in a local pod.</li> </ul>
1.1.8	v1.19 v1.21 v1.23 v1.25	<ul style="list-style-type: none"> <li>• CCE clusters 1.25 are supported.</li> </ul>
1.1.6	v1.19 v1.21 v1.23	None
1.1.5	v1.19 v1.21 v1.23	<ul style="list-style-type: none"> <li>• Optimizes liveness health check.</li> </ul>
1.1.2	v1.19 v1.21 v1.23	<ul style="list-style-type: none"> <li>• Supports wide matching of operating system types.</li> </ul>

Add-on Version	Supported Cluster Version	New Feature
1.0.1	v1.19 v1.21	<ul style="list-style-type: none"><li>Supports traffic statistics persistence and local socket communications.</li></ul>

### 13.3.4 Kubernetes Metrics Server

From version 1.8 onwards, Kubernetes provides resource usage metrics, such as the container CPU and memory usage, through the Metrics API. These metrics can be directly accessed by users (for example, by using the **kubectl top** command) or used by controllers (for example, Horizontal Pod Autoscaler) in a cluster for decision-making. The specific component is metrics-server, which is used to substitute for heapster for providing the similar functions. heapster has been gradually abandoned since v1.11.

metrics-server is an aggregator for monitoring data of core cluster resources. You can quickly install this add-on on the CCE console.

After installing this add-on, you can create HPA policies. For details, see [Creating an HPA Policy](#).

The official community project and documentation are available at <https://github.com/kubernetes-sigs/metrics-server>.

#### Installing the Add-on

**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 **Kubernetes Metrics Server** on the right, and click **Install**.

**Step 2** On the **Install Add-on** page, configure the specifications as needed.

- If you selected **Preset**, you can choose between **Small** or **Large** as needed. The system will automatically set the number of add-on pods and resource quotas according to the preset specifications. You can see the configurations on the console.

The **smaller specification** lacks HA capabilities, while the **large specification** has them. However, deploying multiple pods requires more compute resources.

- If you selected **Custom**, you can adjust the number of pods and resource quotas as needed. High availability is not possible with a single pod. If an error occurs on the node where the add-on instance runs, the add-on will fail.

**Step 3** Configure deployment policies for the add-on pods.

#### NOTE

- Scheduling policies do not take effect on add-on instances of the DaemonSet type.
- When configuring multi-AZ deployment or node affinity, ensure that there are nodes meeting the scheduling policy and that resources are sufficient in the cluster. Otherwise, the add-on cannot run.

**Table 13-39** Configurations for add-on scheduling

Parameter	Description
Multi AZ	<ul style="list-style-type: none"> <li>● <b>Preferred:</b> Deployment pods of the add-on will be preferentially scheduled to nodes in different AZs. If all the nodes in the cluster are deployed in the same AZ, the pods will be scheduled to different nodes in that AZ.</li> <li>● <b>Equivalent mode:</b> Deployment pods of the add-on are evenly scheduled to the nodes in the cluster in each AZ. If a new AZ is added, you are advised to increase add-on pods for cross-AZ HA deployment. With the Equivalent multi-AZ deployment, the difference between the number of add-on pods in different AZs will be less than or equal to 1. If resources in one of the AZs are insufficient, pods cannot be scheduled to that AZ.</li> <li>● <b>Required:</b> Deployment pods of the add-on are forcibly scheduled to nodes in different AZs. There can be at most one pod in each AZ. If nodes in a cluster are not in different AZs, some add-on pods cannot run properly. If a node is faulty, add-on pods on it may fail to be migrated.</li> </ul>
Node Affinity	<ul style="list-style-type: none"> <li>● <b>Not configured:</b> Node affinity is disabled for the add-on.</li> <li>● <b>Node Affinity:</b> Specify the nodes where the add-on is deployed. If you do not specify the nodes, the add-on will be randomly scheduled based on the default cluster scheduling policy.</li> <li>● <b>Specified Node Pool Scheduling:</b> Specify the node pool where the add-on is deployed. If you do not specify the node pool, the add-on will be randomly scheduled based on the default cluster scheduling policy.</li> <li>● <b>Custom Policies:</b> Enter the labels of the nodes where the add-on is to be deployed for more flexible scheduling policies. If you do not specify node labels, the add-on will be randomly scheduled based on the default cluster scheduling policy. If multiple custom affinity policies are configured, ensure that there are nodes that meet all the affinity policies in the cluster. Otherwise, the add-on cannot run.</li> </ul>
Toleration	<p>Using both taints and tolerations allows (not forcibly) the add-on Deployment to be scheduled to a node with the matching taints, and controls the Deployment eviction policies after the node where the Deployment is located is tainted.</p> <p>The add-on adds the default tolerance policy for the <b>node.kubernetes.io/not-ready</b> and <b>node.kubernetes.io/unreachable</b> taints, respectively. The tolerance time window is 60s.</p> <p>For details, see <a href="#">Configuring Tolerance Policies</a>.</p>

**Step 4** Click **Install**.

----End

## Components

**Table 13-40** Add-on components

Component	Description	Resource Type
metrics-server	Aggregator for the monitored data of cluster core resources, which is used to collect and aggregate resource usage metrics obtained through the Metrics API in the cluster	Deployment

## Change History

**Table 13-41** Release history

Add-on Version	Supported Cluster Version	New Feature	Community Version
1.3.60	v1.21 v1.23 v1.25 v1.27 v1.28 v1.29	CCE clusters 1.29 are supported.	<a href="#">0.6.2</a>
1.3.37	v1.21 v1.23 v1.25 v1.27 v1.28	CCE clusters 1.28 are supported.	<a href="#">0.6.2</a>
1.3.12	v1.19 v1.21 v1.23 v1.25 v1.27	None	<a href="#">0.6.2</a>
1.3.8	v1.19 v1.21 v1.23 v1.25	Synchronized time zones used by the add-on and the node.	<a href="#">0.6.2</a>

Add-on Version	Supported Cluster Version	New Feature	Community Version
1.3.6	v1.19 v1.21 v1.23 v1.25	<ul style="list-style-type: none"> <li>Supported anti-affinity scheduling of add-on pods on nodes in different AZs.</li> <li>The default taint tolerance duration is changed to 60s.</li> </ul>	<a href="#">0.6.2</a>
1.3.3	v1.19 v1.21 v1.23 v1.25	<ul style="list-style-type: none"> <li>CCE clusters 1.25 are supported.</li> <li>Allowed CronHPA to adjust the number of Deployment pods with the skip scenario supported.</li> </ul>	<a href="#">0.6.2</a>
1.3.2	v1.19 v1.21 v1.23 v1.25	CCE clusters 1.25 are supported.	<a href="#">0.6.2</a>
1.2.1	v1.19 v1.21 v1.23	CCE clusters 1.23 are supported.	<a href="#">0.4.4</a>
1.1.10	v1.15 v1.17 v1.19 v1.21	CCE clusters 1.21 are supported.	<a href="#">0.4.4</a>
1.1.4	v1.15 v1.17 v1.19	Unified resource specification configuration unit.	<a href="#">0.4.4</a>
1.1.2	v1.15 v1.17 v1.19	Updated the add-on to its community version v0.4.4.	<a href="#">0.4.4</a>
1.1.1	v1.13 v1.15 v1.17 v1.19	Allows you to change the maximum number of invalid pods to 1.	<a href="#">0.3.7</a>
1.1.0	v1.13 v1.15 v1.17 v1.19	CCE clusters 1.19 are supported.	<a href="#">0.3.7</a>

Add-on Version	Supported Cluster Version	New Feature	Community Version
1.0.5	v1.13 v1.15 v1.17	Updated the add-on to its community version v0.3.7.	<b>0.3.7</b>

## 13.3.5 Prometheus

### Introduction

Prometheus is an open-source system monitoring and alerting framework. It is derived from Google's borgmon monitoring system, which was created by former Google employees working at SoundCloud in 2012. Prometheus was developed as an open-source community project and officially released in 2015. In 2016, Prometheus officially joined the Cloud Native Computing Foundation, after Kubernetes.

CCE allows you to quickly install Prometheus as an add-on.

Official website of Prometheus: <https://prometheus.io/>

Open source community: <https://github.com/prometheus/prometheus>

### Notes and Constraints

The Prometheus add-on is supported only in clusters of v1.21 and earlier.

### Features

As a next-generation monitoring framework, Prometheus has the following features:

- Powerful multi-dimensional data model
  - a. Time series data is identified by metric name and key-value pair.
  - b. Multi-dimensional labels can be set for all metrics.
  - c. Data models do not require dot-separated character strings.
  - d. Data models can be aggregated, cut, and sliced.
  - e. The double floating-point format is supported. Labels can all be set to unicode.
- Flexible and powerful query statement (PromQL): One query statement supports addition, multiplication, and connection for multiple metrics.
- Easy to manage: The Prometheus server is a separate binary file that can work locally. It does not depend on distributed storage.
- Efficient: Each sampling point occupies only 3.5 bytes, and one Prometheus server can process millions of metrics.
- The pull mode is used to collect time series data, which facilitates local tests and prevents faulty servers from pushing bad metrics.



- Time series data can be pushed to the Prometheus server in push gateway mode.
- Users can obtain the monitored targets through service discovery or static configuration.
- Multiple visual GUIs are available.
- Easy to scale

## Installing the Add-on

**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 **Prometheus** on the right, and click **Install**.

**Step 2** In the **Configuration** step, set the following parameters:

**Table 13-42** Prometheus add-on parameters

Parameter	Description
Add-on Specifications	Select an add-on specification based on service requirements. The options are as follows: <ul style="list-style-type: none"> <li>• <b>Demo(&lt;= 100 containers)</b>: The specification type applies to the experience and function demonstration environment. In this specification, Prometheus occupies few resources but has limited processing capabilities. You are advised to use this specification when the number of containers in the cluster does not exceed 100.</li> <li>• <b>Small(&lt;= 2000 containers)</b>: You are advised to use this specification when the number of containers in the cluster does not exceed 2,000.</li> <li>• <b>Medium(&lt;= 5000 containers)</b>: You are advised to use this specification when the number of containers in the cluster does not exceed 5000.</li> <li>• <b>Large(&gt; 5000 containers)</b>: You are advised to use this specification when the number of containers in the cluster exceeds 5,000.</li> </ul>
Pods	Number of pods that will be created to match the selected add-on specifications. The number cannot be modified.
Containers	CPU and memory quotas of the container allowed for the selected add-on specifications. The quotas cannot be modified.
Data Retention (days)	Number of days for storing customized monitoring data. The default value is 15 days.

Parameter	Description
Storage	<p>Cloud hard disks can be used as storage. Set the following parameters as prompted:</p> <ul style="list-style-type: none"> <li>• <b>AZ:</b> Set this parameter based on the site requirements. An AZ is a physical region where resources use independent power supply and networks. AZs are physically isolated but interconnected through an internal network.</li> <li>• <b>Disk Type:</b> Common I/O, high I/O, and ultra-high I/O are supported.</li> <li>• <b>Capacity:</b> Enter the storage capacity based on service requirements. The default value is 10 GB.</li> </ul> <p><b>NOTE</b> If a PVC already exists in the namespace monitoring, the configured storage will be used as the storage source.</p>

**Step 3** Click **Install**. After the installation, the add-on deploys the following instances in the cluster.

- prometheus-operator: deploys and manages the Prometheus Server based on CustomResourceDefinitions (CRDs), and monitors and processes the events related to these CRDs. It is the control center of the entire system.
- prometheus (server): a Prometheus Server cluster deployed by the operator based on the Prometheus CRDs that can be regarded as StatefulSets.
- prometheus-kube-state-metrics: converts the Prometheus metric data into a format that can be identified by Kubernetes APIs.
- custom-metrics-apiserver: aggregates custom metrics to the native Kubernetes API server.
- prometheus-node-exporter: deployed on each node to collect node monitoring data.
- grafana: visualizes monitoring data.

----End

## Providing Resource Metrics Through the Metrics API

Resource metrics of containers and nodes, such as CPU and memory usage, can be obtained through the Kubernetes Metrics API. Resource metrics can be directly accessed, for example, by using the **kubectl top** command, or used by HPA or CustomedHPA policies for auto scaling.

The add-on can provide the Kubernetes Metrics API that is disabled by default. To enable the API, create the following APIService object:

```
apiVersion: apiregistration.k8s.io/v1
kind: APIService
metadata:
  labels:
    app: custom-metrics-apiserver
    release: cceaddon-prometheus
    name: v1beta1.metrics.k8s.io
spec:
```

```
group: metrics.k8s.io
groupPriorityMinimum: 100
insecureSkipTLSVerify: true
service:
  name: custom-metrics-apiserver
  namespace: monitoring
  port: 443
version: v1beta1
versionPriority: 100
```

You can save the object as a file, name it **metrics-apiservice.yaml**, and run the following command:

```
kubectl create -f metrics-apiservice.yaml
```

Run the **kubectl top pod -n monitoring** command. If the following information is displayed, the Metrics API can be accessed:

```
# kubectl top pod -n monitoring
NAME                                CPU(cores)  MEMORY(bytes)
.....
custom-metrics-apiserver-d4f556ff9-l2j2m    38m        44Mi
.....
```

#### NOTICE

To uninstall the add-on, run the following kubectl command and delete the APIService object. Otherwise, the metrics-server add-on cannot be installed due to residual APIService resources.

```
kubectl delete APIService v1beta1.metrics.k8s.io
```

## Reference

- For details about the Prometheus concepts and configurations, see the [Prometheus Official Documentation](#).
- For details about how to install Node Exporter, see the [node\\_exporter GitHub](#).

## Change History

**Table 13-43** Release history

Add-on Version	Supported Cluster Version	New Feature	Community Version
2.23.32	v1.17 v1.19 v1.21	None	<b>2.10.0</b>
2.23.31	v1.15	<ul style="list-style-type: none"> <li>• CCE clusters 1.15 are supported.</li> </ul>	<b>2.10.0</b>

Add-on Version	Supported Cluster Version	New Feature	Community Version
2.23.30	v1.17 v1.19 v1.21	<ul style="list-style-type: none"> <li>CCE clusters 1.21 are supported.</li> </ul>	<a href="#">2.10.0</a>
2.21.14	v1.17 v1.19 v1.21	<ul style="list-style-type: none"> <li>CCE clusters 1.21 are supported.</li> </ul>	<a href="#">2.10.0</a>
2.21.12	v1.15	<ul style="list-style-type: none"> <li>CCE clusters 1.15 are supported.</li> </ul>	<a href="#">2.10.0</a>
2.21.11	v1.17 v1.19	<ul style="list-style-type: none"> <li>CCE clusters 1.19 are supported.</li> </ul>	<a href="#">2.10.0</a>
1.15.1	v1.15 v1.17	<ul style="list-style-type: none"> <li>The add-on is a monitoring system and time series library.</li> </ul>	<a href="#">2.10.0</a>

## 13.4 Cloud Native Heterogeneous Computing Add-ons

### 13.4.1 CCE AI Suite (NVIDIA GPU)

#### Introduction

NVIDIA GPU is a device management add-on that supports GPUs in containers. To use GPU nodes in a cluster, this add-on must be installed.

#### Notes and Constraints

- The driver to be downloaded must be a **.run** file.
- Only NVIDIA Tesla drivers are supported, not GRID drivers.
- When installing or reinstalling the add-on, ensure that the driver download address is correct and accessible. CCE does not verify the address validity.
- The `gpu-beta` add-on only enables you to download the driver and execute the installation script. The add-on status only indicates that how the add-on is running, not whether the driver is successfully installed.
- CCE does not guarantee the compatibility between the GPU driver version and the CUDA library version of your application. You need to check the compatibility by yourself.
- If a custom OS image has had a GPU driver installed, CCE cannot ensure that the GPU driver is compatible with other GPU components such as the monitoring components used in CCE.
- If the version of the GPU driver you used is not included in the [Supported GPU Drivers](#), the GPU driver may be incompatible with the OS, instance type,

or container runtime. As a result, the driver installation may fail or the GPU add-on may be abnormal. If you use a customized GPU driver, verify its availability.

## Installing the Add-on

**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 **CCE AI Suite (NVIDIA GPU)** on the right, and click **Install**.

**Step 2** Configure the add-on parameters.

**Table 13-44** Add-on parameters

Parameter	Description
Default Cluster Driver	<p>All GPU nodes in a cluster use the same driver. You can select a proper GPU driver version or customize the driver link and enter the download link of the NVIDIA driver.</p> <p><b>NOTICE</b></p> <ul style="list-style-type: none"> <li>• If the download link is a public network address, for example, <a href="https://us.download.nvidia.com/tesla/470.103.01/NVIDIA-Linux-x86_64-470.103.01.run">https://us.download.nvidia.com/tesla/470.103.01/NVIDIA-Linux-x86_64-470.103.01.run</a>, bind an EIP to each GPU node. For details about how to obtain the driver link, see <a href="#">Obtaining the Driver Link from Public Network</a>.</li> <li>• If the download link is an OBS URL, you do not need to bind an EIP to GPU nodes. For details about how to obtain the driver link, see <a href="#">Obtaining the Driver Link from OBS</a>.</li> <li>• Ensure that the NVIDIA driver version matches the GPU node. For details about the version mapping, see <a href="#">Supported GPU Drivers</a>.</li> <li>• If the driver version is changed, restart the node to apply the change.</li> <li>• Use driver version 470 or later for Huawei Cloud EulerOS 2.0 on which Linux Kernel 5.x is built, and driver 515 or later for Ubuntu 22.04.</li> </ul>

 **NOTE**

After the add-on is installed, you can configure GPU virtualization and node pool drivers on the **Heterogeneous Resources** tab in **Settings**.

**Step 3** Click **Install**.

 **NOTE**

If the add-on is uninstalled, GPU pods newly scheduled to the nodes cannot run properly, but GPU pods already running on the nodes will not be affected.

----End

## Verifying the Add-on

After the add-on is installed, run the **nvidia-smi** command on the GPU node and the container that schedules GPU resources to verify the availability of the GPU device and driver.

- GPU node:  
# If the add-on version is earlier than 2.0.0, run the following command:  
`cd /opt/cloud/cce/nvidia/bin && ./nvidia-smi`  
  
# If the add-on version is 2.0.0 or later and the driver installation path is changed, run the following command:  
`cd /usr/local/nvidia/bin && ./nvidia-smi`
- Container:  
`cd /usr/local/nvidia/bin && ./nvidia-smi`

If GPU information is returned, the device is available and the add-on has been installed.

```

+-----+
| NVIDIA-SMI 440.118.02   Driver Version: 440.118.02   CUDA Version: 10.2   |
+-----+-----+
| GPU  Name           Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp   Perf    Pwr:Usage/Cap|  Memory-Usage | GPU-Util  Compute M. |
+-----+-----+-----+-----+-----+-----+
|   0   Tesla V100-SXM2...    Off   | 00000000:21:01.0 Off  |
| N/A   31C    P0     23W / 300W |  0MiB / 16160MiB |      0%    Default  |
+-----+-----+-----+-----+-----+-----+
+-----+-----+
| Processes:
| GPU      PID   Type   Process name                               GPU Memory
|-----|-----|-----|-----|-----|
| No running processes found
+-----+-----+

```

## Supported GPU Drivers

### NOTICE

- The list of supported GPU drivers applies only to GPU add-ons of 1.2.28 and later versions.
- If you want to use the latest GPU driver, upgrade your GPU add-on to the latest version.

**Table 13-45** Supported GPU drivers

GPU Model	Supported Cluster Type	Specification	OS							
			Huawei Cloud EulerOS 2.0 (GPU Virtualization Supported)	Ubuntu 22.04.4	Ubuntu 22.04.3	CentOS Linux release 7.6	EulerOS release 2.9	EulerOS release 2.5	Ubuntu 18.04 (end of maintenance)	EulerOS release 2.3 (end of maintenance)
Tesla T4	CCE standard cluster	g6 pi2	535.54.03 510.47.03 470.57.02	535.161.08	535.54.03 470.141.03	535.54.03 470.141.03	535.54.03 470.141.03	535.54.03 470.141.03	470.141.03	470.141.03
Volta V100	CCE standard cluster	p2s p2vs p2v	535.54.03 510.47.03 470.57.02	535.161.08	535.54.03 470.141.03	535.54.03 470.141.03	535.54.03 470.141.03	535.54.03 470.141.03	470.141.03	470.141.03

## Obtaining the Driver Link from Public Network

- Step 1** Log in to the CCE console.
- Step 2** Click **Create Node** and select the GPU node to be created in the **Specifications** area. The GPU card model of the node is displayed in the lower part of the page.
- Step 3** Visit <https://www.nvidia.com/Download/Find.aspx?lang=en>.
- Step 4** Select the driver information on the **NVIDIA Driver Downloads** page, as shown in **Figure 13-7**. **Operating System** must be **Linux 64-bit**.

**Figure 13-7** Setting parameters

### NVIDIA Driver Downloads

Official Advanced Driver Search | NVIDIA

<b>Product Type:</b> Data Center / Tesla	<b>Operating System:</b> Linux 64-bit
<b>Product Series:</b> V-Series	<b>CUDA Toolkit:</b> Any
<b>Product:</b> Tesla V100	<b>Language:</b> English (US)
	<b>Recommended/Beta:</b> All ?

Click the Search button to perform your search.

**Step 5** After confirming the driver information, click **SEARCH**. A page is displayed, showing the driver information, as shown in **Figure 13-8**. Click **DOWNLOAD**.

**Figure 13-8** Driver information

### Data Center Driver For Linux X64

<b>Version:</b>	470.103.01
<b>Release Date:</b>	2022.1.31
<b>Operating System:</b>	Linux 64-bit
<b>CUDA Toolkit:</b>	11.4
<b>Language:</b>	English (US)
<b>File Size:</b>	259.86 MB

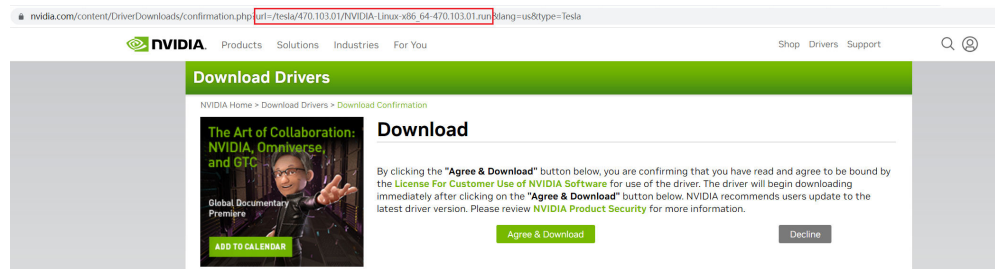
Release Highlights	Supported Products	Additional Information
Release notes, supported GPUs and other documentation can be found at: <a href="https://docs.nvidia.com/datacenter/tesla/index.html">https://docs.nvidia.com/datacenter/tesla/index.html</a>		

**Step 6** Obtain the driver link in either of the following ways:

- Method 1: As shown in **Figure 13-9**, find `url=/tesla/470.103.01/NVIDIA-Linux-x86_64-470.103.01.run` in the browser address box. Then, supplement it to obtain the driver link [https://us.download.nvidia.com/tesla/470.103.01/NVIDIA-Linux-x86\\_64-470.103.01.run](https://us.download.nvidia.com/tesla/470.103.01/NVIDIA-Linux-x86_64-470.103.01.run). By using this method, you must bind an EIP to each GPU node.
- Method 2: As shown in **Figure 13-9**, click **AGREE & DOWNLOAD** to download the driver. Then, upload the driver to OBS and record the OBS URL. By using this method, you do not need to bind an EIP to GPU nodes.



Figure 13-9 Obtaining the link



----End

## Obtaining the Driver Link from OBS

**Step 1** Upload the driver to OBS and set the driver file to public read.

 **NOTE**

When the node is restarted, the driver will be downloaded and installed again. Ensure that the OBS bucket link of the driver is valid.

**Step 2** In the bucket list, click a bucket name, and then the **Overview** page of the bucket is displayed.

**Step 3** In the navigation pane, choose **Objects**.

**Step 4** Select the name of the target object and copy the driver link on the object details page.

----End

## Components

Table 13-46 Add-on components

Component	Description	Resource Type
nvidia-driver-installer	A workload for installing the NVIDIA GPU driver on a node, which only uses resources during the installation process (Once the installation is finished, no resources are used.)	DaemonSet
nvidia-gpu-device-plugin	A Kubernetes device plugin that provides NVIDIA GPU heterogeneous compute for containers	DaemonSet
nvidia-operator	A component that provides NVIDIA GPU node management capabilities for clusters	Deployment

## Change History

**Table 13-47** Release history

Add-on Version	Supported Cluster Version	New Feature
2.6.4	v1.28 v1.29	Updated the isolation logic of GPU cards.
2.5.4	v1.28	Clusters 1.28 are supported.
2.0.46	v1.21 v1.23 v1.25 v1.27	<ul style="list-style-type: none"> <li>Supported Nvidia driver 535.</li> <li>Non-root users can use xGPUs.</li> <li>Optimized startup logic.</li> </ul>
2.0.18	v1.21 v1.23 v1.25 v1.27	Supported Huawei Cloud EulerOS 2.0.
1.2.28	v1.19 v1.21 v1.23 v1.25	<ul style="list-style-type: none"> <li>Optimized the automatic mounting of the GPU driver directory.</li> </ul>
1.2.24	v1.19 v1.21 v1.23 v1.25	<ul style="list-style-type: none"> <li>Enabled a node pool to configure GPU driver versions.</li> <li>Supported GPU metric collection.</li> </ul>
1.2.20	v1.19 v1.21 v1.23 v1.25	Set the add-on alias to <b>gpu</b> .
1.2.17	v1.15 v1.17 v1.19 v1.21 v1.23	Added the nvidia-driver-install pod limits configuration.

Add-on Version	Supported Cluster Version	New Feature
1.2.15	v1.15 v1.17 v1.19 v1.21 v1.23	CCE clusters 1.23 are supported.
1.2.11	v1.15 v1.17 v1.19 v1.21	Supported EulerOS 2.10.
1.2.10	v1.15 v1.17 v1.19 v1.21	CentOS supports the GPU driver of the new version.
1.2.9	v1.15 v1.17 v1.19 v1.21	CCE clusters 1.21 are supported.
1.2.2	v1.15 v1.17 v1.19	Supported the new EulerOS kernel.
1.2.1	v1.15 v1.17 v1.19	<ul style="list-style-type: none"> <li>• CCE clusters 1.19 are supported.</li> <li>• Added taint tolerance configuration.</li> </ul>
1.1.13	v1.13 v1.15 v1.17	Supported kernel-3.10.0-1127.19.1.el7.x86_64 for CentOS 7.6.
1.1.11	v1.15 v1.17	<ul style="list-style-type: none"> <li>• Allowed users to customize driver addresses to download drivers.</li> <li>• Clusters 1.15 and 1.17 are supported.</li> </ul>

## 13.4.2 CCE AI Suite (Ascend NPU)

### Introduction

CCE AI Suite (Ascend NPU) supports and manages NPUs in containers.

After this add-on is installed, you can create Ascend-accelerated nodes to quickly and efficiently process inference and image recognition.

### Notes and Constraints

- To use Ascend-accelerated nodes in a cluster, the Ascend NPU add-on must be installed.
- After an AI-accelerated node is migrated, the node will be reset. Manually reinstall the NPU driver.

### Installing the Add-on

- 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 **CCE AI Suite (Ascend NPU)** on the right, and click **Install**.
- Step 2** On the **Install Add-on** page, configure the specifications as needed. You can adjust the number of add-on instances and resource quotas as required.
- Step 3** Determine whether to automatically install the driver (supported only when the add-on version is 1.2.5 or later).

- **Enabled:** You can specify the driver version based on the NPU model for easier driver maintenance.

After the driver is enabled, the add-on automatically installs the driver based on the specified driver version. By default, the recommended driver is used. You can also select **Custom driver** from the drop-down list and enter a driver address.

#### NOTE

- The add-on installs the driver based on the driver version selected for the specified model. Such installation is only for nodes with no NPU driver installed. Nodes with an NPU driver installed remain unchanged. If you change the driver version when upgrading the add-on or updating add-on parameters, such change takes effect only on the nodes with no NPU driver installed.
- After the driver is successfully installed, restart the node for the driver to take effect. For details about how to check whether the driver is successfully installed, see [How to Check Whether the NPU Driver Has Been Installed on a Node](#).
- Uninstalling the add-on does not automatically delete the installed NPU driver. For details about how to uninstall the NPU driver, see [Uninstalling the NPU Driver](#).
- **Disabled:** Driver versions are decided by the system, and the drivers cannot be maintained using the add-on. When you add an NPU node on the console, the console adds the command to install the NPU driver (version and type decided by the system) and automatically restarts the node after the installation is complete. Adding an NPU node in another way, such as using an API, requires you to add the driver installation command to **Post-installation Command**.

- The supported NPU types and OSs are as follows:

NPU Type	Supported OS
D310	EulerOS 2.5 x86, CentOS 7.6 x86, EulerOS 2.9 x86, and EulerOS 2.8 arm

**Step 4** Click **Install**.

----End

## Components

**Table 13-48** Add-on components

Component	Description	Resource Type
npu-driver-installer	Used for installing an NPU driver on NPU nodes.	DaemonSet
huawei-npu-device-plugin	The CCE AI Suite (Ascend NPU) add-on can be used in containers.	DaemonSet

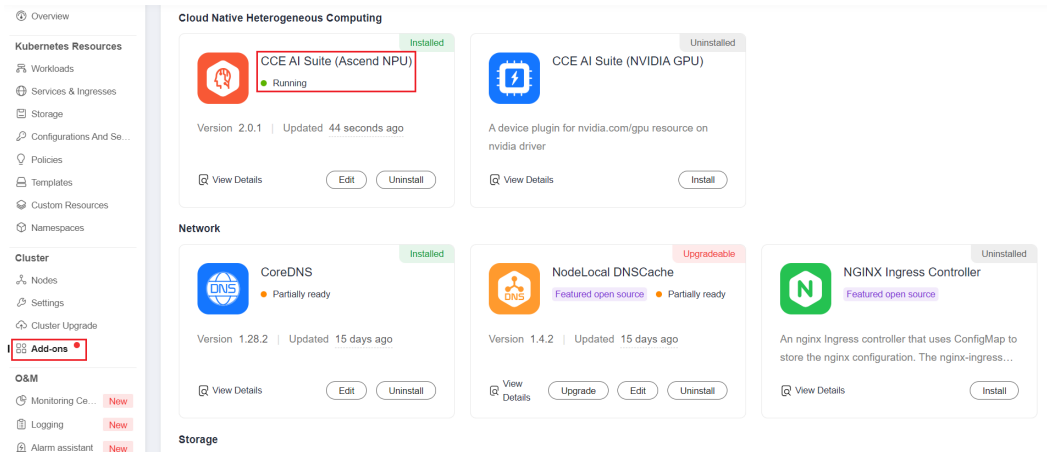
## NPU Metrics

Metric	Monitoring Level	Remarks
cce_npu_memory_total	NPU cards	Total NPU memory
cce_npu_memory_used	NPU cards	NPU memory usage
cce_npu_utilization	NPU cards	NPU compute usage

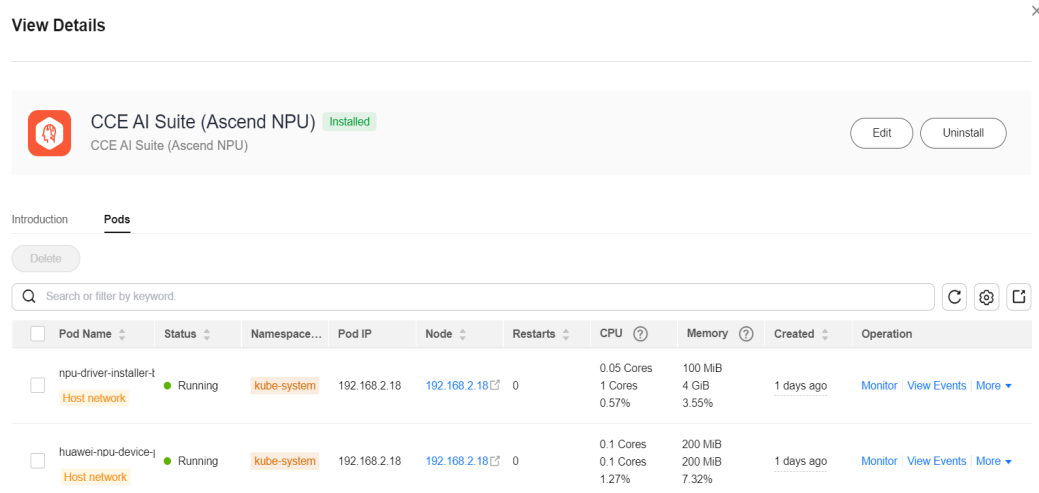
## How to Check Whether the NPU Driver Has Been Installed on a Node

After ensuring that the driver is successfully installed, restart the node for the driver to take effect. Otherwise, the driver cannot take effect and NPU resources are unavailable. To check whether the driver is installed, perform the following operations:

**Step 1** On the **Add-ons** page, click **CCE AI Suite (Ascend NPU)**.



**Step 2** Verify that the node where npu-driver-installer is deployed is in the **Running** state.



**NOTE**

If the node is restarted before the NPU driver is installed, the driver installation may fail and a message is displayed on the **Nodes** page of the cluster indicating that the Ascend driver is not ready. In this case, uninstall the NPU driver from the node and restart the **npu-driver-installer** pod to reinstall the NPU driver. After confirming that the driver is installed, restart the node. For details about how to uninstall the driver, see [Uninstalling the NPU Driver](#).

----End

## Uninstalling the NPU Driver

Log in to the node, obtain the driver operation records in the **/var/log/ascend\_seclog/operation.log** file, and find the driver run package used in last installation. If the log file does not exist, the driver is installed using the **npu\_x86\_latest.run** or **npu\_arm\_latest.run** driver combined package. After finding the driver installation package, run the **bash {run package name} --uninstall** command to uninstall the driver and restart the node as prompted.

**Step 1** Log in to the node where the NPU driver needs to be uninstalled and find the **/var/log/ascend\_seclog/operation.log** file.

**Step 2** If the `/var/log/ascend_seclog/operation.log` file can be found, view the driver installation log to find the driver installation record.

```
[root@00379955-w-ails-e25 ~]# ll /var/log/ascend_seclog/operation.log
-rw-r----- 1 root root 285 Dec 1 20:00 /var/log/ascend_seclog/operation.log
[root@00379955-w-ails-e25 ~]# cat /var/log/ascend_seclog/operation.log
Install SUGGESTION root 2022-12-01 19:53:47 127.0.0.1 Ascend310-hdk-npu-driver 6.0.rc1 linux-x86-64.run success install_type=full; cmdlist=-quiet --full.
```

If the `/var/log/ascend_seclog/operation.log` file cannot be found, the driver may be installed using the `npu_x86_latest.run` or `npu_arm_latest.run` driver combined package. You can confirm this by checking whether the `/usr/local/HiAI/driver/` directory exists.

**NOTE**

The combined package of the NPU driver is stored in the `/root/d310_driver` directory, and other driver installation packages are stored in the `/root/npu-drivers` directory.

**Step 3** After finding the driver installation package, run the `bash {run package path} --uninstall` command to uninstall the driver. The following uses `Ascend310-hdk-npu-driver_6.0.rc1_linux-x86-64.run` as an example:

```
bash /root/npu-drivers/Ascend310-hdk-npu-driver_6.0.rc1_linux-x86-64.run --uninstall
```

```
[root@y00379955-w-ails-e25 npu-drivers]# ./Ascend310-hdk-npu-driver_6.0.rc1_linux-x86-64.run --uninstall
Verifying archive integrity... 100% SHA256 checksums are OK. All good.
Uncompressing ASCEND DRIVER RUN PACKAGE 100%
[Driver] [2022-12-01 19:59:53] [INFO]Start time: 2022-12-01 19:59:53
[Driver] [2022-12-01 19:59:53] [INFO]LogFile: /var/log/ascend_seclog/ascend_install.log
[Driver] [2022-12-01 19:59:53] [INFO]OperationLogFile: /var/log/ascend_seclog/operation.log
[Driver] [2022-12-01 19:59:53] [INFO]base version is 22.0.3.

[Driver] [2022-12-01 20:00:04] [INFO]Driver package uninstalled successfully! Reboot needed for uninstallation to take effect!
[Driver] [2022-12-01 20:00:04] [INFO]End time: 2022-12-01 20:00:04
```

**Step 4** Restart the node as prompted. (The installation and uninstallation of the current NPU driver take effect only after the node is restarted.)

----End

## Change History

**Table 13-49** Release history

Add-on Version	Supported Cluster Version	New Feature
2.1.7	v1.21 v1.23 v1.25 v1.27 v1.28 v1.29	Fixed some issues.

Add-on Version	Supported Cluster Version	New Feature
2.1.5	v1.21 v1.23 v1.25 v1.27 v1.28 v1.29	<ul style="list-style-type: none"> <li>• CCE clusters 1.29 are supported.</li> <li>• Added silent fault codes.</li> </ul>
2.0.5	v1.21 v1.23 v1.25 v1.27 v1.28	<ul style="list-style-type: none"> <li>• CCE clusters 1.28 are supported.</li> <li>• Supported liveness probe.</li> </ul>
1.2.14	v1.19 v1.21 v1.23 v1.25 v1.27	Supported NPU monitoring.
1.2.6	v1.19 v1.21 v1.23 v1.25	Supports automatic installation of NPU drivers.
1.2.5	v1.19 v1.21 v1.23 v1.25	Supports automatic installation of NPU drivers.
1.2.4	v1.19 v1.21 v1.23 v1.25	CCE clusters 1.25 are supported.
1.2.2	v1.19 v1.21 v1.23	CCE clusters 1.23 are supported.
1.2.1	v1.19 v1.21 v1.23	CCE clusters 1.23 are supported.



Add-on Version	Supported Cluster Version	New Feature
1.1.8	v1.15 v1.17 v1.19 v1.21	CCE clusters 1.21 are supported.
1.1.2	v1.15 v1.17 v1.19	Adds the default seccomp profile.
1.1.1	v1.15 v1.17 v1.19	CCE clusters 1.15 are supported.
1.1.0	v1.17 v1.19	CCE clusters 1.19 are supported.
1.0.8	v1.13 v1.15 v1.17	Adapts to the D310 C75 driver.
1.0.6	v1.13 v1.15 v1.17	Supports the Ascend C75 driver.
1.0.5	v1.13 v1.15 v1.17	Allows containers to use Huawei NPU add-ons.
1.0.3	v1.13 v1.15 v1.17	Allows containers to use Huawei NPU add-ons.

## 13.5 Container Network Add-ons

### 13.5.1 CoreDNS

#### Introduction

CoreDNS is a DNS server that provides domain name resolution for Kubernetes clusters through a chain add-on.

CoreDNS is an open-source software and has been a part of CNCF. It provides a means for cloud services to discover each other in cloud-native deployments. Each of the plugins chained by CoreDNS provides a particular DNS function. You can

integrate CoreDNS with only the plugins you need to make it fast, efficient, and flexible. When used in a Kubernetes cluster, CoreDNS can automatically discover services in the cluster and provide domain name resolution for these services. By working with DNS server, CoreDNS can resolve external domain names for workloads in a cluster.

**This add-on is installed by default during cluster creation.**

Kubernetes backs CoreDNS as the official default DNS for all clusters going forward.

CoreDNS official website: <https://coredns.io/>

Open source community: <https://github.com/coredns/coredns>

 **NOTE**

For details, see [DNS](#).

## Notes and Constraints

To run CoreDNS properly or upgrade CoreDNS in a cluster, ensure the number of available nodes in the cluster is greater than or equal to the number of CoreDNS instances and all CoreDNS instances are running. Otherwise, the add-on will malfunction or the upgrade will fail.

## Installing the Add-on

This add-on has been installed by default. If it is uninstalled due to some reasons, you can reinstall it by performing the following steps:

**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 **Install**.

**Step 2** On the **Install Add-on** page, configure the specifications as needed.

- If you selected **Preset**, you can choose between **Small**, **Medium**, or **Large** as needed. The system will automatically set the number of add-on pods and resource quotas according to the preset specifications. You can see the configurations on the console.

The small one can handle up to 2500 external and 10,000 internal domain names QPS. The medium specification can handle up to 5000 external and 20,000 internal domain names QPS. The large specification can handle up to 10,000 external and 40,000 internal domain names QPS.

- If you selected **Custom**, you can adjust the number of pods and resource quotas as needed. QPS of the CoreDNS add-on is positively correlated with the CPU consumption. If the number of nodes or containers in the cluster grows, the CoreDNS pod will bear heavier workloads. It is recommended that you adjust the number of the CoreDNS pods and their CPU and memory quotas based on the cluster scale. For details, see [Table 13-50](#).

**Table 13-50** Recommended CoreDNS quotas

Nodes	Recommended QPS	Pods	Requested vCPUs	vCPU Limit	Requested Memory	Memory Limit
50	2500	2	500m	500m	512 MiB	512 MiB
200	5000	2	1000m	1000m	1024 MiB	1024 MiB
1000	10000	2	2000m	2000m	2048 MiB	2048 MiB
2000	20000	4	2000m	2000m	2048 MiB	2048Mi

**Step 3** Configure the add-on parameters.

**Table 13-51** Add-on parameters

Parameter	Description
Stub Domain	A domain name server for a custom domain name. The format is a key-value pair. The key is a domain name suffix, and the value is one or more DNS IP addresses, for example, <b>acme.local -- 1.2.3.4,6.7.8.9</b> . For details, see <a href="#">Configuring the Stub Domain for CoreDNS</a> .

Parameter	Description
Extended Parameter Settings	<ul style="list-style-type: none"> <li>● <b>parameterSyncStrategy</b>: indicates whether to configure consistency check when the add-on is upgraded. <ul style="list-style-type: none"> <li>- <b>ensureConsistent</b>: indicates that the configuration consistency check is enabled. If the configuration provided during the add-on upgrade does not match the current effective configuration, the add-on cannot be upgraded.</li> <li>- <b>force</b>: indicates that the configuration consistency check is ignored during an upgrade. The configuration provided during the add-on upgrade will be used, so it is important to make sure that it matches the current effective configuration. After the add-on is upgraded, you need to restore the value of <b>parameterSyncStrategy</b> to <b>ensureConsistent</b> to enable the configuration consistency check again.</li> <li>- <b>inherit</b>: If the configuration provided during the add-on upgrade differs from the current effective configuration, the current effective configuration will be used instead. After the add-on is upgraded, you need to restore the value of <b>parameterSyncStrategy</b> to <b>ensureConsistent</b> to enable the configuration consistency check again.</li> </ul> </li> <li>● <b>servers</b>: nameservers, which are available in CoreDNS v1.23.1 and later versions. You can customize nameservers. For details, see <a href="#">dns-custom-nameservers</a>.  <b>plugins</b> indicates the configuration of each component in CoreDNS. Retain the default settings typically to prevent CoreDNS from being unavailable due to configuration errors. Each plugin component contains <b>name</b>, <b>parameters</b> (optional), and <b>configBlock</b> (optional). The format of the generated Corefile is as follows: <pre style="background-color: #f0f0f0; padding: 5px;">\$name \$parameters {   \$configBlock }</pre> <p><b>Table 13-52</b> describes common plugins. For details, see <a href="#">Plugins</a>.</p> </li> <li>● <b>upstream_nameservers</b>: specifies the IP address of the upstream DNS server.</li> </ul> <p>Example:</p> <pre style="background-color: #f0f0f0; padding: 5px;">{   "annotations": {},   "parameterSyncStrategy": "ensureConsistent",   "servers": [     {       "plugins": [         {           "name": "bind",           "parameters": "\${POD_IP}"         },         {           "name": "cache",           "configBlock": "servfail 5s",           "parameters": 30         }       ]     }   ] }</pre>

Parameter	Description
	<pre>         "name": "errors"       },       {         "name": "health",         "parameters": "{\$POD_IP}:8080"       },       {         "name": "ready",         "parameters": "{\$POD_IP}:8081"       },       {         "configBlock": "pods insecure\nfallthrough in-addr.arpa ip6.arpa",         "name": "kubernetes",         "parameters": "cluster.local in-addr.arpa ip6.arpa"       },       {         "name": "loadbalance",         "parameters": "round_robin"       },       {         "name": "prometheus",         "parameters": "{\$POD_IP}:9153"       },       {         "configBlock": "policy random",         "name": "forward",         "parameters": ". /etc/resolv.conf"       },       {         "name": "reload"       }     ],     "port": 5353,     "zones": [       {         "zone": "."       }     ]   },   "upstream_nameservers": ["8.8.8.8", "8.8.4.4"] } </pre>

**Table 13-52** Default CoreDNS configurations

Plugin Name	Type	Description
bind	Default configuration	Host IP address listened by CoreDNS. Retain the default value <code>{SPOD_IP}</code> . For details, see <a href="#">bind</a> .
cache	Default configuration	Enables DNS cache. For details, see <a href="#">cache</a> . If the add-on version is 1.25.10 or later, the servfail cache can be disabled. To disable the servfail cache, set <b>configBlock</b> to <b>servfail 0</b> . Otherwise, the unit of the servfail cache is second and cannot be omitted.

Plugin Name	Type	Description
errors	Default configuration	Errors are logged to stdout. For details, see <a href="#">errors</a> .
health	Default configuration	Health check for CoreDNS. {\$POD_IP}:8080 is listened to. Retain the default setting. Otherwise, the CoreDNS health check will fail and the add-on will restart repeatedly. For details, see <a href="#">health</a> .
ready	Default configuration	Whether the backend server is ready to receive traffic. {\$POD_IP}:8081 is listened to. If the backend server is not ready, CoreDNS will suspend DNS resolution until the backend server is ready. For details, see <a href="#">ready</a> .
kubernetes	Default configuration	CoreDNS Kubernetes plugin, which provides the service parsing capability in a cluster. For details, see <a href="#">kubernetes</a> .
loadbalance	Default configuration	Round-robin DNS load balancer that randomizes the order of A, AAAA, and MX records in an answer. For details, see <a href="#">loadbalance</a> .
prometheus	Default configuration	API for obtaining CoreDNS metrics. {\$POD_IP}:9153 is listened to by default. Retain the default setting. Otherwise, Prometheus cannot collect CoreDNS metrics. For details, see <a href="#">Prometheus</a> .
forward	Default configuration	Forwards any queries that are not within the cluster domain of Kubernetes to predefined resolvers ( <code>/etc/resolv.conf</code> ). For details, see <a href="#">forward</a> .
reload	Default configuration	Automatically reloads modified Corefiles. After you modify a ConfigMap, wait for two minutes for the modification to take effect. For details, see <a href="#">reload</a> .
log	Extended configuration	Enables CoreDNS logging. For details, see <a href="#">log</a> . Example: <pre>{   "name": "log" }</pre>
template	Extended configuration	A quick response template, where <b>AAAA</b> indicates an IPv6 request. If <b>NXDOMAIN</b> is returned in an <b>rcode</b> response, no IPv6 resolution result is returned. For details, see <a href="#">template</a> . Example: <pre>{   "configBlock": "rcode NXDOMAIN",   "name": "template",   "parameters": "ANY AAAA" }</pre>

**Step 4** Configure deployment policies for the add-on pods.

 **NOTE**

- Scheduling policies do not take effect on add-on instances of the DaemonSet type.
- When configuring multi-AZ deployment or node affinity, ensure that there are nodes meeting the scheduling policy and that resources are sufficient in the cluster. Otherwise, the add-on cannot run.

**Table 13-53** Configurations for add-on scheduling

Parameter	Description
Multi AZ	<ul style="list-style-type: none"> <li>• <b>Preferred:</b> Deployment pods of the add-on will be preferentially scheduled to nodes in different AZs. If all the nodes in the cluster are deployed in the same AZ, the pods will be scheduled to different nodes in that AZ.</li> <li>• <b>Equivalent mode:</b> Deployment pods of the add-on are evenly scheduled to the nodes in the cluster in each AZ. If a new AZ is added, you are advised to increase add-on pods for cross-AZ HA deployment. With the Equivalent multi-AZ deployment, the difference between the number of add-on pods in different AZs will be less than or equal to 1. If resources in one of the AZs are insufficient, pods cannot be scheduled to that AZ.</li> <li>• <b>Required:</b> Deployment pods of the add-on are forcibly scheduled to nodes in different AZs. There can be at most one pod in each AZ. If nodes in a cluster are not in different AZs, some add-on pods cannot run properly. If a node is faulty, add-on pods on it may fail to be migrated.</li> </ul>
Node Affinity	<ul style="list-style-type: none"> <li>• <b>Not configured:</b> Node affinity is disabled for the add-on.</li> <li>• <b>Node Affinity:</b> Specify the nodes where the add-on is deployed. If you do not specify the nodes, the add-on will be randomly scheduled based on the default cluster scheduling policy.</li> <li>• <b>Specified Node Pool Scheduling:</b> Specify the node pool where the add-on is deployed. If you do not specify the node pool, the add-on will be randomly scheduled based on the default cluster scheduling policy.</li> <li>• <b>Custom Policies:</b> Enter the labels of the nodes where the add-on is to be deployed for more flexible scheduling policies. If you do not specify node labels, the add-on will be randomly scheduled based on the default cluster scheduling policy.</li> </ul> <p>If multiple custom affinity policies are configured, ensure that there are nodes that meet all the affinity policies in the cluster. Otherwise, the add-on cannot run.</p>

Parameter	Description
Toleration	<p>Using both taints and tolerations allows (not forcibly) the add-on Deployment to be scheduled to a node with the matching taints, and controls the Deployment eviction policies after the node where the Deployment is located is tainted.</p> <p>The add-on adds the default tolerance policy for the <b>node.kubernetes.io/not-ready</b> and <b>node.kubernetes.io/unreachable</b> taints, respectively. The tolerance time window is 60s.</p> <p>For details, see <a href="#">Configuring Tolerance Policies</a>.</p>

**Step 5** Click **Install**.

----End

## Components

**Table 13-54** Add-on components

Component	Description	Resource Type
CoreDNS	DNS server for clusters	Deployment

## How Does Domain Name Resolution Work in Kubernetes?

DNS policies can be configured for each pod. Kubernetes supports DNS policies **Default**, **ClusterFirst**, **ClusterFirstWithHostNet**, and **None**. For details, see [DNS for Services and Pods](#). These policies are specified in the **dnsPolicy** field in the pod-specific.

- **Default:** Pods inherit the name resolution configuration from the node that the pods run on. The custom upstream DNS server and the stub domain cannot be used together with this policy.
- **ClusterFirst:** Any DNS query that does not match the configured cluster domain suffix, such as **www.kubernetes.io**, is forwarded to the upstream name server inherited from the node. Cluster administrators may have extra stub domains and upstream DNS servers configured.
- **ClusterFirstWithHostNet:** For pods running with **hostNetwork**, set its DNS policy **ClusterFirstWithHostNet**.
- **None:** It allows a pod to ignore DNS settings from the Kubernetes environment. All DNS settings are supposed to be provided using the **dnsPolicy** field in the pod-specific.



**NOTE**

- Clusters of Kubernetes v1.10 and later support **Default**, **ClusterFirst**, **ClusterFirstWithHostNet**, and **None**. Clusters earlier than Kubernetes v1.10 support only **Default**, **ClusterFirst**, and **ClusterFirstWithHostNet**.
- **Default** is not the default DNS policy. If **dnsPolicy** is not explicitly specified, **ClusterFirst** is used.

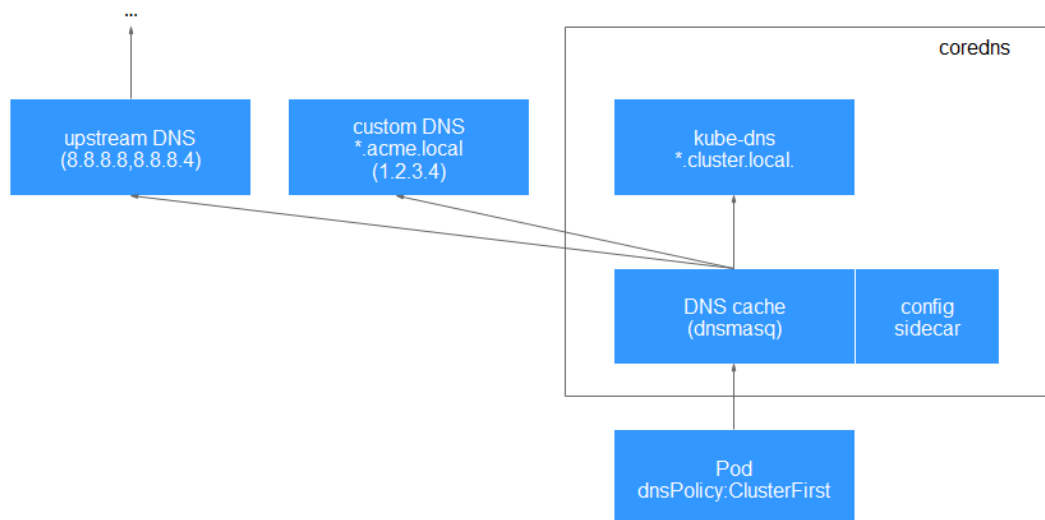
**Routing**

**Without stub domain configurations:** Any query that does not match the configured cluster domain suffix, such as **www.kubernetes.io**, is forwarded to the upstream DNS server inherited from the node.

**With stub domain configurations:** If stub domains and upstream DNS servers are configured, DNS queries are routed according to the following flow:

1. The query is first sent to the DNS caching layer in CoreDNS.
2. From the caching layer, the suffix of the request is examined and then the request is forwarded to the corresponding DNS:
  - Names with the cluster suffix, for example, **.cluster.local**: The request is sent to CoreDNS.
  - Names with the stub domain suffix, for example, **.acme.local**: The request is sent to the configured custom DNS resolver that listens, for example, on 1.2.3.4.
  - Names that do not match the suffix (for example, **widget.com**): The request is forwarded to the upstream DNS.

**Figure 13-10 Routing**



## Change History

**Table 13-55** Release history

Add-on Version	Supported Cluster Version	New Feature	Community Version
1.29.4	v1.21 v1.23 v1.25 v1.27 v1.28 v1.29	CCE clusters 1.29 are supported.	<a href="#">1.10.1</a>
1.28.4	v1.21 v1.23 v1.25 v1.27 v1.28	CCE clusters 1.28 are supported.	<a href="#">1.10.1</a>
1.27.4	v1.19 v1.21 v1.23 v1.25 v1.27	None	<a href="#">1.10.1</a>
1.25.14	v1.19 v1.21 v1.23 v1.25	<ul style="list-style-type: none"> <li>• Supports association between add-on specifications and cluster specifications.</li> <li>• Synchronized time zones used by the add-on and the node.</li> </ul>	<a href="#">1.10.1</a>
1.25.11	v1.19 v1.21 v1.23 v1.25	<ul style="list-style-type: none"> <li>• Supported anti-affinity scheduling of add-on pods on nodes in different AZs.</li> <li>• Upgrades to its community version 1.10.1.</li> </ul>	<a href="#">1.10.1</a>
1.25.1	v1.19 v1.21 v1.23 v1.25	CCE clusters 1.25 are supported.	<a href="#">1.8.4</a>

Add-on Version	Supported Cluster Version	New Feature	Community Version
1.23.3	v1.15 v1.17 v1.19 v1.21 v1.23	Regular upgrade of add-on dependencies	<a href="#">1.8.4</a>
1.23.2	v1.15 v1.17 v1.19 v1.21 v1.23	Regular upgrade of add-on dependencies	<a href="#">1.8.4</a>
1.23.1	v1.15 v1.17 v1.19 v1.21 v1.23	CCE clusters 1.23 are supported.	<a href="#">1.8.4</a>
1.17.15	v1.15 v1.17 v1.19 v1.21	CCE clusters 1.21 are supported.	<a href="#">1.8.4</a>
1.17.9	v1.15 v1.17 v1.19	Regular upgrade of add-on dependencies	<a href="#">1.8.4</a>
1.17.7	v1.15 v1.17 v1.19	Updated the add-on to its community version v1.8.4.	<a href="#">1.8.4</a>
1.17.4	v1.17 v1.19	CCE clusters 1.19 are supported.	<a href="#">1.6.5</a>
1.17.3	v1.17	Supported clusters 1.17 and fixed stub domain configuration issues.	<a href="#">1.6.5</a>
1.17.1	v1.17	Clusters 1.17 are supported.	<a href="#">1.6.5</a>

## 13.5.2 NGINX Ingress Controller

### Introduction

Kubernetes uses kube-proxy to expose Services and provide load balancing. The implementation is at the transport layer. When it comes to Internet applications, where a bucket-load of information is generated, forwarding needs to be more fine-grained, precisely and flexibly controlled by policies and load balancers to deliver higher performance.

This is where ingresses enter. Ingresses provide application-layer forwarding functions, such as virtual hosts, load balancing, SSL proxy, and HTTP routing, for Services that can be directly accessed outside a cluster.

Kubernetes has officially released the Nginx-based Ingress controller. CCE Nginx Ingress controller directly uses community templates and images. The NGINX Ingress Controller generates Nginx configuration and stores the configuration using ConfigMap. The configuration will be written to Nginx pods through the Kubernetes API. In this way, the Nginx configuration is modified and updated. For details, see [How the Add-on Works](#).

Open source community: <https://github.com/kubernetes/ingress-nginx>

#### NOTE

- Starting from version 2.3.3, NGINX Ingress Controller only supports TLS v1.2 and v1.3 by default. If the TLS version is earlier than v1.2, an error will occur during the negotiation process between the client and Nginx Ingress. If more versions of TLS are needed, see [TLS/HTTPS](#).
- When installing the NGINX Ingress Controller, you can specify Nginx parameters. These parameters take effect globally and are contained in the `nginx.conf` file. You can search for the parameters in [ConfigMaps](#). If the parameters are not included in [ConfigMaps](#), the configurations will not take effect.
- Do not manually modify or delete the load balancer and listener that are automatically created by CCE. Otherwise, the workload will be abnormal. If you have modified or deleted them by mistake, uninstall the nginx-ingress add-on and re-install it.

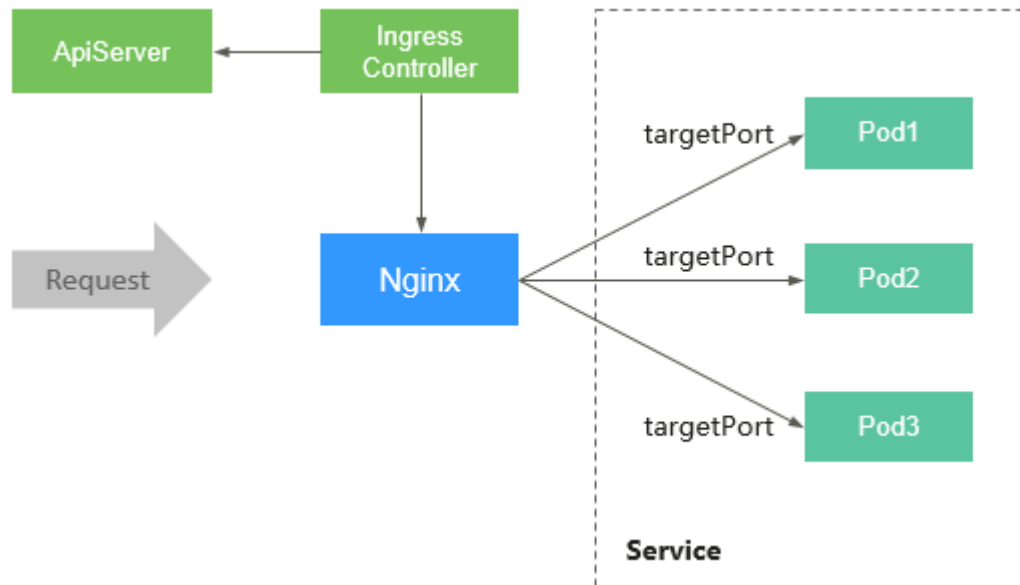
### How the Add-on Works

NGINX Ingress Controller consists of the ingress object, ingress controller, and Nginx. The ingress controller assembles ingresses into the Nginx configuration file (`nginx.conf`) and reloads Nginx to make the changed configurations take effect. When it detects that the pod in a Service changes, it dynamically changes the upstream server group configuration of Nginx. In this case, the Nginx process does not need to be reloaded. [Figure 13-11](#) shows how this add-on works.

- An ingress is a group of access rules that forward requests to specified Services based on domain names or URLs. Ingresses are stored in the object storage service etcd and are added, deleted, modified, and queried through APIs.
- The ingress controller monitors the changes of resource objects such as ingresses, Services, endpoints, secrets (mainly TLS certificates and keys), nodes, and ConfigMaps in real time and automatically performs operations on Nginx.

- Nginx implements load balancing and access control at the application layer.

**Figure 13-11** Working principles of NGINX Ingress Controller



## Precautions

- For clusters earlier than v1.23, **kubernetes.io/ingress.class: "nginx"** must be added to the annotation of the ingress created through the API.
- A dedicated load balancer must be of the network type (TCP/UDP) and support private networks (with a private IP address).
- If the node where NGINX Ingress Controller runs and containers on this node cannot access Nginx ingress, you need to configure anti-affinity for the workload pods and Nginx Ingress Controller. For details, see [Configuring Anti-affinity Between a Workload and Nginx Ingress Controller](#).
- During the NGINX Ingress Controller pod upgrade, 10s are reserved for deleting the NGINX Ingress Controller at the ELB backend.
- The timeout duration for the graceful exit of the NGINX Ingress Controller is 300s. If the timeout duration is longer than 300s during the upgrade of the NGINX Ingress Controller, persistent connections will be disconnected, and connectivity will be interrupted for a short period of time.

## Prerequisites

Before installing this add-on, you have one available cluster and there is a node running properly. If no cluster is available, create one according to [Buying a CCE Standard/Turbo Cluster](#).

## Installing the Add-on

- 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 **NGINX Ingress Controller** on the right, and click **Install**.

**Step 2** On the **Install Add-on** page, configure the specifications as needed.

You can adjust the number of add-on instances and resource quotas as required. High availability is not possible with a single pod. If an error occurs on the node where the add-on instance runs, the add-on will fail.

**Step 3** Configure the add-on parameters.

- **Load Balancer:** Select a shared or dedicated load balancer. If no load balancer is available, create one. The load balancer has at least two listeners, and ports 80 and 443 are not occupied by listeners.
- **Admission Check:** Admission control is performed on Ingresses to ensure that the controller can generate valid configurations. Admission verification is performed on the configuration of Nginx Ingresses. If the verification fails, the request will be intercepted. For details about admission verification, see [Access Control](#).

 **NOTE**

- Admission check slows down the responses to Ingress requests.
- Only add-ons of version 2.4.1 or later support admission verification.
- **Nginx Parameters:** You can configure the **nginx.conf** file, which will affect all managed ingresses.

To configure custom parameters supported by the Kubernetes community, find the related parameters in [ConfigMaps](#). For example, you can use the **keep-alive-requests** parameter to describe how to set the maximum number of requests for keeping active connections to 100.

```
{  
  "keep-alive-requests": "100"  
}
```

 **NOTE**

- If the parameters you configure are not listed in [ConfigMaps](#), the configurations will not take effect.
- The parameter value must be a string. Otherwise, the installation fails.
- **404 Service:** By default, the 404 service provided by the add-on is used. To customize the 404 service, enter the namespace/service name. If the service does not exist, the add-on installation will fail.

**Step 4** Configure deployment policies for the add-on pods.

 **NOTE**

- Scheduling policies do not take effect on add-on instances of the DaemonSet type.
- When configuring multi-AZ deployment or node affinity, ensure that there are nodes meeting the scheduling policy and that resources are sufficient in the cluster. Otherwise, the add-on cannot run.

**Table 13-56** Configurations for add-on scheduling

Parameter	Description
Multi AZ	<ul style="list-style-type: none"> <li>• <b>Preferred:</b> Deployment pods of the add-on will be preferentially scheduled to nodes in different AZs. If all the nodes in the cluster are deployed in the same AZ, the pods will be scheduled to different nodes in that AZ.</li> <li>• <b>Equivalent mode:</b> Deployment pods of the add-on are evenly scheduled to the nodes in the cluster in each AZ. If a new AZ is added, you are advised to increase add-on pods for cross-AZ HA deployment. With the Equivalent multi-AZ deployment, the difference between the number of add-on pods in different AZs will be less than or equal to 1. If resources in one of the AZs are insufficient, pods cannot be scheduled to that AZ.</li> <li>• <b>Required:</b> Deployment pods of the add-on are forcibly scheduled to nodes in different AZs. There can be at most one pod in each AZ. If nodes in a cluster are not in different AZs, some add-on pods cannot run properly. If a node is faulty, add-on pods on it may fail to be migrated.</li> </ul>
Node Affinity	<ul style="list-style-type: none"> <li>• <b>Not configured:</b> Node affinity is disabled for the add-on.</li> <li>• <b>Node Affinity:</b> Specify the nodes where the add-on is deployed. If you do not specify the nodes, the add-on will be randomly scheduled based on the default cluster scheduling policy.</li> <li>• <b>Specified Node Pool Scheduling:</b> Specify the node pool where the add-on is deployed. If you do not specify the node pool, the add-on will be randomly scheduled based on the default cluster scheduling policy.</li> <li>• <b>Custom Policies:</b> Enter the labels of the nodes where the add-on is to be deployed for more flexible scheduling policies. If you do not specify node labels, the add-on will be randomly scheduled based on the default cluster scheduling policy. If multiple custom affinity policies are configured, ensure that there are nodes that meet all the affinity policies in the cluster. Otherwise, the add-on cannot run.</li> </ul>
Toleration	<p>Using both taints and tolerations allows (not forcibly) the add-on Deployment to be scheduled to a node with the matching taints, and controls the Deployment eviction policies after the node where the Deployment is located is tainted.</p> <p>The add-on adds the default tolerance policy for the <b>node.kubernetes.io/not-ready</b> and <b>node.kubernetes.io/unreachable</b> taints, respectively. The tolerance time window is 60s.</p> <p>For details, see <a href="#">Configuring Tolerance Policies</a>.</p>

**Step 5** Click **Install**.

----End

## Components

**Table 13-57** Add-on components

Component	Description	Resource Type
cceaddon-nginx-ingress- <Controller name>- controller (The controller name in versions earlier than 2.5.4 is <b>cceaddon-nginx-ingress-controller</b> .)	Nginx Ingress controller, which provides flexible routing and forwarding for clusters	Deployment
cceaddon-nginx-ingress- <Controller name>- backend (The controller name in versions earlier than 2.5.4 is <b>cceaddon-nginx-ingress-default-backend</b> .)	Default backend of Nginx Ingress. The message "default backend - 404" is returned.	Deployment

## Configuring Anti-affinity Between a Workload and Nginx Ingress Controller

To avoid a situation where the node running NGINX Ingress Controller and its containers cannot access the Nginx Ingress Controller, you should set up anti-affinity between the workload and Nginx Ingress Controller. This means that the



workload pods cannot be scheduled to the same node as the Nginx Ingress Controller.

```

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:alpine
          imagePullPolicy: IfNotPresent
          name: nginx
          imagePullSecrets:
            - name: default-secret
      affinity:
        podAntiAffinity:
          requiredDuringSchedulingIgnoredDuringExecution:
            - labelSelector:
                matchExpressions: # Implement anti-affinity through the label of the NGINX Ingress
                  - key: app
                    operator: In
                    values:
                      - nginx-ingress #If multiple NGINX Ingress Controllers are installed in the cluster, the label
value is nginx-ingress-<Controller name>.
            - key: component
              operator: In
              values:
                - controller
      namespaces:
        - kube-system
      topologyKey: kubernetes.io/hostname

```

## Change History

**Table 13-58** Release history for NGINX Ingress Controller 2.6.x

Add-on Version	Supported Cluster Version	New Feature	Community Version
2.6.5	v1.25 v1.27 v1.28 v1.29	Metric collection can be disabled in the startup command.	<a href="#">1.9.6</a>
2.6.4	v1.25 v1.27 v1.28 v1.29	CCE clusters 1.29 are supported.	<a href="#">1.9.6</a>

**Table 13-59** Release history for NGINX Ingress Controller 2.4.x

Add-on Version	Supported Cluster Version	New Feature	Community Version
2.4.6	v1.25 v1.27 v1.28	<ul style="list-style-type: none"> <li>• CCE clusters 1.28 are supported.</li> <li>• Supported admission verification.</li> <li>• Supported graceful shutdown and hitless upgrade.</li> <li>• Supported equivalent distribution of add-on instances in multi-AZ deployment mode.</li> <li>• Fixed the <a href="#">CVE-2023-44487</a> vulnerability.</li> </ul>	<a href="#">1.9.3</a>

**Table 13-60** Release history for NGINX Ingress Controller 2.3.x

Add-on Version	Supported Cluster Version	New Feature	Community Version
2.3.5	v1.27	None	<a href="#">1.8.0</a>

**Table 13-61** Release history for NGINX Ingress Controller 2.2.x

Add-on Version	Supported Cluster Version	New Feature	Community Version
2.2.53	v1.23 v1.25	Fixed some issues.	<a href="#">1.5.1</a>
2.2.42	v1.23 v1.25	<ul style="list-style-type: none"> <li>• Supported graceful shutdown and hitless upgrade.</li> <li>• Supported equivalent distribution of add-on instances in multi-AZ deployment mode.</li> </ul>	<a href="#">1.5.1</a>
2.2.7	v1.25	<ul style="list-style-type: none"> <li>• Synchronized time zones used by the add-on and the node.</li> <li>• Supports IPv4 and IPv6 dual stack.</li> </ul>	<a href="#">1.5.1</a>

Add-on Version	Supported Cluster Version	New Feature	Community Version
2.2.3	v1.25	<ul style="list-style-type: none"> <li>Supported anti-affinity scheduling of add-on pods on nodes in different AZs.</li> <li>Added the tolerance time during which the pods with temporary storage volumes cannot be scheduled.</li> <li>The default taint tolerance duration is changed to 60s.</li> </ul>	<a href="#">1.5.1</a>
2.2.1	v1.25	<ul style="list-style-type: none"> <li>CCE clusters 1.25 are supported.</li> <li>Updated the add-on to its community version v1.5.1.</li> </ul>	<a href="#">1.5.1</a>

**Table 13-62** Release history for NGINX Ingress Controller 2.1.x

Add-on Version	Supported Cluster Version	New Feature	Community Version
2.1.33	v1.19 v1.21 v1.23	<ul style="list-style-type: none"> <li>Supported graceful shutdown and hitless upgrade.</li> <li>Supported equivalent distribution of add-on instances in multi-AZ deployment mode.</li> </ul>	<a href="#">1.2.1</a>
2.1.9	v1.19 v1.21 v1.23	<ul style="list-style-type: none"> <li>Supported anti-affinity scheduling of add-on pods on nodes in different AZs.</li> <li>The default taint tolerance duration is changed to 60s.</li> <li>Synchronized time zones used by the add-on and the node.</li> <li>Supports IPv4 and IPv6 dual stack.</li> </ul>	<a href="#">1.2.1</a>

Add-on Version	Supported Cluster Version	New Feature	Community Version
2.1.5	v1.19 v1.21 v1.23	<ul style="list-style-type: none"> <li>Supported anti-affinity scheduling of add-on pods on nodes in different AZs.</li> <li>The default taint tolerance duration is changed to 60s.</li> </ul>	<a href="#">1.2.1</a>
2.1.3	v1.19 v1.21 v1.23	Enables publishService for nginx-ingress.	<a href="#">1.2.1</a>
2.1.1	v1.19 v1.21 v1.23	Updated the add-on to its community version v1.2.1.	<a href="#">1.2.1</a>
2.1.0	v1.19 v1.21 v1.23	<ul style="list-style-type: none"> <li>Updated the add-on to its community version v1.2.0.</li> <li>Fixed the <a href="#">CVE-2021-25746</a> vulnerability and added <a href="#">rules</a> to disable some annotations values that may cause unauthorized operations.</li> <li>Fixed the <a href="#">CVE-2021-25745</a> vulnerability and added <a href="#">rules</a> to disable some access paths that may cause unauthorized operations.</li> </ul>	<a href="#">1.2.0</a>

**Table 13-63** Release history for NGINX Ingress Controller 2.0.x

Add-on Version	Supported Cluster Version	New Feature	Community Version
2.0.1	v1.19 v1.21 v1.23	<ul style="list-style-type: none"> <li>CCE clusters 1.23 are supported.</li> <li>Updated the add-on to its community version v1.1.1.</li> </ul>	<a href="#">1.1.1</a>

**Table 13-64** Release history for NGINX Ingress Controller 1.3.x

Add-on Version	Supported Cluster Version	New Feature	Community Version
1.3.2	v1.15 v1.17 v1.19 v1.21	<ul style="list-style-type: none"> <li>CCE clusters 1.21 are supported.</li> <li>Updated the add-on to its community version v0.49.3.</li> </ul>	<b>0.49.3</b>

**Table 13-65** Release history for NGINX Ingress Controller 1.2.x

Add-on Version	Supported Cluster Version	New Feature	Community Version
1.2.6	v1.15 v1.17 v1.19	Adds the default seccomp profile.	<b>0.46.0</b>
1.2.5	v1.15 v1.17 v1.19	Updated the add-on to its community version v0.46.0.	<b>0.46.0</b>
1.2.3	v1.15 v1.17 v1.19	CCE clusters 1.19 are supported.	<b>0.43.0</b>
1.2.2	v1.15 v1.17	Updated the add-on to its community version v0.43.0.	<b>0.43.0</b>

## 13.5.3 NodeLocal DNSCache

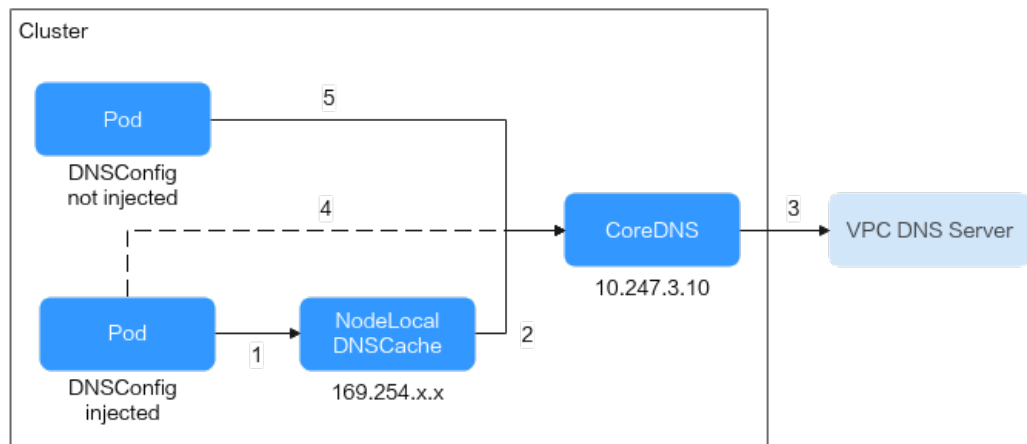
### Introduction

NodeLocal DNSCache is an add-on developed based on the community [NodeLocal DNSCache](#). This add-on functions as a DaemonSet to run the DNS cache proxy on cluster nodes to improve cluster DNS performance.

Open source community: <https://github.com/kubernetes/dns>

After NodeLocal DNSCache is enabled, a DNS query goes through the path as shown below.

**Figure 13-12** NodeLocal DNSCache query path



The resolution lines are described as follows:

- 1. By default, pods that have been injected into the DNS local cache will use the NodeLocal DNSCache to resolve requested domain names.
- 2. If the NodeLocal DNSCache's cache cannot resolve a request, it will ask the cluster's CoreDNS for resolution.
- 3. CoreDNS resolves domain names outside of the cluster by using the DNS server in the VPC.
- 4. If a pod injected into the local DNS cache cannot access the NodeLocal DNSCache, the domain name will be resolved through CoreDNS.
- 5. By default, CoreDNS resolves domain names for pods that are not injected into the local DNS cache.

## Notes and Constraints

- This feature is available only to clusters of v1.19 or later.

## Installing the Add-on

**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 **NodeLocal DNSCache** on the right, and click **Install**.

**Step 2** On the **Install Add-on** page, configure the specifications as needed.

- If you selected **Preset**, you can choose between **Small** or **Large** based on the cluster scale. The system will automatically set the number of add-on pods and resource quotas according to the preset specifications. You can see the configurations on the console.

The small specification specifies that the add-on runs in one pod, which is ideal for clusters with up to 50 nodes and 1000 pods. The large specification specifies that the add-on runs in two pods, which are more appropriate for clusters with a maximum of 500 nodes and 1000 pods.

- If you selected **Custom**, you can adjust the number of pods and resource quotas as needed. High availability is not possible with a single pod. If an error occurs on the node where the add-on instance runs, the add-on will fail.

**Step 3** Configure the add-on parameters.

- **DNSConfig Automatic Injection:** After this function is enabled, a DNSConfig admission controller will be created. The controller intercepts pod creation requests in the namespace labeled with **node-local-dns-injection=enabled** based on admission webhooks and automatically configures DNSConfig for pods. If this function is disabled or the pod belongs to a non-target namespace, you must manually configure DNSConfig for the pod.

**Step 4** Configure deployment policies for the add-on pods.

 **NOTE**

- Scheduling policies do not take effect on add-on instances of the DaemonSet type.
- When configuring multi-AZ deployment or node affinity, ensure that there are nodes meeting the scheduling policy and that resources are sufficient in the cluster. Otherwise, the add-on cannot run.

**Table 13-66** Configurations for add-on scheduling

Parameter	Description
Multi AZ	<ul style="list-style-type: none"> <li>• <b>Preferred:</b> Deployment pods of the add-on will be preferentially scheduled to nodes in different AZs. If all the nodes in the cluster are deployed in the same AZ, the pods will be scheduled to different nodes in that AZ.</li> <li>• <b>Equivalent mode:</b> Deployment pods of the add-on are evenly scheduled to the nodes in the cluster in each AZ. If a new AZ is added, you are advised to increase add-on pods for cross-AZ HA deployment. With the Equivalent multi-AZ deployment, the difference between the number of add-on pods in different AZs will be less than or equal to 1. If resources in one of the AZs are insufficient, pods cannot be scheduled to that AZ.</li> <li>• <b>Required:</b> Deployment pods of the add-on are forcibly scheduled to nodes in different AZs. There can be at most one pod in each AZ. If nodes in a cluster are not in different AZs, some add-on pods cannot run properly. If a node is faulty, add-on pods on it may fail to be migrated.</li> </ul>

Parameter	Description
Node Affinity	<ul style="list-style-type: none"> <li>• <b>Not configured:</b> Node affinity is disabled for the add-on.</li> <li>• <b>Node Affinity:</b> Specify the nodes where the add-on is deployed. If you do not specify the nodes, the add-on will be randomly scheduled based on the default cluster scheduling policy.</li> <li>• <b>Specified Node Pool Scheduling:</b> Specify the node pool where the add-on is deployed. If you do not specify the node pool, the add-on will be randomly scheduled based on the default cluster scheduling policy.</li> <li>• <b>Custom Policies:</b> Enter the labels of the nodes where the add-on is to be deployed for more flexible scheduling policies. If you do not specify node labels, the add-on will be randomly scheduled based on the default cluster scheduling policy. If multiple custom affinity policies are configured, ensure that there are nodes that meet all the affinity policies in the cluster. Otherwise, the add-on cannot run.</li> </ul>
Toleration	<p>Using both taints and tolerations allows (not forcibly) the add-on Deployment to be scheduled to a node with the matching taints, and controls the Deployment eviction policies after the node where the Deployment is located is tainted.</p> <p>The add-on adds the default tolerance policy for the <b>node.kubernetes.io/not-ready</b> and <b>node.kubernetes.io/unreachable</b> taints, respectively. The tolerance time window is 60s.</p> <p>For details, see <a href="#">Configuring Tolerance Policies</a>.</p>

**Step 5** Click **Install**.

----End

## Components

**Table 13-67** Add-on components

Component	Description	Resource Type
node-local-dns-admission-controller	Automatic DNSConfig injecting	Deployment
node-local-dns-cache	DNS cache proxy on nodes to improve the DNS performance of the cluster	DaemonSet



## Using NodeLocal DNSCache

By default, application requests are sent through the CoreDNS proxy. To use NodeLocal DNSCache as the DNS cache proxy, use any of the following methods: (For details, see [Using NodeLocal DNSCache](#).)

- Auto injection: Automatically configure the **dnsConfig** field of the pod when creating the pod. (This function is not available for pods in system namespaces such as kube-system.)
- Manual configuration: Manually configure the **dnsConfig** field of the pod.

## Uninstalling the Add-on

Uninstalling the add-on will affect the pods that have used the node-local-dns address for domain name resolution. Before uninstalling the add-on, delete the **node-local-dns-injection=enabled** label from the involved namespaces, and delete and recreate the pods with the **node-local-dns-webhook.k8s.io/status: injected** label.

### Step 1 Check the 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 **NodeLocal DNSCache** on the right, and click **Edit**.
2. In the **Parameters** area, check whether **DNSConfig Automatic Injection** is enabled.

If **DNSConfig Automatic Injection** has been enabled:

- a. In the navigation pane, choose **Namespaces**.
- b. Locate the rows that contain the namespaces with the **node-local-dns-injection=enabled** label and delete the label. For details, see [Managing Namespace Labels](#).
- c. Delete the pods in these namespaces and recreate pods.

If **DNSConfig Automatic Injection** has not been enabled:

- a. Use kubectl to access the cluster.
- b. Check the pods with DNSConfig manually injected. If multiple namespaces are involved, check all the pods in these namespaces.

For example, to check pods in the **default** namespace, run the following command:

```
kubectl get pod -n default -o yaml
```

- c. Manually remove DNSConfig and recreate pods.

### Step 2 Uninstall NodeLocal DNSCache.

1. In the navigation pane, choose **Add-ons**. Locate **NodeLocal DNSCache** and click **Uninstall**.
2. In the displayed dialog box, click **Yes**.

----End

## Helpful Links

[Using NodeLocal DNSCache to Improve DNS Performance](#)

## Change History

**Table 13-68** Release history

Add-on Version	Supported Cluster Version	New Feature	Community Version
1.6.8	v1.23 v1.25 v1.27 v1.28 v1.29	Optimized custom injection configuration experience.	1.22.20
1.6.7	v1.23 v1.25 v1.27 v1.28 v1.29	Added custom injection configurations.	1.22.20
1.5.0	v1.21 v1.23 v1.25 v1.27 v1.28	<ul style="list-style-type: none"> <li>• CCE clusters 1.28 are supported.</li> <li>• Supported equivalent distribution of add-on instances in multi-AZ deployment mode.</li> <li>• Changed the basic image OS of the add-on pods to Huawei Cloud EulerOS 2.0.</li> </ul>	1.22.20
1.4.0	v1.19 v1.21 v1.23 v1.25 v1.27	Resolved the issue that CCI pods took an excessively long time to access the Internet.	1.22.20
1.3.1	v1.19 v1.21 v1.23 v1.25	<ul style="list-style-type: none"> <li>• Enables automatic DNS Config injection for namespaces.</li> <li>• Synchronized time zones used by the add-on and the node.</li> </ul>	1.22.20
1.2.7	v1.19 v1.21 v1.23 v1.25	Supported anti-affinity scheduling of add-on pods on nodes in different AZs.	1.21.1

Add-on Version	Supported Cluster Version	New Feature	Community Version
1.2.4	v1.19 v1.21 v1.23 v1.25	CCE clusters 1.25 are supported.	1.21.1
1.2.2	v1.19 v1.21 v1.23	Supports customized NodeLocal DNSCache specifications.	1.21.1

## 13.6 Container Storage Add-ons

### 13.6.1 CCE Container Storage (Everest)

#### Introduction

Everest is a cloud native container storage system, which enables clusters of Kubernetes v1.15.6 or later to access cloud storage services through the CSI.

**Everest is a system resource add-on. It is installed by default when a cluster of Kubernetes v1.15 or later is created.**

#### Notes and Constraints

- In version 1.2.0 of the Everest add-on, **key authentication** is optimized when OBS is used. After the Everest add-on is upgraded from a version earlier than 1.2.0, restart all workloads that use OBS in the cluster. Otherwise, workloads may not be able to use OBS.

#### Installing the Add-on

This add-on has been installed by default. If it is uninstalled due to some reasons, you can reinstall it by performing the following steps:

**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 **CCE Container Storage (Everest)** on the right, and click **Install**.

**Step 2** On the **Install Add-on** page, configure the specifications as needed.

- If you selected **Preset**, you can choose between **Small**, **Medium**, or **Large** as needed. The system will automatically set the number of add-on pods and resource quotas according to the preset specifications. You can see the configurations on the console.

The small specification is best for clusters with up to 50 nodes and 500 PVCs. The medium specification works well for clusters with up to 200 nodes and

2000 PVCs. The large specification is perfect for clusters with up to 1000 nodes and 10,000 PVCs.

- If you selected **Custom**, you can adjust the number of pods and resource quotas as needed. The requested CPU and memory can be adjusted based on the number of nodes and PVCs. For details, see [Table 13-69](#).

In non-typical scenarios, the formulas for estimating the limits are as follows:

- everest-csi-controller
  - CPU limit: 250m for 200 or fewer nodes, 350m for 1000 nodes, and 500m for 2000 nodes
  - Memory limit = (200 MiB + Number of nodes x 1 MiB + Number of PVCs x 0.2 MiB) x 1.2
- everest-csi-driver
  - CPU limit: 300m for 200 or fewer nodes, 500m for 1000 nodes, and 800m for 2000 nodes
  - Memory limit: 300 MiB for 200 or fewer nodes, 600 MiB for 1000 nodes, and 900 MiB for 2000 nodes

**Table 13-69** Recommended configuration limits in typical scenarios

Configuration Scenario			everest-csi-controller		everest-csi-driver	
Nodes	PVs/ PVCs	Add-on Pods	CPU Cores (Limit = Request)	Memory (Limit = Request)	CPU Cores (Limit = Request)	Memory (Limit = Request)
50	1000	2	250m	600 MiB	300m	300 MiB
200	1000	2	250m	1 GiB	300m	300 MiB
1000	1000	2	350m	2 GiB	500m	600 MiB
1000	5000	2	450m	3 GiB	500m	600 MiB
2000	5000	2	550m	4 GiB	800m	900 MiB
2000	10000	2	650m	5 GiB	800m	900 MiB

**Step 3** Configure the add-on parameters.

**Table 13-70** Add-on parameters

Item	Configuration Method	Description
Reserved EVS Disks for Non-Container Workloads	Visualized GUI configuration	<p>Number of disks on the node reserved for custom use. This parameter is supported when the add-on version is 2.3.11 or later.</p> <p>Assume that a maximum of 20 EVS disks can be attached to a node, and the value of this parameter is set to <b>6</b>. Then 14 (20-6) disks can be attached to this node when the system schedules the EVS disk attachment workloads. The reserved six disks include one system disk and one data disk that have been attached to the node. You can attach four EVS disks to this node as additional data disks or raw disks for a local storage pool.</p>
Prohibit Global Secret from Mounting Object Storage	Visualized GUI configuration	<p>Whether the default AK/SK can be used when an object bucket or parallel file system is mounted.</p> <ul style="list-style-type: none"> <li>• <b>is</b>: If you do not specify a custom secret for mounting OBS storage, the global secret you uploaded will be used.</li> <li>• <b>No</b>: If you do not specify a custom secret for mounting OBS storage, only the global secret you uploaded can be used. Otherwise, the mounting will fail.</li> </ul>
Concurrent EVS Disk Attaching Tasks	Visualized GUI configuration	<p>Number of workers that can be concurrently processed by Everest for attaching EVS volumes. The default value is <b>60</b>.</p>
Concurrent EVS Disk Detaching Tasks	Visualized GUI configuration	<p>Number of workers that can be concurrently processed by Everest for detaching EVS volumes. The default value is <b>60</b>.</p>
Distributed Volume Mounting	Visualized GUI configuration	<p>When enabled, the everest-csi-driver component on each node is responsible for attaching or detaching EVS disks.</p> <p>If disabled, the everest-csi-controller component takes on this responsibility.</p>

Item	Configuration Method	Description
flow_control	Extended Parameter Settings	The default configuration is used. After installing the CCE Container Storage (Everest) add-on, you can refer to ConfigMap everest-driver-th-config in the kube-system namespace for more details.
over_subscription	Extended parameter settings	Overcommitment ratio of the local storage pool ( <b>local_storage</b> ). The default value for the local storage pool size is set to <b>80</b> . If the pool size is 100 GiB, it can be overcommitted up to 180 GiB. (Total capacity after overcommitment = (1 + Overcommitment ratio) x Actual capacity)
enable_local_autoexpander	Extended parameter settings	Whether to enable automatic scale-out for the local storage pool of thin volumes. If this function is enabled, automatic scale-out is triggered based on the scale-out threshold and scale-out step of the local storage pool.
expansion_threshold	Extended parameter settings	Capacity expansion threshold of the local storage pool of the thin volume. When the usage of the local storage pool of the thin volume exceeds the threshold, the local storage pool is automatically expanded.
expansion_step	Extended parameter settings	Capacity expansion step of a single EVS disk in the local storage pool of thin provisioning volumes (unit: Gi)
expansion_max_evs_size	Extended parameter settings	Maximum capacity of a single EVS disk in the local storage pool of thin provisioning volumes (unit: Gi)

Item	Configuration Method	Description
volume_attaching_flow_ctrl	Extended parameter settings	Maximum number of EVS volumes that can be attached by the Everest add-on within 1 minute. The default value is <b>0</b> , indicating that the performance of attaching EVS volumes is determined by the underlying storage resources.

 **NOTE**

In the extended parameter settings, you can customize the advanced configurations that are not displayed on the GUI. If the settings in the extended parameters conflict with those on the GUI, the settings in the extended parameters will work.

**Step 4** Configure deployment policies for the add-on pods.

 **NOTE**

- Scheduling policies do not take effect on add-on instances of the DaemonSet type.
- When configuring multi-AZ deployment or node affinity, ensure that there are nodes meeting the scheduling policy and that resources are sufficient in the cluster. Otherwise, the add-on cannot run.

**Table 13-71** Configurations for add-on scheduling

Parameter	Description
Multi AZ	<ul style="list-style-type: none"> <li>• <b>Preferred:</b> Deployment pods of the add-on will be preferentially scheduled to nodes in different AZs. If all the nodes in the cluster are deployed in the same AZ, the pods will be scheduled to different nodes in that AZ.</li> <li>• <b>Equivalent mode:</b> Deployment pods of the add-on are evenly scheduled to the nodes in the cluster in each AZ. If a new AZ is added, you are advised to increase add-on pods for cross-AZ HA deployment. With the Equivalent multi-AZ deployment, the difference between the number of add-on pods in different AZs will be less than or equal to 1. If resources in one of the AZs are insufficient, pods cannot be scheduled to that AZ.</li> <li>• <b>Required:</b> Deployment pods of the add-on are forcibly scheduled to nodes in different AZs. There can be at most one pod in each AZ. If nodes in a cluster are not in different AZs, some add-on pods cannot run properly. If a node is faulty, add-on pods on it may fail to be migrated.</li> </ul>

Parameter	Description
Node Affinity	<ul style="list-style-type: none"> <li>• <b>Not configured:</b> Node affinity is disabled for the add-on.</li> <li>• <b>Node Affinity:</b> Specify the nodes where the add-on is deployed. If you do not specify the nodes, the add-on will be randomly scheduled based on the default cluster scheduling policy.</li> <li>• <b>Specified Node Pool Scheduling:</b> Specify the node pool where the add-on is deployed. If you do not specify the node pool, the add-on will be randomly scheduled based on the default cluster scheduling policy.</li> <li>• <b>Custom Policies:</b> Enter the labels of the nodes where the add-on is to be deployed for more flexible scheduling policies. If you do not specify node labels, the add-on will be randomly scheduled based on the default cluster scheduling policy. If multiple custom affinity policies are configured, ensure that there are nodes that meet all the affinity policies in the cluster. Otherwise, the add-on cannot run.</li> </ul>
Toleration	<p>Using both taints and tolerations allows (not forcibly) the add-on Deployment to be scheduled to a node with the matching taints, and controls the Deployment eviction policies after the node where the Deployment is located is tainted.</p> <p>The add-on adds the default tolerance policy for the <b>node.kubernetes.io/not-ready</b> and <b>node.kubernetes.io/unreachable</b> taints, respectively. The tolerance time window is 60s.</p> <p>For details, see <a href="#">Configuring Tolerance Policies</a>.</p>

**Step 5** Click **Install**.

----End

## Components

**Table 13-72** Add-on components

Component	Description	Resource Type
everest-csi-controller	Used to create, delete, snapshot, expand, attach, and detach storage volumes. If the cluster version is 1.19 or later and the add-on version is 1.2.x, the pod of the everest-csi-controller component also has an everest-localvolume-manager container by default. This container manages the creation of LVM storage pools and local PVs on the node.	Deployment



Component	Description	Resource Type
everest-csi-driver	Used to mount and unmount PVs and resize file systems. If the add-on version is 1.2.x and the region where the cluster is located supports node-attacher, the pod of the everest-csi-driver component also contains an everest-node-attacher container. This container is responsible for distributed EVS attaching. This configuration item is available in some regions.	DaemonSet

## Change History

**Table 13-73** Release history

Add-on Version	Supported Cluster Version	New Feature
2.4.28	v1.23 v1.25 v1.27 v1.28 v1.29	Fixed some issues.
2.3.14	v1.21 v1.23 v1.25 v1.27 v1.28	CCE clusters 1.28 are supported.
2.1.51	v1.19 v1.21 v1.23 v1.25 v1.27	Supported Huawei Cloud EulerOS 2.0.
2.1.38	v1.19 v1.21 v1.23 v1.25	Supports association between add-on specifications and cluster specifications.
2.1.30	v1.19 v1.21 v1.23 v1.25	<ul style="list-style-type: none"> <li>Supported anti-affinity scheduling of add-on pods on nodes in different AZs.</li> <li>Adapts the obsfs package to Ubuntu 22.04.</li> </ul>

Add-on Version	Supported Cluster Version	New Feature
2.1.13	v1.19 v1.21 v1.23 v1.25	Optimized the performance of creating subpath PVCs in batches for SFS Turbo volumes.
2.1.9	v1.19 v1.21 v1.23 v1.25	<ul style="list-style-type: none"> <li>Supported graceful exit of the controller.</li> <li>CCE clusters 1.25 are supported.</li> </ul>
2.0.9	v1.19 v1.21 v1.23	<ul style="list-style-type: none"> <li>Rebuilt certain code and architecture of everest to improve its scalability and stability.</li> <li>Enabled graceful exit.</li> <li>Supported OBS process monitoring.</li> </ul>
1.3.28	v1.19 v1.21 v1.23	<ul style="list-style-type: none"> <li>Enabled graceful exit.</li> <li>Supported OBS process monitoring.</li> </ul>
1.3.22	v1.19 v1.21 v1.23	Handled occasional read and write failures after repeated disk mounting.
1.3.20	v1.19 v1.21 v1.23	Handled occasional read and write failures after repeated disk mounting.
1.3.17	v1.19 v1.21 v1.23	<ul style="list-style-type: none"> <li>Updated the rollingUpdates.maxUnavailable of everest-csi-driver from 10 to 10%.</li> <li>Supported user-defined pod anti-affinity rules.</li> <li>Counted the maximum number of SCSI volumes that can be managed by the CSI plug-in on a node.</li> <li>Drivers can be deployed based on customized resource specifications.</li> </ul>
1.3.8	v1.23	CCE clusters 1.23 are supported.
1.3.6	v1.23	CCE clusters 1.23 are supported.

Add-on Version	Supported Cluster Version	New Feature
1.2.78	v1.15 v1.17 v1.19 v1.21	Supported anti-affinity scheduling of add-on pods on nodes in different AZs.
1.2.70	v1.15 v1.17 v1.19 v1.21	Optimized the performance of creating subpath PVCs in batches for SFS Turbo volumes.
1.2.67	v1.15 v1.17 v1.19 v1.21	<ul style="list-style-type: none"> <li>Supported graceful exit of the controller.</li> <li>Supported OBS process monitoring.</li> </ul>
1.2.61	v1.15 v1.17 v1.19 v1.21	<ul style="list-style-type: none"> <li>Enabled graceful exit.</li> <li>Supported OBS process monitoring.</li> </ul>
1.2.55	v1.15 v1.17 v1.19 v1.21	Handled occasional read and write failures after repeated disk mounting.
1.2.53	v1.15 v1.17 v1.19 v1.21	Handled occasional read and write failures after repeated disk mounting.
1.2.51	v1.15 v1.17 v1.19 v1.21	<ul style="list-style-type: none"> <li>Updated the <code>rollingUpdates.maxUnavailable</code> of <code>everest-csi-driver</code> from 10 to 10%.</li> <li>Supported user-defined pod anti-affinity rules.</li> <li>Counted the maximum number of SCSI volumes that can be managed by the CSI plug-in on a node.</li> </ul>
1.2.44	v1.15 v1.17 v1.19 v1.21	<ul style="list-style-type: none"> <li>Enterprise projects can be selected for EVS and OBS volumes.</li> <li>By default, the <code>enable_noobj_cache</code> parameter is no longer used for mounting OBS buckets.</li> </ul>

Add-on Version	Supported Cluster Version	New Feature
1.2.42	v1.15 v1.17 v1.19 v1.21	<ul style="list-style-type: none"> <li>Enterprise projects can be selected for EVS and OBS volumes.</li> <li>By default, the <b>enable_noobj_cache</b> parameter is no longer used for mounting OBS buckets.</li> </ul>
1.2.30	v1.15 v1.17 v1.19 v1.21	Supported emptyDir.
1.2.28	v1.15 v1.17 v1.19 v1.21	CCE clusters 1.21 are supported.
1.2.27	v1.15 v1.17 v1.19 v1.21	Supports ultra-fast SSD (ESSD) and general-purpose SSD (GPSSD) EVS disks.
1.2.13	v1.15 v1.17 v1.19	Supported EulerOS 2.10.
1.2.9	v1.15 v1.17 v1.19	<ul style="list-style-type: none"> <li>Enhances the reliability of PV resource lifecycle maintenance.</li> <li>Attach/Detach Controller can be used to attach or detach volumes in clusters 1.19.10.</li> <li>Improves SFS mounting stability.</li> <li>Changes the default EVS creation type of a new cluster to SAS.</li> </ul>
1.2.5	v1.15 v1.17 v1.19	<ul style="list-style-type: none"> <li>Improves the reliability of mounting-related capabilities.</li> <li>Optimizes the authentication function of using OBS, which requires you to upload the access key.</li> <li>Improves the compatibility of the everest add-on with FlexVolume volumes.</li> <li>Improves running stability of the add-on.</li> </ul>
1.1.12	v1.15 v1.17	Enhances the reliability of the everest-csi-controller component.

Add-on Version	Supported Cluster Version	New Feature
1.1.11	v1.15 v1.17	<ul style="list-style-type: none"><li>• Supports security hardening.</li><li>• Supports third-party OBS storage.</li><li>• Switches to the EVS query API with better performance.</li><li>• Uses snapshots to create disks in clone mode by default.</li><li>• Optimizes and enhances disk status detection and log output for attaching and detaching operations.</li><li>• Improves the reliability of determining authentication expiration.</li></ul>
1.1.8	v1.15 v1.17	Supported CCE 1.17. If CCE 1.13 is upgraded to 1.15, Everest can take over all functions of the FlexVolume driver.
1.1.7	v1.15 v1.17	Supported CCE 1.17. If CCE 1.13 is upgraded to 1.15, Everest can take over all functions of the FlexVolume driver.

## 13.7 Other Add-ons

### 13.7.1 Kubernetes Dashboard

#### Introduction

Kubernetes Dashboard is a general purpose, web-based UI for Kubernetes clusters. It allows users to manage applications running in the cluster and troubleshoot them, as well as manage the cluster itself, by running commands.

With Kubernetes Dashboard, you can:

- Deploy containerized applications to a Kubernetes cluster.
- Diagnose containerized application problems.
- Manage cluster resources.
- View applications running in a cluster.
- Create and modify Kubernetes resources (such as Deployments, jobs, and DaemonSets).
- Check errors that occur in a cluster.

For example, you can scale a Deployment, perform a rolling update, restart a pod, or deploy a new application.

Open source community: <https://github.com/kubernetes/dashboard>

## Installing the Add-on

**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 **Kubernetes Dashboard** on the right, and click **Install**.

**Step 2** In the **Parameters** area, configure the following parameters:

- **Access Mode:** NodePort is supported. This add-on can be accessed through the EIP bound to the node. If the cluster node does not have an EIP bound to it, then this function will not be available.
- **Certificate:** Configure a certificate for the dashboard.
  - Use a custom certification.  
You need to fill in the **Certificate File** and **Certificate Private Key** in PEM format by referring to the example.
  - Use the default certificate.

---

### NOTICE

The default certificate generated by the dashboard is invalid, which affects the normal access to the dashboard through a browser. You are advised to manually upload a valid certificate so that the browser can verify your access and secure your connection.

---

**Step 3** Click **Install**.

----End

## Accessing the dashboard Add-on

**Step 1** Log in to the CCE console and click the cluster name to access the cluster console. Choose **Add-ons** in the navigation pane. On the page displayed, verify that the dashboard add-on is in the **Running** state and click **Access**.

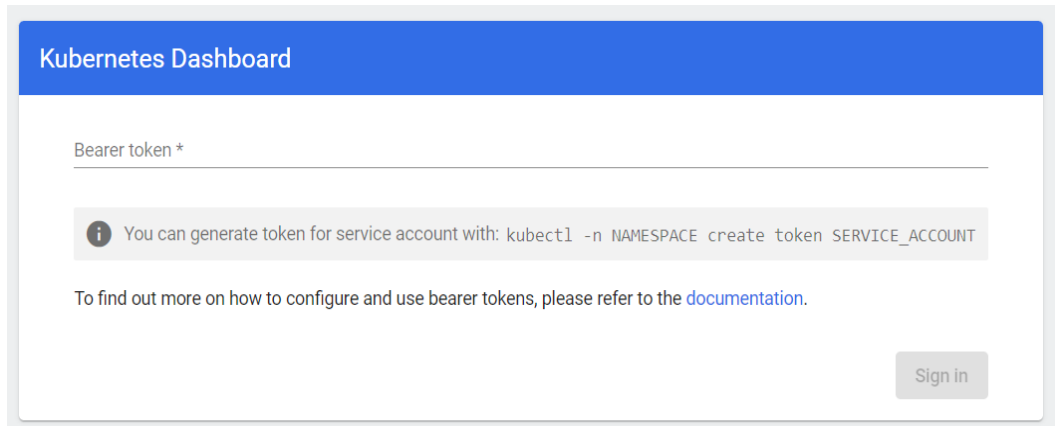
**Step 2** Copy the token in the dialog box displayed.

**Step 3** On the dashboard login page, select **Token**, paste the copied token, and click **SIGN IN**.

### NOTE

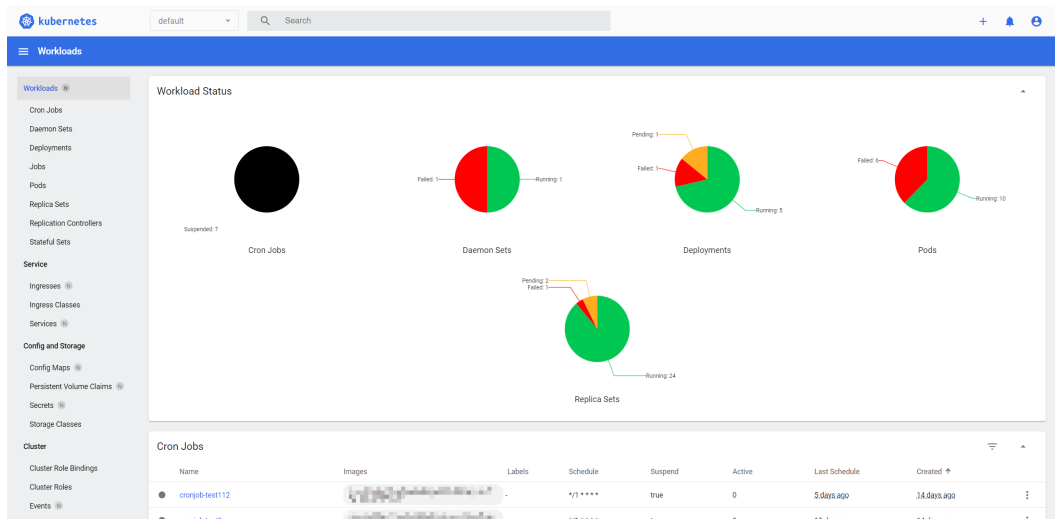
By default, this add-on does not support login using kubeconfig authenticated by certificate. You are advised to use the token mode for login. For details, see <https://github.com/kubernetes/dashboard/issues/2474#issuecomment-348912376>.

**Figure 13-13** Token login



**Step 4** View the dashboard page as shown in **Figure 13-14**.

**Figure 13-14** Dashboard page



----End

## Modifying Permissions

After the dashboard is installed, the initial role can only view a majority of resources that are displayed on the dashboard. To apply for the permissions to perform other operations on the dashboard, modify RBAC authorization resources in the background.

### Procedure

Modify the **kubernetes-dashboard-minimal** rule in the ClusterRole.

For details about how to use RBAC authorization, visit <https://kubernetes.io/docs/reference/access-authn-authz/rbac/>.

## Components

**Table 13-74** Add-on components

Component	Description	Resource Type
Dashboard	Visualized monitoring UI	Deployment

## Troubleshooting Access Problems

When Google Chrome is used to access the dashboard, the error message "ERR\_CERT\_INVALID", instead of the login page, is displayed. The possible cause is that the default certificate generated by the dashboard does not pass Google Chrome verification. There are two solutions to this problem:

**Figure 13-15** Error message displayed on Google Chrome



### Your connection is not private

Attackers might be trying to steal your information from **www.illaskme.com** (for example, passwords, messages, or credit cards). [Learn more](#)

NET::ERR\_CERT\_COMMON\_NAME\_INVALID

Help improve Safe Browsing by sending some [system information and page content](#) to Google. [Privacy policy](#)

Advanced

Back to safety

- **Solution 1:** Use the Firefox browser to access the dashboard. In the **Exceptions** area of the **Proxy Settings** page, add the dashboard address to the addresses that will bypass the proxy server. Then, the dashboard login page will be displayed.
- **Solution 2:** Start Google Chrome with the **--ignore-certificate-errors** flag to ignore the certificate error.

**Windows:** Save the dashboard address. Close all active Google Chrome windows. Press the Windows key + R to display the **Run** dialog box. Enter **chrome --ignore-certificate-errors** in the **Run** dialog box to open a new Google Chrome window. In the address bar, enter the dashboard address to open the login page.



## Change History

**Table 13-75** Release history

Add-on Version	Supported Cluster Version	New Feature	Community Version
3.0.2	v1.27 v1.28 v1.29	<ul style="list-style-type: none"> <li>Supported clusters of v1.27, v1.28, and v1.29.</li> <li>Updated the add-on to its community version 7.3.2.</li> </ul>	<b>7.3.2</b>
2.2.27	v1.21 v1.23 v1.25	Fixed some issues.	<b>2.7.0</b>
2.2.7	v1.21 v1.23 v1.25	None	<b>2.7.0</b>
2.2.5	v1.21 v1.23 v1.25	Synchronized time zones used by the add-on and the node.	<b>2.7.0</b>
2.2.3	v1.21 v1.23 v1.25	None	<b>2.7.0</b>
2.1.1	v1.19 v1.21 v1.23	<ul style="list-style-type: none"> <li>CCE clusters 1.23 are supported.</li> <li>Updated the add-on to its community version v2.5.0.</li> </ul>	<b>2.5.0</b>
2.0.10	v1.15 v1.17 v1.19 v1.21	CCE clusters 1.21 are supported.	<b>2.0.0</b>
2.0.4	v1.15 v1.17 v1.19	Adds the default seccomp profile.	<b>2.0.0</b>
2.0.3	v1.15 v1.17 v1.19	CCE clusters 1.15 are supported.	<b>2.0.0</b>
2.0.2	v1.17 v1.19	CCE clusters 1.19 are supported.	<b>2.0.0</b>

Add-on Version	Supported Cluster Version	New Feature	Community Version
2.0.1	v1.15 v1.17		<b>2.0.0</b>
2.0.0	v1.17	Enables interconnection with CCE v1.17	<b>2.0.0</b>

# 14 Helm Chart

---

## 14.1 Overview of a Chart

CCE provides a console for managing Helm charts, helping you easily deploy applications using the charts and manage applications on the console.

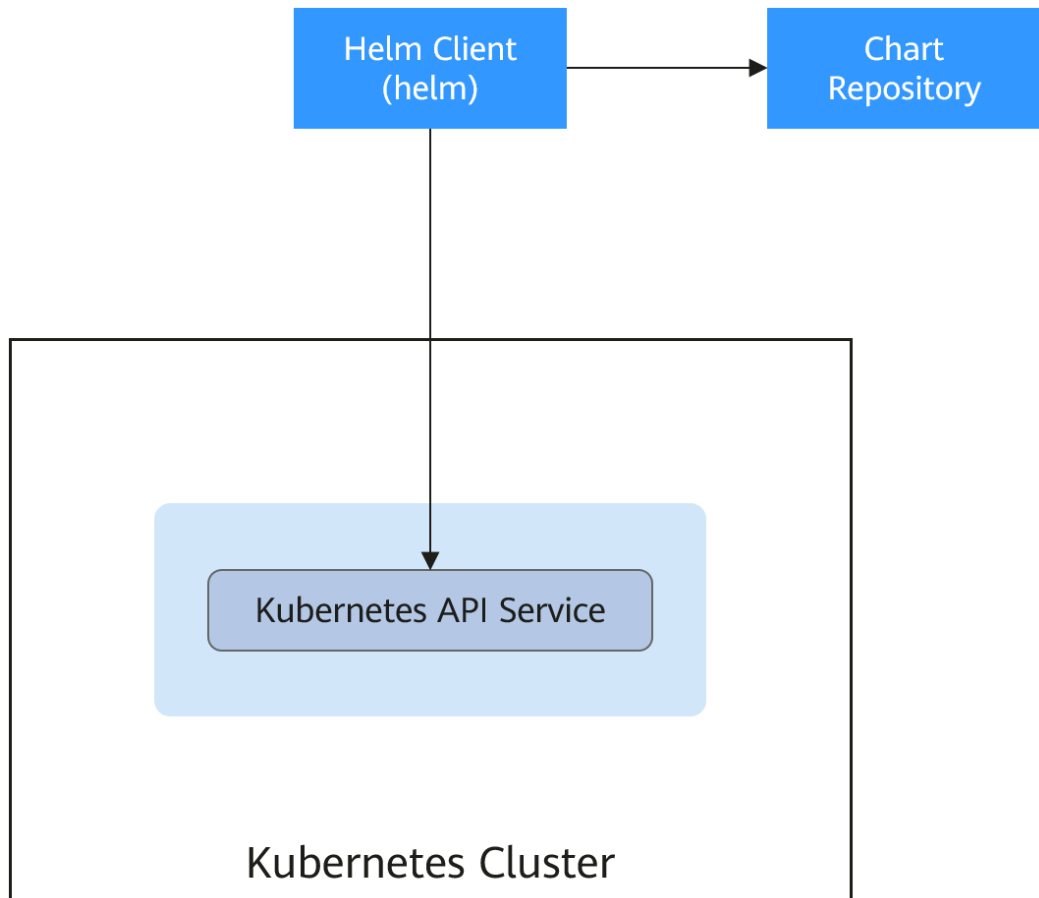
### Helm

**Helm** is a package manager for Kubernetes and manages charts. A Helm chart is a series of YAML files used to encapsulate native Kubernetes applications. When deploying an application, you can customize some metadata of the application for easy application distribution. Application releasors can use Helm to package applications, manage application dependencies and application versions, and release applications to the software repository. After using Helm, users do not need to compile complex application deployment files. They can easily search for, install, upgrade, roll back, and uninstall applications on Kubernetes.

The relationship between Helm and Kubernetes is as follows:

- Helm <-> Kubernetes
- Apt <-> Ubuntu
- Yum <-> CentOS
- Pip <-> Python

The following figure shows the solution architecture:



Helm can help application orchestration for Kubernetes:

- Manages, edits, and updates a large number of Kubernetes configuration files.
- Deploys a complex Kubernetes application that contains a large number of configuration files.
- Shares and reuses Kubernetes configurations and applications.
- Supports multiple environments with parameter-based configuration templates.
- Manages the release of applications, including rolling back the application, finding differences (using the **diff** command), and viewing the release history.
- Controls phases in a deployment cycle.
- Tests and verifies the released version.

## 14.2 Deploying an Application from a Chart

On the CCE console, you can upload a Helm chart package, deploy it, and manage the deployed pods.

### Notes and Constraints

- The number of charts that can be uploaded by a single user is limited. The value displayed on the console of each region is the allowed quantity.

- A chart with multiple versions consumes the same amount of portion of chart quota.
- Users with chart operation permissions can perform multiple operations on clusters. Therefore, exercise caution when assigning users the chart lifecycle management permissions, including uploading charts and creating, deleting, and updating chart releases.

## Chart Specifications

The Redis workload is used as an example to illustrate the chart specifications.

- **Naming Requirement**

A chart package is named in the format of **{name}-{version}.tgz**, where **{version}** indicates the version number in the format of *Major version number.Minor version number.Revision number*, for example, **redis-0.4.2.tgz**.

 **NOTE**

The chart name {name} can contain a maximum of 64 characters.

The version number must comply with the [semantic versioning](#) rules.

- The main and minor version numbers are mandatory, and the revision number is optional.
- The major and minor version numbers and revision number must be integers, greater than or equal to 0, and less than or equal to 99.


- **Directory Structure**

The directory structure of a chart is as follows:

```
redis/
  templates/
  values.yaml
  README.md
  Chart.yaml
  .helmignore
```

As listed in [Table 14-1](#), the parameters marked with \* are mandatory.

**Table 14-1** Parameters in the directory structure of a chart

Parameter	Description
* templates	Stores all templates.
* values.yaml	<p>Describes configuration parameters required by templates.</p> <p><b>NOTICE</b></p> <p>Make sure that the image address set in the <b>values.yaml</b> file is the same as the image address in the container image repository. Otherwise, an exception occurs when you create a workload, and the system displays a message indicating that the image fails to be pulled.</p> <p>To obtain the image address, perform the following operations: Log in to the CCE console. In the navigation pane, choose <b>Image Repository</b> to access the SWR console. Choose <b>My Images &gt; Private Images</b> and click the name of the uploaded image. On the <b>Image Tags</b> tab page, obtain the image address from the pull command. You can click  to copy the command in the <b>Image Pull Command</b> column.</p>

Parameter	Description
README.md	A markdown file, including: <ul style="list-style-type: none"> <li>The workload or services provided by the chart.</li> <li>Prerequisites for running the chart.</li> <li>Configurations in the <b>values.yaml</b> file.</li> <li>Information about chart installation and configuration.</li> </ul>
* Chart.yaml	Basic information about the chart. Note: The API version of Helm v3 is switched from v1 to v2.
.helmignore	Files or data that does not need to read templates during workload installation.

## Uploading a Chart

- Step 1** Log in to the CCE console and click the cluster name to access the cluster console. Choose **App Templates** in the navigation pane and click **Upload Chart** in the upper right corner.
- Step 2** Click **Select File**, select the chart to be uploaded, and click **Upload**.

 **NOTE**

When you upload a chart, the naming rule of the OBS bucket is changed from `cce-charts-{region}-{domain_name}` to `cce-charts-{region}-{domain_id}`. In the old naming rule, the system converts the `domain_name` value into a Base64 string and uses the first 63 characters. If you cannot find the chart in the OBS bucket with the new name, search for the bucket with the old name.

----End

## Creating a Release

- Step 1** Log in to the CCE console and click the cluster name to access the cluster console. In the navigation pane, choose **App Templates**.
- Step 2** On the **My Charts** tab page, click **Install** of the target chart.
- Step 3** Set workload installation parameters by referring to [Table 14-2](#).

**Table 14-2** Installation parameters

Parameter	Description
Instance	Unique name of the chart release.
Namespace	Namespace to which the workload will be deployed.
Select Version	Version of a chart.

Parameter	Description
Configuration File	<p>You can import and replace the <b>values.yaml</b> file or directly edit the chart parameters online.</p> <p><b>NOTE</b></p> <p>An imported <b>values.yaml</b> file must comply with YAML specifications, that is, KEY:VALUE format. The fields in the file are not restricted.</p> <p>The key value of the imported values.yaml must be the same as that of the selected chart package. Otherwise, the values.yaml does not take effect. That is, the key cannot be changed.</p> <ol style="list-style-type: none"> <li>1. Click <b>Select File</b>.</li> <li>2. Select the corresponding <b>values.yaml</b> file and click <b>Open</b>.</li> </ol>

**Step 4** Click **Install**.

On the **Releases** tab page, you can view the installation status of the release.

----End

## Upgrading a Chart-based Workload

**Step 1** Log in to the CCE console and click the cluster name to access the cluster console. Choose **App Templates** in the navigation pane. In the right pane, click the **Releases** tab.

**Step 2** Click **Upgrade** in the row where the desired workload resides and set the parameters for the workload.

**Step 3** Select a chart version for **Chart Version**.

**Step 4** Follow the prompts to modify the chart parameters. Confirm the modification and click **Upgrade**.

**Step 5** If the execution status is **Upgraded**, the workload has been upgraded.

----End

## Rolling Back a Chart-based Workload

**Step 1** Log in to the CCE console and click the cluster name to access the cluster console. Choose **App Templates** in the navigation pane. In the right pane, click the **Releases** tab.

**Step 2** Click **More > Roll Back** for the workload to be rolled back, select the workload version, and click **Roll back to this version**.

In the workload list, if the status is **Rollback successful**, the workload is rolled back successfully.

----End

## Uninstalling a Chart-based Workload

- Step 1** Log in to the CCE console and click the cluster name to access the cluster console. Choose **App Templates** in the navigation pane. In the right pane, click the **Releases** tab.
- Step 2** Click **More > Uninstall** next to the release to be uninstalled, and click **Yes**. Exercise caution when performing this operation because releases cannot be restored after being uninstalled.

----End

## 14.3 Differences Between Helm v2 and Helm v3 and Adaptation Solutions

Helm v2 stops at version 2.17.0. Currently, Helm v3 is the standard in the Helm community. You are advised to switch your charts to **Helm v3 format** as soon as possible.

Changes since Helm v2:

### 1. Removal of Tiller

Helm v3 is simpler and easier to use. It removes tiller and directly connects to the API server using kubeconfig, simplifying the security model.

### 2. Improved upgrade strategy: 3-way strategic merge patches

Helm v2 used a two-way strategic merge patch. During an upgrade, it compared the most recent chart's manifest against the proposed chart's manifest to determine what changes needed to be applied to the resources in Kubernetes. If changes were applied to the cluster out-of-band (such as during a kubectl edit), those changes were not considered. This resulted in resources being unable to roll back to its previous state.

Helm v3 uses a three-way strategic merge patch. Helm considers the original manifest, its live state, and the new manifest when generating a patch. Helm compares the current live state with the live state of the original manifest, checks whether the new manifest is modified, and automatically supplements the new manifest to generate the final update patch.

For details and examples, see [https://v3.helm.sh/docs/faq/changes\\_since\\_helm2](https://v3.helm.sh/docs/faq/changes_since_helm2).

### 3. Secrets as the default storage driver

Helm v2 used ConfigMaps by default to store release information. In Helm v3, Secrets are now used as the default storage driver.

### 4. Release names are now scoped to the namespace

In Helm v2, the information about each release was stored in the same namespace as Tiller. In practice, this meant that once a name was used by a release, no other release could use that same name, even if it was deployed in a different namespace. In Helm v3, information about a particular release is now stored in the same namespace as the release itself. This means that the release name can be used in different namespaces. The namespace of the application is the same as that of the release.



#### 5. Verification mode change

Helm v3 verifies the chart format more strictly. For example, Helm v3 bumps the `apiVersion` in `Chart.yaml` from v1 to v2. For the `Chart.yaml` of v2, `apiVersion` must be set to v1. After installing the Helm v3 client, you can run the `helm lint` command to check whether the chart format complies with the Helm v3 specifications.

**Adaptation solution:** Adapt the Helm v3 chart based on the Helm official document <https://helm.sh/docs/topics/charts/>. The `apiVersion` field is mandatory.

#### 6. Removal of the `crd-install` hook

The `crd-install` hook has been removed in favor of the `crds/` directory in Helm v3. Note that the resources in the `crds/` directory are deployed only during the release installation and are not updated during the upgrade. When the resources are deleted, the resources are retained in the `crds/` directory. If the CRD already exists, it will be skipped with a warning during the repeated installation.

**Adaptation solution:** According to the [Helm document](#), you can hold your CRD in the `crds/` directory or a separate chart. Helm cannot upgrade or delete the CRD. Therefore, you are advised to put the CRD in one chart, and then put any resources that use that CRD in another chart.

#### 7. Resources that are not created using Helm are not forcibly updated. Releases are not forcibly upgraded by default.

The forcible upgrade logic of Helm v3 is changed. After the upgrade fails, the system does not delete and rebuild the Helm v3. Instead, the system directly uses the `put` logic. Therefore, the CCE release upgrade uses the non-forcible update logic by default. Resources that cannot be updated through patches will make the release unable to be upgraded. If a release with the same name exists in the environment and does not have the home tag `app.kubernetes.io/managed-by: Helm` of Helm v3, a conflict message is displayed.

**Adaptation solution:** Delete related resources and create them using Helm.

#### 8. Limit on release historical records

Only the latest 10 release versions are retained by default.

For more changes and details, see [Helm official documents](#).

- Differences between Helm v2 and Helm v3: [https://v3.helm.sh/docs/faq/changes\\_since\\_helm2](https://v3.helm.sh/docs/faq/changes_since_helm2)
- How to migrate from Helm v2 to Helm v3: [https://helm.sh/docs/topics/v2\\_v3\\_migration](https://helm.sh/docs/topics/v2_v3_migration)

## 14.4 Deploying an Application Through the Helm v2 Client

### Prerequisites

The Kubernetes cluster created on CCE has been connected to `kubectl`. For details, see [Using kubectl](#).

## Installing Helm v2

This section uses Helm v2.17.0 as an example.

For other versions, visit <https://github.com/helm/helm/releases>.

**Step 1** Download the Helm client from the VM connected to the cluster.

```
wget https://get.helm.sh/helm-v2.17.0-linux-amd64.tar.gz
```

**Step 2** Decompress the Helm package.

```
tar -xzf helm-v2.17.0-linux-amd64.tar.gz
```

**Step 3** Copy Helm to the system path, for example, `/usr/local/bin/helm`.

```
mv linux-amd64/helm /usr/local/bin/helm
```

**Step 4** RBAC is enabled on the Kubernetes API server. Create the service account name **tiller** for the tiller and assign `cluster-admin`, a system ClusterRole, to the tiller. Create a tiller resource account as follows:

### vim tiller-rbac.yaml

```
apiVersion: v1
kind: ServiceAccount
metadata:
  name: tiller
  namespace: kube-system
---
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  name: tiller
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: cluster-admin
subjects:
- kind: ServiceAccount
  name: tiller
  namespace: kube-system
```

**Step 5** Deploy the tiller resource account.

```
kubectl apply -f tiller-rbac.yaml
```

**Step 6** Initialize the Helm and deploy the pod of tiller.

```
helm init --service-account tiller --skip-refresh
```

**Step 7** Query the status.

```
kubectl get pod -n kube-system -l app=helm
```

Command output:

```
NAME                READY STATUS  RESTARTS  AGE
tiller-deploy-7b56c8dfb7-fxk5g  1/1   Running  1         23h
```

**Step 8** Query the Helm version.

```
# helm version
Client: &version.Version{SemVer:"v2.17.0", GitCommit:"a690bad98af45b015bd3da1a41f6218b1a451dbe", GitTreeState:"clean"}
Server: &version.Version{SemVer:"v2.17.0", GitCommit:"a690bad98af45b015bd3da1a41f6218b1a451dbe", GitTreeState:"clean"}
```

----End

## Installing the Helm Chart

If the charts provided by CCE do not meet requirements, download a chart and install it.

You can obtain the required chart in the **stable** directory on this [website](#), download the chart, and upload it to the node.

1. Download and decompress the obtained chart. Generally, the chart is in ZIP format.  
`unzip chart.zip`
2. Install the Helm chart.  
`helm install aerospike/`
3. After the installation is complete, run the **helm list** command to check the status of the chart releases.

## Common Issues

- The following error message is displayed after the **Helm version** command is run:

```
Client:
&version.Version{SemVer:"v2.17.0",
GitCommit:"a690bad98af45b015bd3da1a41f6218b1a451d8e", GitTreeState:"clean"}
E0718 11:46:10.132102 7023 portforward.go:332] an error occurred
forwarding 41458 -> 44134: error forwarding port 44134 to pod
d566b78f997eea6c4b1c0322b34ce8052c6c2001e8edff243647748464cd7919, uid : unable
to do port forwarding: socat not found.
Error: cannot connect to Tiller
```

The preceding information is displayed because the socat is not installed. Run the following command to install the socat:

```
yum install socat -y
```

- When you run the **yum install socat -y** command on a node running EulerOS 2.9 or Huawei Cloud EulerOS, the following error message is displayed:

```
No match for argument: socat
Error: Unable to find a match: socat
```

The image does not contain socat. In this case, manually download the RPM chart and run the following command to install it (replace the RPM chart name with the actual one):

```
rpm -i socat-1.7.3.2-8.oe1.x86_64.rpm
```

**Table 14-3** Addresses for downloading socat RPM charts

OS	Where to Download
EulerOS 2.9	<ul style="list-style-type: none"> <li>• <a href="#">x86</a></li> <li>• <a href="#">ARM</a></li> </ul>
Huawei Cloud EulerOS 2.0	<ul style="list-style-type: none"> <li>• <a href="#">x86</a></li> <li>• <a href="#">Arm</a></li> </ul>
Huawei Cloud EulerOS 1.1	<a href="#">x86</a>

- When the socat has been installed and the following error message is displayed after the **helm version** command is run:

```
test@local:~/k8s/helm/test$ helm version
Client: &version.Version{SemVer:"v3.3.0",
GitCommit:"021cb0ac1a1b2f888144ef5a67b8dab6c2d45be6", GitTreeState:"clean"}
```

The Helm chart reads the configuration certificate from the **.Kube/config** file to communicate with Kubernetes. The preceding error indicates that the kubectl configuration is incorrect. In this case, reconnect the cluster to kubectl. For details, see [Using kubectl](#).

- Storage fails to be created after you have connected to cloud storage services. This issue may be caused by the **annotation** field in the created PVC. Change the chart name and install the chart again.

- If kubectl is not properly configured, the following error message is displayed after the **helm install** command is run:

```
[root@prometheus-57046 ~]# helm install prometheus/ --generate-name
WARNING: This chart is deprecated
Error: Kubernetes cluster unreachable: Get "http://localhost:8080/version?timeout=32s": dial tcp
[::1]:8080: connect: connection refused
```

**Solution:** Configure kubeconfig for the node. For details, see [Using kubectl](#).

## 14.5 Deploying an Application Through the Helm v3 Client

### Prerequisites

- The Kubernetes cluster created on CCE has been connected to kubectl. For details, see [Using kubectl](#).
- To pull a public image when deploying Helm, ensure an EIP has been bound to the node.

### Installing Helm v3

This section uses Helm v3.3.0 as an example.

For other versions, visit <https://github.com/helm/helm/releases>.

**Step 1** Download the Helm client from the VM connected to the cluster.

```
wget https://get.helm.sh/helm-v3.3.0-linux-amd64.tar.gz
```

**Step 2** Decompress the Helm package.

```
tar -xzf helm-v3.3.0-linux-amd64.tar.gz
```

**Step 3** Copy Helm to the system path, for example, **/usr/local/bin/helm**.

```
mv linux-amd64/helm /usr/local/bin/helm
```

**Step 4** Query the Helm version.

```
helm version
version.BuildInfo{Version:"v3.3.0", GitCommit:"e29ce2a54e96cd02ccfce88bee4f58bb6e2a28b6",
GitTreeState:"clean", GoVersion:"go1.13.4"}
```

----End

## Installing the Helm Chart

You can use Helm to install a chart. Before using Helm, you may need to understand the following concepts to better use Helm:

- **Chart:** contains resource definitions and a large number of configuration files of Kubernetes applications.
- **Repository:** stores shared charts. You can download charts from the repository to a local path for installation or install them online.
- **Release:** running result of after a chart is installed in a Kubernetes cluster using Helm. A chart can be installed multiple times in a cluster. A new release will be created for each installation. A MySQL chart is used as an example. To run two databases in a cluster, install the chart twice. Each database has its own release and release name.

For more details, see [Using Helm](#).

**Step 1** Search for a chart from the [Artifact Hub](#) repository recommended by Helm and configure the Helm repository.

```
helm repo add {repo_name} {repo_addr}
```

The following uses the [WordPress chart](#) as an example:

```
helm repo add bitnami https://charts.bitnami.com/bitnami
```

**Step 2** Run the **helm install** command to install the chart.

```
helm install {release_name} {chart_name} --set key1=val1
```

For example, to install WordPress, the WordPress chart added in [Step 1](#) is **bitnami/wordpress**, the release name is **my-wordpress**, and mandatory parameters have been configured.

```
helm install my-wordpress bitnami/wordpress \  
  --set mariadb.primary.persistence.enabled=true \  
  --set mariadb.primary.persistence.storageClass=csi-disk \  
  --set mariadb.primary.persistence.size=10Gi \  
  --set persistence.enabled=false
```

Run the **helm show values {chart\_name}** command to view the configurable options of the chart. For example, to view the configurable items of WordPress, run the following command:

```
helm show values bitnami/wordpress
```

**Step 3** View the installed chart release.

```
helm list
```

----End

## Common Issues

- The following error message is displayed after the **helm version** command is run:  
Client:  
&version.Version{SemVer:"v3.3.0",  
GitCommit:"012cb0ac1a1b2f888144ef5a67b8dab6c2d45be6", GitTreeState:"clean"}  
E0718 11:46:10.132102 7023 portforward.go:332] an error occurred  
forwarding 41458 -> 44134: error forwarding port 44134 to pod  
d566b78f997eea6c4b1c0322b34ce8052c6c2001e8edff243647748464cd7919, uid : unable  
to do port forwarding: socat not found.  
Error: cannot connect to Tiller

The preceding information is displayed because the socat is not installed. Run the following command to install the socat:

```
yum install socat -y
```

- When you run the **yum install socat -y** command on a node running EulerOS 2.9 or Huawei Cloud EulerOS, the following error message is displayed:

```
No match for argument: socat
Error: Unable to find a match: socat
```

The node image does not contain socat. In this case, manually download the RPM chart and run the following command to install it (replace the RPM chart name with the actual one):

```
rpm -i socat-1.7.3.2-8.oe1.x86_64.rpm
```

**Table 14-4** Addresses for downloading socat RPM charts

OS	Where to Download
EulerOS 2.9	<ul style="list-style-type: none"> <li>• <a href="#">x86</a></li> <li>• <a href="#">Arm</a></li> </ul>
Huawei Cloud EulerOS 2.0	<ul style="list-style-type: none"> <li>• <a href="#">x86</a></li> <li>• <a href="#">Arm</a></li> </ul>
Huawei Cloud EulerOS 1.1	<a href="#">x86</a>

- When the socat has been installed and the following error message is displayed after the **helm version** command is run:

```
$ helm version
Client: &version.Version{SemVer:"v3.3.0",
GitCommit:"021cb0ac1a1b2f888144ef5a67b8dab6c2d45be6", GitTreeState:"clean"}
Error: cannot connect to Tiller
```

The Helm chart reads the configuration certificate in **.Kube/config** to communicate with Kubernetes. The preceding error indicates that the kubectl configuration is incorrect. In this case, reconnect the cluster to kubectl. For details, see [Using kubectl](#).

- Storage fails to be created after you have connected to cloud storage services. This issue may be caused by the **annotation** field in the created PVC. Change the chart name and install the chart again.
- If kubectl is not properly configured, the following error message is displayed after the **helm install** command is run:

```
# helm install prometheus/ --generate-name
WARNING: This chart is deprecated
Error: Kubernetes cluster unreachable: Get "http://localhost:8080/version?timeout=32s": dial tcp [::1]:8080: connect: connection refused
```

**Solution:** Configure kubeconfig for the node. For details, see [Using kubectl](#).

## 14.6 Converting a Release from Helm v2 to v3

### Context

CCE fully supports Helm v3. This section guides you to convert a Helm v2 release to Helm v3. Helm v3 discards or reconstructs some Helm v2 functions at the

bottom layer. Therefore, the conversion is risky to some extent. Simulation is required before conversion.

For details, see the [community documentation](#).

## Precautions

- Helm v2 stores release information in ConfigMaps. Helm v3 does so in secrets.
- When you query, update, or operate a Helm v2 release on the CCE console, CCE will attempt to convert the release to v3. If you operate in the background, convert the release by following the instructions below.

## Conversion Process (Without Using the Helm v3 Client)

**Step 1** Download the helm 2-to-3 conversion plugin on the CCE node.

```
wget https://github.com/helm/helm-2to3/releases/download/v0.10.2/helm-2to3_0.10.2_linux_amd64.tar.gz
```

**Step 2** Decompress the plugin package.

```
tar -xvzf helm-2to3_0.10.2_linux_amd64.tar.gz
```

**Step 3** Perform the simulated conversion.

Take the test-convert release as an example. Run the following command to simulate the conversion: If the following information is displayed, the simulation is successful.

```
# ./2to3 convert --dry-run --tiller-out-cluster -s configmaps test-convert
NOTE: This is in dry-run mode, the following actions will not be executed.
Run without --dry-run to take the actions described below:
Release "test-convert" will be converted from Helm v2 to Helm v3.
[Helm 3] Release "test-convert" will be created.
[Helm 3] ReleaseVersion "test-convert.v1" will be created.
```

**Step 4** Perform the conversion. If the following information is displayed, the conversion is successful.

```
# ./2to3 convert --tiller-out-cluster -s configmaps test-convert
Release "test-convert" will be converted from Helm v2 to Helm v3.
[Helm 3] Release "test-convert" will be created.
[Helm 3] ReleaseVersion "test-convert.v1" will be created.
[Helm 3] ReleaseVersion "test-convert.v1" created.
[Helm 3] Release "test-convert" created.
Release "test-convert" was converted successfully from Helm v2 to Helm v3.
Note: The v2 release information still remains and should be removed to avoid conflicts with the migrated v3 release.
v2 release information should only be removed using `helm 2to3` cleanup and when all releases have been migrated over.
```

**Step 5** After the conversion is complete, simulate the resource clearance. After the simulation, clear the v2 release resources.

Simulated clearance:

```
# ./2to3 cleanup --dry-run --tiller-out-cluster -s configmaps --name test-convert
NOTE: This is in dry-run mode, the following actions will not be executed.
Run without --dry-run to take the actions described below:
WARNING: "Release 'test-convert' Data" will be removed.

[Cleanup/confirm] Are you sure you want to cleanup Helm v2 data? [y/N]: y
Helm v2 data will be cleaned up.
[Helm 2] Release 'test-convert' will be deleted.
[Helm 2] ReleaseVersion "test-convert.v1" will be deleted.
```

Formal clearance:

```
# ./2to3 cleanup --tiller-out-cluster -s configmaps --name test-convert
WARNING: "Release 'test-convert' Data" will be removed.

[Cleanup/confirm] Are you sure you want to cleanup Helm v2 data? [y/N]: y
Helm v2 data will be cleaned up.
[Helm 2] Release 'test-convert' will be deleted.
[Helm 2] ReleaseVersion "test-convert.v1" will be deleted.
[Helm 2] ReleaseVersion "test-convert.v1" d
```

----End

## Conversion Process (Using the Helm v3 Client)

**Step 1** Install the Helm v3 client. For details, see [Installing Helm v3](#).

**Step 2** Install the conversion plugin.

```
# helm plugin install https://github.com/helm/helm-2to3
Downloading and installing helm-2to3 v0.10.2 ...
https://github.com/helm/helm-2to3/releases/download/v0.10.2/helm-2to3_0.10.2_linux_amd64.tar.gz
Installed plugin: 2to3
```

**Step 3** Check whether the plugin has been installed.

```
# helm plugin list
NAME VERSION DESCRIPTION
2to3 0.10.2 migrate and cleanup Helm v2 configuration and releases in-place to Helm v3
```

**Step 4** Perform the simulated conversion.

Take the test-convert release as an example. Run the following command to simulate the conversion: If the following information is displayed, the simulated conversion is successful.

```
# helm 2to3 convert --dry-run --tiller-out-cluster -s configmaps test-convert
NOTE: This is in dry-run mode, the following actions will not be executed.
Run without --dry-run to take the actions described below:
Release "test-convert" will be converted from Helm v2 to Helm v3.
[Helm 3] Release "test-convert" will be created.
[Helm 3] ReleaseVersion "test-convert.v1" will be created.
```

**Step 5** Perform the conversion. If the following information is displayed, the conversion is successful.

```
# helm 2to3 convert --tiller-out-cluster -s configmaps test-convert
Release "test-convert" will be converted from Helm v2 to Helm v3.
[Helm 3] Release "test-convert" will be created.
[Helm 3] ReleaseVersion "test-convert.v1" will be created.
[Helm 3] ReleaseVersion "test-convert.v1" created.
[Helm 3] Release "test-convert" created.
Release "test-convert" was converted successfully from Helm v2 to Helm v3.
Note: The v2 release information still remains and should be removed to avoid conflicts with the migrated v3 release.
v2 release information should only be removed using `helm 2to3` cleanup and when all releases have been migrated over.
```

**Step 6** After the conversion, you can view the converted release by running **helm list**.

```
# helm list
NAME                NAMESPACE  REVISION UPDATED                               STATUS  CHART          APP
VERSION
test-convert        default     1          2022-08-29 06:56:28.166918487 +0000 UTC  deployed test-
helmold-1
```

**Step 7** After the conversion is complete, simulate the resource clearance. After the simulation, clear the v2 release resources.

Simulated clearance:



```
# helm 2to3 cleanup --dry-run --tiller-out-cluster -s configmaps --name test-convert
NOTE: This is in dry-run mode, the following actions will not be executed.
Run without --dry-run to take the actions described below:
WARNING: "Release 'test-convert' Data" will be removed.

[Cleanup/confirm] Are you sure you want to cleanup Helm v2 data? [y/N]: y
Helm v2 data will be cleaned up.
[Helm 2] Release 'test-convert' will be deleted.
[Helm 2] ReleaseVersion "test-convert.v1" will be deleted.
```

**Formal clearance:**

```
# helm 2to3 cleanup --tiller-out-cluster -s configmaps --name test-convert
WARNING: "Release 'test-convert' Data" will be removed.

[Cleanup/confirm] Are you sure you want to cleanup Helm v2 data? [y/N]: y
Helm v2 data will be cleaned up.
[Helm 2] Release 'test-convert' will be deleted.
[Helm 2] ReleaseVersion "test-convert.v1" will be deleted.
[Helm 2] ReleaseVersion "test-convert.v1" deleted.
[Helm 2] Release 'test-convert' deleted.
Helm v2 data was cleaned up successfully.
```

**----End**

# 15 Permissions

---

## 15.1 Permissions Overview

CCE permissions management allows you to assign permissions to IAM users and user groups under your tenant accounts. CCE combines the advantages of Identity and Access Management (IAM) and Kubernetes Role-based Access Control (RBAC) authorization to provide a variety of authorization methods, including IAM fine-grained authorization, IAM token authorization, cluster-scoped authorization, and namespace-wide authorization.

CCE allows you to manage permissions on clusters and related resources at a finer granularity, for example, to control the access of employees in different departments to cloud resources.

This section describes the CCE permissions management mechanism and related concepts. If your account has met your service requirements, you can skip this section.

### CCE Permissions Management

CCE permissions are described as follows:

- **Cluster-level permissions:** Cluster-level permissions management evolves out of the system policy authorization feature of IAM. IAM users in the same user group have the same permissions. On IAM, you can configure system policies to describe which IAM user groups can perform which operations on cluster resources. For example, you can grant user group A to create and delete cluster X, add a node, or install an add-on, while granting user group B to view information about cluster X.

Cluster-level permissions involve non-Kubernetes APIs in CCE clusters and support fine-grained IAM policies.

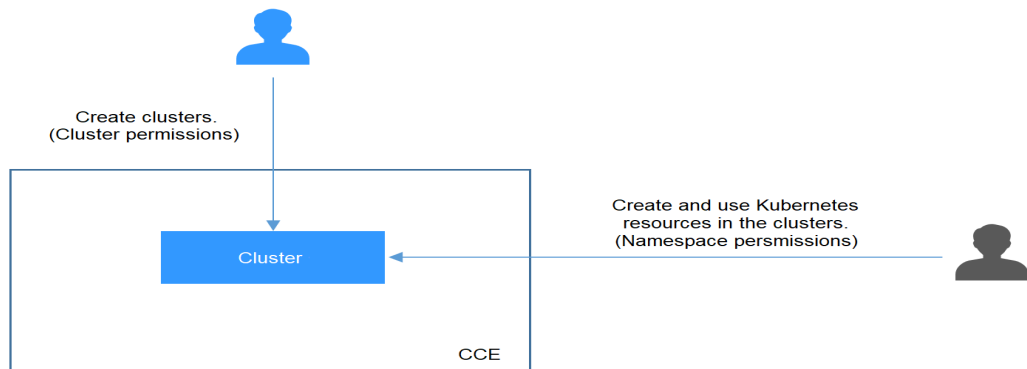
- **Namespace-level permissions:** You can regulate users' or user groups' access to Kubernetes resources in a single namespace based on their Kubernetes RBAC roles. CCE has also been enhanced based on open-source capabilities. It supports RBAC authorization based on IAM user or user group, and RBAC authentication on access to APIs using IAM tokens.

Namespace-level permissions involve CCE Kubernetes APIs and are enhanced based on the Kubernetes RBAC capabilities. Namespace-level permissions can

be granted to IAM users or user groups for authentication and authorization, but are independent of fine-grained IAM policies.

In general, you configure CCE permissions in two scenarios. The first is creating and managing clusters and related resources, such as nodes. The second is creating and using Kubernetes resources in the cluster, such as workloads and Services.

**Figure 15-1** Illustration on CCE permissions



These permissions allow you to manage resource users at a finer granularity.

## Cluster Permissions (IAM-based) and Namespace Permissions (Kubernetes RBAC-based)

Users with different cluster permissions (assigned using IAM) have different namespace permissions (assigned using Kubernetes RBAC). [Table 15-1](#) lists the namespace permissions of different users.

**Table 15-1** Differences in namespace permissions

User	Clusters of v1.13 and Later
User with the Tenant Administrator permissions	All namespace permissions
IAM user with the CCE Administrator role	All namespace permissions
IAM user with the CCE FullAccess or CCE ReadOnlyAccess role	Requires Kubernetes RBAC authorization.
IAM user with the Tenant Guest role	Requires Kubernetes RBAC authorization.

## kubectl Permissions

You can use [kubectl](#) to access Kubernetes resources in a cluster.

When you access a cluster using kubectl, CCE uses the kubeconfig.json file generated on the cluster for authentication. This file contains user information, based on which CCE determines which Kubernetes resources can be accessed by

kubectl. The permissions recorded in a kubeconfig.json file vary from user to user. The permissions that a user has are listed in [Table 15-1](#).

## 15.2 Granting Cluster Permissions to an IAM User

CCE cluster-level permissions are assigned based on **IAM system policies** and **custom policies**. You can use user groups to assign permissions to IAM users.

---

### CAUTION

- Cluster permissions are granted to users for operating cluster-related resources only (such as clusters and nodes). To operate Kubernetes resources like workloads and Services, you must be granted the [namespace permissions](#) as well.
  - When viewing a cluster on the CCE console, the information displayed depends on the namespace permissions. If you have no namespace permissions, you cannot view the resources in the cluster. For details, see [Permission Dependency of the CCE Console](#).
- 

### Prerequisites

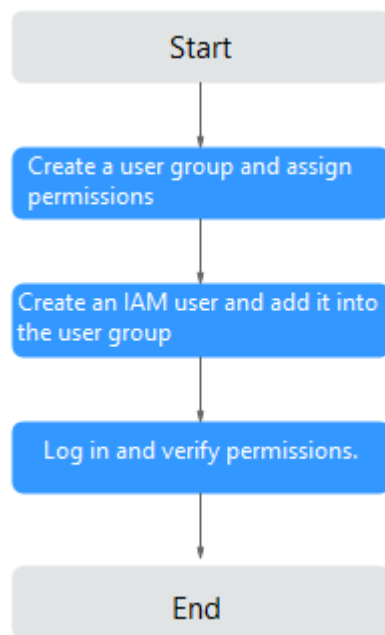
- A user with the Security Administrator role (for example, your account) has all IAM permissions except role switching. Only these users can view user groups and their permissions on the **Permissions** page on the CCE console.

### Configuration

On the CCE console, when you choose **Permissions > Cluster-Level Permissions** to create a user group, you will be directed to the IAM console to complete the process. After the user group is created and its permissions are configured, you can view the information on the **Cluster-Level Permissions** tab page. This section describes the operations in IAM.

## Process Flow

**Figure 15-2** Process of assigning CCE permissions



1. **Create a user group and assign permissions** to it.

Create a user group on the IAM console, and assign CCE permissions, for example, the **CCE ReadOnlyAccess** policy to the group.

 **NOTE**

CCE is deployed by region. On the IAM console, select **Region-specific projects** when assigning CCE permissions.

2. **Create a user and add it to a user group.**

Create a user on the IAM console and add the user to the group created in **1**.

---

**NOTICE**

IAM users need programmatic and management console access to use CCE.

---

3. **Log in** and verify permissions.

Log in to the management console as the user you created, and verify that the user has the assigned permissions.

- Log in to the management console, switch to the CCE console, and buy a cluster. If you fail to do so (assuming that only the **CCE ReadOnlyAccess** permission is assigned), the **CCE ReadOnlyAccess** policy has already taken effect.
- Switch to the console of any other service. If a message appears indicating that you do not have the required permissions for accessing the service, the **CCE ReadOnlyAccess** policy has already taken effect.

## System-defined Roles

Roles are a type of coarse-grained authorization mechanism that defines service-level permissions based on user responsibilities. Only a limited number of service-level roles are available for authorization. Roles are not ideal for fine-grained authorization and least privilege access.

The preset system role for CCE in IAM is **CCE Administrator**. When assigning this role to a user group, you must also select other roles and policies on which this role depends, such as **Tenant Guest**, **Server Administrator**, **ELB Administrator**, **OBS Administrator**, **SFS Administrator**, **SWR Admin**, and **APM FullAccess**.

## System-defined Policies

The system policies preset for CCE in IAM are **CCE FullAccess** and **CCE ReadOnlyAccess**.

- **CCE FullAccess**: common operation permissions on CCE cluster resources, excluding the namespace-level permissions for the clusters (with Kubernetes RBAC enabled) and the privileged administrator operations, such as agency configuration and cluster certificate generation
- **CCE ReadOnlyAccess**: permissions to view CCE cluster resources, excluding the namespace-level permissions of the clusters (with Kubernetes RBAC enabled)

### NOTE

When purchasing a cluster or node that is billed on a yearly/monthly basis, add **custom policies** and configure payment permissions such as **bss:\*:\*** for the Billing Center.

## Custom Policies

Custom policies can be created as a supplement to the system-defined policies of CCE.

You can create custom policies in either of the following ways:

- Visual editor: Select cloud services, actions, resources, and request conditions. This does not require knowledge of policy syntax.
- JSON: Edit JSON policies from scratch or based on an existing policy.

This section provides examples of common custom CCE policies.

### Example Custom Policies:

- Example 1: Creating a cluster named **test**

```
{
  "Version": "1.1",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "cce:cluster:create"
      ]
    }
  ]
}
```

- Example 2: Denying node deletion

A policy with only "Deny" permissions must be used with other policies. If the permissions assigned to a user contain both "Allow" and "Deny", the "Deny" permissions take precedence over the "Allow" permissions.

The following method can be used if you need to assign permissions of the **CCEFullAccess** policy to a user but you want to prevent the user from deleting nodes (**cce:node:delete**). Create a custom policy for denying node deletion, and attach both policies to the group to which the user belongs. Then, the user can perform all operations on CCE except deleting nodes. The following is an example of a deny policy:

```
{
  "Version": "1.1",
  "Statement": [
    {
      "Effect": "Deny",
      "Action": [
        "cce:node:delete"
      ]
    }
  ]
}
```

- Example 3: Defining permissions for multiple services in a policy

A custom policy can contain the actions of multiple services that are of the global or project-level type. The following is an example policy containing actions of multiple services:

```
{
  "Version": "1.1",
  "Statement": [
    {
      "Action": [
        "ecs:cloudServers:resize",
        "ecs:cloudServers:delete",
        "ecs:cloudServers:delete",
        "ims:images:list",
        "ims:serverImages:create"
      ],
      "Effect": "Allow"
    }
  ]
}
```

## CCE Cluster Permissions and Enterprise Projects

CCE supports resource management and permission allocation by cluster and enterprise project.

Note that:

- IAM projects are based on physical isolation of resources, whereas enterprise projects provide global logical groups of resources, which better meet the actual requirements of enterprises. In addition, IAM policies can be managed based on enterprise projects. Therefore, use enterprise projects for permissions management.
- When there are both IAM projects and enterprise projects, IAM preferentially matches the IAM project policies.
- When creating a cluster or node using purchased cloud resources, ensure that IAM users have been granted the required permissions in the enterprise project to use these resources. Otherwise, the cluster or node may fail to be created.

- If a resource does not support enterprise projects, the permissions granted to the resource will not take effect.

Resource Type	Resource Name	Description
Supporting enterprise projects	cluster	Cluster
	node	Node
	nodepool	Node pool
	job	Job
	tag	Cluster label
	addonInstance	Add-on instance
	release	Helm release
	storage	Storage
Not supporting enterprise projects	quota	Cluster quota
	chart	Chart
	addonTemplate	Add-on template

## CCE Cluster Permissions and IAM RBAC

CCE is compatible with IAM system roles for permissions management. Use fine-grained policies provided by IAM to simplify permissions management.

CCE supports the following roles:

- Basic IAM roles:
  - `te_admin` (Tenant Administrator): Users with this role can call all APIs of all services except IAM.
  - `readonly` (Tenant Guest): Users with this role can call APIs with the read-only permissions of all services except IAM.
- Custom CCE administrator role: CCE Administrator

### NOTE

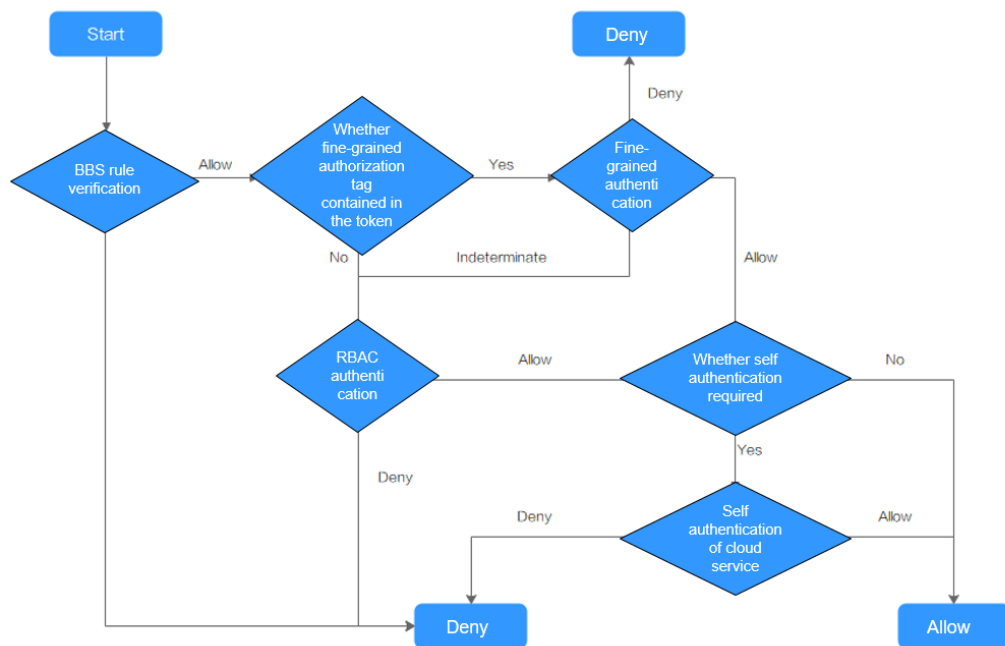
If a user has the Tenant Administrator or CCE Administrator system role, the user has the cluster-admin permissions in Kubernetes RBAC and the permissions cannot be removed after the cluster is created.

If the user is the cluster creator, the cluster-admin permissions in Kubernetes RBAC are granted to the user by default. The permissions can be manually removed after the cluster is created.

- Method 1: Choose **Permissions Management > Namespace-Level Permissions > Delete** in the same role as cluster-creator on the CCE console.
- Method 2: Delete **ClusterRoleBinding: cluster-creator** through the API or kubectl.

When RBAC and IAM policies co-exist, the backend authentication logic for open APIs or console operations on CCE is as follows.





## 15.3 Namespace Permissions (Kubernetes RBAC-based)

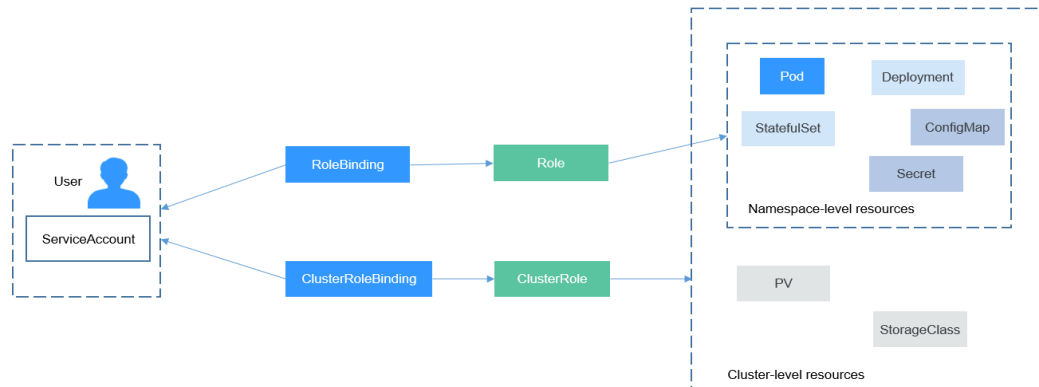
### Namespace Permissions (Kubernetes RBAC-based)

You can regulate users' or user groups' access to Kubernetes resources in a single namespace based on their Kubernetes RBAC roles. The RBAC API declares four kinds of Kubernetes objects: Role, ClusterRole, RoleBinding, and ClusterRoleBinding, which are described as follows:

- Role: defines a set of rules for accessing Kubernetes resources in a namespace.
- RoleBinding: defines the relationship between users and roles.
- ClusterRole: defines a set of rules for accessing Kubernetes resources in a cluster (including all namespaces).
- ClusterRoleBinding: defines the relationship between users and cluster roles.

Role and ClusterRole specify actions that can be performed on specific resources. RoleBinding and ClusterRoleBinding bind roles to specific users, user groups, or ServiceAccounts. Illustration:

**Figure 15-3** Role binding



On the CCE console, you can assign permissions to a user or user group to access resources in one or multiple namespaces. By default, the CCE console provides the following ClusterRoles:

- view (read-only): read-only permission on most resources in all or selected namespaces.
- edit (development): read and write permissions on most resources in all or selected namespaces. If this ClusterRole is configured for all namespaces, its capability is the same as the O&M permission.
- admin (O&M): read and write permissions on most resources in all namespaces, and read-only permission on nodes, storage volumes, namespaces, and quota management.
- cluster-admin (administrator): read and write permissions on all resources in all namespaces.
- geip-editor-role: use of global EIPs in the cluster, but only read and write permissions on GEIPs.
- icagent-clusterRole: report Kubernetes event logs to LTS.

In addition to the preceding typical ClusterRoles, you can define Role and RoleBinding to grant the permissions to add, delete, modify, and obtain global resources (such as nodes, PVs, and CustomResourceDefinitions) and different resources (such as pods, Deployments, and Services) in namespaces for refined permission control.

## Cluster Permissions (IAM-based) and Namespace Permissions (Kubernetes RBAC-based)

Users with different cluster permissions (assigned using IAM) have different namespace permissions (assigned using Kubernetes RBAC). [Table 15-2](#) lists the namespace permissions of different users.

**Table 15-2** Differences in namespace permissions

User	Clusters of v1.13 and Later
User with the Tenant Administrator permissions	All namespace permissions

User	Clusters of v1.13 and Later
IAM user with the CCE Administrator role	All namespace permissions
IAM user with the CCE FullAccess or CCE ReadOnlyAccess role	Requires Kubernetes RBAC authorization.
IAM user with the Tenant Guest role	Requires Kubernetes RBAC authorization.

## Precautions

- After you create a cluster, CCE automatically assigns the cluster-admin permission to you, which means you have full control on all resources in all namespaces in the cluster. The ID of a federated user changes upon each login and logout. Therefore, the user with the permissions is displayed as deleted. In this case, do not delete the permissions. Otherwise, the authentication fails. You are advised to grant the cluster-admin permission to a user group on CCE and add federated users to the user group.
- A user with the Security Administrator role has all IAM permissions except role switching. For example, an account in the admin user group has this role by default. Only these users can assign permissions on the **Permissions** page on the CCE console.

## Configuring Namespace Permissions (on the Console)

You can regulate users' or user groups' access to Kubernetes resources in a single namespace based on their Kubernetes RBAC roles.

- Step 1** Log in to the CCE console. In the navigation pane, choose **Permissions**.
- Step 2** Select a cluster for which you want to add permissions from the drop-down list on the right.
- Step 3** Click **Add Permissions** in the upper right corner.
- Step 4** Confirm the cluster name and select the namespace to assign permissions for. For example, select **All namespaces**, the target user or user group, and select the permissions.

### NOTE

If you do not have IAM permissions, you cannot select users or user groups when configuring permissions for other users or user groups. In this case, you can enter a user ID or user group ID.

Permissions can be customized as required. After selecting **Custom** for **Permission Type**, click **Add Custom Role** on the right of the **Custom** parameter. In the dialog box displayed, enter a name and select a rule. After the custom rule is created, you can select a value from the **Custom** drop-down list box.

Custom permissions 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](#).

- A ClusterRole is a cluster-level resource that can be used to configure cluster access permissions.
- A Role is used to configure access permissions in a namespace. When creating a Role, specify the namespace to which the Role belongs.

**Step 5** Click **OK**.

----End

## Using kubectl to Configure Namespace Permissions

### NOTE

When you access a cluster using kubectl, CCE uses **kubeconfig.json** generated on the cluster for authentication. This file contains user information, based on which CCE determines which Kubernetes resources can be accessed by kubectl. The permissions recorded in a kubeconfig.json file vary from user to user. The permissions that a user has are listed in [Cluster Permissions \(IAM-based\)](#) and [Namespace Permissions \(Kubernetes RBAC-based\)](#).

In addition to cluster-admin, admin, edit, and view, you can define Roles and RoleBindings to configure the permissions to add, delete, modify, and obtain resources, such as pods, Deployments, and Services, in the namespace.

The procedure for creating a Role is very simple. To be specific, specify a namespace and then define rules. The rules in the following example are to allow GET and LIST operations on pods in the default namespace.

```
kind: Role
apiVersion: rbac.authorization.k8s.io/v1
metadata:
  namespace: default          # Namespace
  name: role-example
rules:
- apiGroups: [""]
  resources: ["pods"]         # The pod can be accessed.
  verbs: ["get", "list"]     # The GET and LIST operations can be performed.
```

- **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](#).

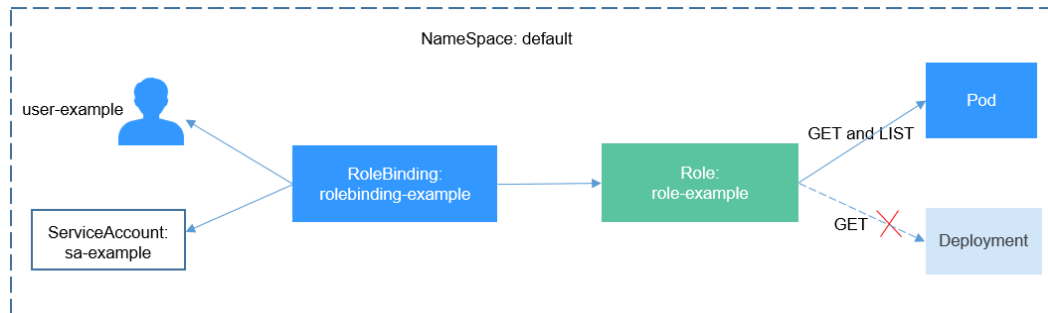
After creating a Role, you can bind the Role to a specific user, which is called RoleBinding. The following shows an example:

```
kind: RoleBinding
apiVersion: rbac.authorization.k8s.io/v1
metadata:
  name: RoleBinding-example
  namespace: default
  annotations:
    CCE.com/IAM: 'true'
roleRef:
  kind: Role
  name: role-example
  apiGroup: rbac.authorization.k8s.io
subjects:
```

```
- kind: User
name: 0c97ac3cb280f4d91fa7c0096739e1f8 # User ID of the user-example
apiGroup: rbac.authorization.k8s.io
```

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.

**Figure 15-4** Binding a role to a user



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
```

Use the IAM user `user-example` to connect to the cluster and obtain the pod information. The following is an example of the returned pod information.

```
# kubectl get pod
NAME                                READY STATUS RESTARTS AGE
deployment-389584-2-6f6bd4c574-2n9rk 1/1   Running 0      4d7h
deployment-389584-2-6f6bd4c574-7s5qw 1/1   Running 0      4d7h
deployment-3895841-746b97b455-86g77 1/1   Running 0      4d7h
deployment-3895841-746b97b455-twvnpn 1/1   Running 0      4d7h
nginx-658dff48ff-7rkph                1/1   Running 0      4d9h
nginx-658dff48ff-njdhj                1/1   Running 0      4d9h
# 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 that **user-example** does not have the required permissions. Try querying the pods in namespace `kube-system`. The output shows that **user-example** does not have the required permissions. 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"
```

## Example: Assigning Cluster Administrator Permissions (cluster-admin)

You can use the cluster-admin role to assign all permissions on a cluster. This role contains the permissions for all cluster resources.

In the following example kubectl output, a ClusterRoleBinding has been created and the cluster-admin role is bound to the user group **cce-role-group**.

```
# kubectl get clusterrolebinding
NAME                                     ROLE                                     AGE
clusterrole_cluster-admin_group0c96fad22880f32a3f84c009862af6f7 ClusterRole/cluster-admin 61s

# kubectl get clusterrolebinding clusterrole_cluster-admin_group0c96fad22880f32a3f84c009862af6f7 -oyaml
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  annotations:
    CCE.com/IAM: "true"
  creationTimestamp: "2021-06-23T09:15:22Z"
  name: clusterrole_cluster-admin_group0c96fad22880f32a3f84c009862af6f7
  resourceVersion: "36659058"
  selfLink: /apis/rbac.authorization.k8s.io/v1/clusterrolebindings/clusterrole_cluster-admin_group0c96fad22880f32a3f84c009862af6f7
  uid: d6cd43e9-b4ca-4b56-bc52-e36346fc1320
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: cluster-admin
subjects:
- apiGroup: rbac.authorization.k8s.io
  kind: Group
  name: 0c96fad22880f32a3f84c009862af6f7
```

Connect to the cluster as an authorized user. If the PVs and StorageClasses can be queried, the permission configuration takes effect.

```
# kubectl get pv
No resources found
# kubectl get sc
NAME          PROVISIONER          RECLAIMPOLICY  VOLUMEBINDINGMODE
ALLOWVOLUMEEXPANSION  AGE
csi-disk      everest-csi-provisioner  Delete         Immediate         true             75d
csi-disk-topology everest-csi-provisioner  Delete         WaitForFirstConsumer true             75d
csi-nas       everest-csi-provisioner  Delete         Immediate         true             75d
csi-obs       everest-csi-provisioner  Delete         Immediate         false            75d
```

## Example: Assigning Namespace O&M Permissions (admin)

The admin role has the read and write permissions on most namespace resources. You can grant the admin permission on all namespaces to a user or user group.

In the following example kubectl output, a RoleBinding has been created and the admin role is bound to the user group **cce-role-group**.

```
# kubectl get rolebinding
NAME                                     ROLE                                     AGE
clusterrole_admin_group0c96fad22880f32a3f84c009862af6f7 ClusterRole/admin 18s
# kubectl get rolebinding clusterrole_admin_group0c96fad22880f32a3f84c009862af6f7 -oyaml
apiVersion: rbac.authorization.k8s.io/v1
kind: RoleBinding
metadata:
  annotations:
    CCE.com/IAM: "true"
  creationTimestamp: "2021-06-24T01:30:08Z"
  name: clusterrole_admin_group0c96fad22880f32a3f84c009862af6f7
  resourceVersion: "36963685"
```

```
selfLink: /apis/rbac.authorization.k8s.io/v1/namespaces/default/rolebindings/
clusterrole_admin_group0c96fad22880f32a3f84c009862af6f7
uid: 6c6f46a6-8584-47da-83f5-9eef1f7b75d6
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: admin
subjects:
- apiGroup: rbac.authorization.k8s.io
  kind: Group
  name: 0c96fad22880f32a3f84c009862af6f7
```

Connect to the cluster as an authorized user. If the PVs and StorageClasses can be queried but a namespace cannot be created, the permission configuration takes effect.

```
# kubectl get pv
No resources found
# kubectl get sc
NAME             PROVISIONER          RECLAIMPOLICY  VOLUMEBINDINGMODE
ALLOWVOLUMEEXPANSION  AGE
csi-disk         everest-csi-provisioner  Delete         Immediate         true            75d
csi-disk-topology everest-csi-provisioner  Delete         WaitForFirstConsumer true            75d
csi-nas          everest-csi-provisioner  Delete         Immediate         true            75d
csi-obs          everest-csi-provisioner  Delete         Immediate         false           75d
# kubectl apply -f namespaces.yaml
Error from server (Forbidden): namespaces is forbidden: User "0c97ac3cb280f4d91fa7c0096739e1f8" cannot
create resource "namespaces" in API group "" at the cluster scope
```

## Example: Assigning Namespace Developer Permissions (edit)

The edit role has the read and write permissions on most namespace resources. You can grant the edit permission on all namespaces to a user or user group.

In the following example kubectl output, a RoleBinding has been created, the edit role is bound to the user group **cce-role-group**, and the target namespace is the default namespace.

```
# kubectl get rolebinding
NAME             ROLE             AGE
clusterrole_admin_group0c96fad22880f32a3f84c009862af6f7  ClusterRole/admin 18s
# kubectl get rolebinding clusterrole_admin_group0c96fad22880f32a3f84c009862af6f7 -oyaml
apiVersion: rbac.authorization.k8s.io/v1
kind: RoleBinding
metadata:
  annotations:
    CCE.com/IAM: "true"
  creationTimestamp: "2021-06-24T01:30:08Z"
  name: clusterrole_admin_group0c96fad22880f32a3f84c009862af6f7
  namespace: default
  resourceVersion: "36963685"
  selfLink: /apis/rbac.authorization.k8s.io/v1/namespaces/default/rolebindings/
clusterrole_admin_group0c96fad22880f32a3f84c009862af6f7
uid: 6c6f46a6-8584-47da-83f5-9eef1f7b75d6
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: edit
subjects:
- apiGroup: rbac.authorization.k8s.io
  kind: Group
  name: 0c96fad22880f32a3f84c009862af6f7
```

Connect to the cluster as an authorized user. In this example, you can create and obtain resources in the default namespace, but cannot query resources in the kube-system namespace or cluster resources.

```
# kubectl get pod
NAME                READY STATUS RESTARTS AGE
test-568d96f4f8-brdrp 1/1   Running 0       33m
test-568d96f4f8-cgjqp 1/1   Running 0       33m
# kubectl get pod -nkube-system
Error from server (Forbidden): pods is forbidden: User "0c97ac3cb280f4d91fa7c0096739e1f8" cannot list resource "pods" in API group "" in the namespace "kube-system"
# kubectl get pv
Error from server (Forbidden): persistentvolumes is forbidden: User "0c97ac3cb280f4d91fa7c0096739e1f8" cannot list resource "persistentvolumes" in API group "" at the cluster scope
```

## Example: Assigning Read-Only Namespace Permissions (view)

The view role has the read-only permissions on a namespace. You can assign permissions to users to view one or multiple namespaces.

In the following example kubectl output, a RoleBinding has been created, the view role is bound to the user group **cce-role-group**, and the target namespace is the default namespace.

```
# kubectl get rolebinding
NAME                                ROLE          AGE
clusterrole_view_group0c96fad22880f32a3f84c009862af6f7 ClusterRole/view 7s

# kubectl get rolebinding clusterrole_view_group0c96fad22880f32a3f84c009862af6f7 -oyaml
apiVersion: rbac.authorization.k8s.io/v1
kind: RoleBinding
metadata:
  annotations:
    CCE.com/IAM: "true"
  creationTimestamp: "2021-06-24T01:36:53Z"
  name: clusterrole_view_group0c96fad22880f32a3f84c009862af6f7
  namespace: default
  resourceVersion: "36965800"
  selfLink: /apis/rbac.authorization.k8s.io/v1/namespaces/default/rolebindings/clusterrole_view_group0c96fad22880f32a3f84c009862af6f7
  uid: b86e2507-e735-494c-be55-c41a0c4ef0dd
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: view
subjects:
- apiGroup: rbac.authorization.k8s.io
  kind: Group
  name: 0c96fad22880f32a3f84c009862af6f7
```

Connect to the cluster as an authorized user. In this example, you can query resources in the default namespace but cannot create resources.

```
# kubectl get pod
NAME                READY STATUS RESTARTS AGE
test-568d96f4f8-brdrp 1/1   Running 0       40m
test-568d96f4f8-cgjqp 1/1   Running 0       40m
# kubectl run -i --tty --image tutum/dnsutils dnsutils --restart=Never --rm /bin/sh
Error from server (Forbidden): pods is forbidden: User "0c97ac3cb280f4d91fa7c0096739e1f8" cannot create resource "pods" in API group "" in the namespace "default"
```

## Example: Assigning Permissions for a Specific Kubernetes Resource Object

You can assign permissions on a specific Kubernetes resource object, such as pod, Deployment, and Service. For details, see [Using kubectl to Configure Namespace Permissions](#).



## 15.4 Example: Designing and Configuring Permissions for Users in a Department

### Overview

The conventional distributed task scheduling mode is being replaced by Kubernetes. CCE allows you to easily deploy, manage, and scale containerized applications in the cloud by providing support for you to use Kubernetes.

To help enterprise administrators manage resource permissions in clusters, CCE provides multi-dimensional, fine-grained permission policies and management measures. CCE permissions are described as follows:

- **Cluster-level permissions:** allowing a user group to perform operations on clusters, nodes, node pools, charts, and add-ons. These permissions are assigned based on IAM system policies.
- **Namespace-level permissions:** allowing a user or user group to perform operations on Kubernetes resources, such as workloads, networking, storage, and namespaces. These permissions are assigned based on Kubernetes RBAC.

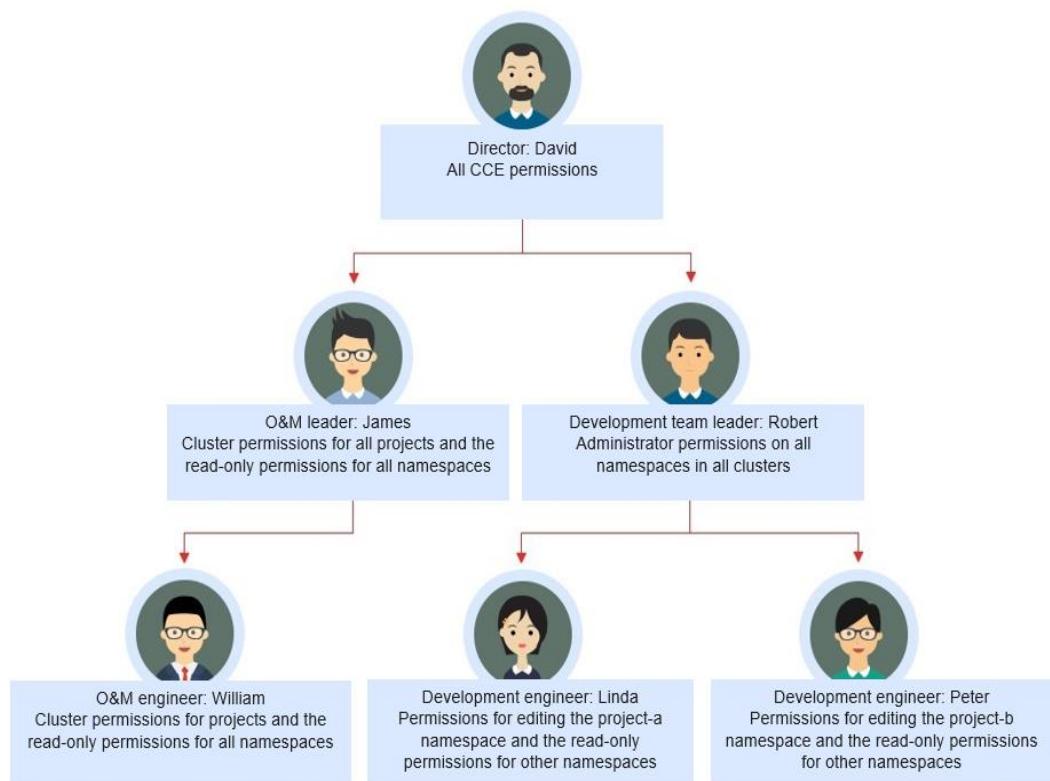
Cluster permissions and namespace permissions are independent of each other but must be used together. The permissions set for a user group apply to all users in the user group. When multiple permissions are added to a user or user group, they take effect at the same time (the union set is used).

### Permission Design

The following uses company X as an example.

Generally, a company has multiple departments or projects, and each department has multiple members. Design how permissions are to be assigned to different groups and projects, and set a user name for each member to facilitate subsequent user group and permissions configuration.

The following figure shows the organizational structure of a department in a company and the permissions to be assigned to each member:



## Director: David

David is a department director of company X. To assign him all CCE permissions (both cluster and namespace permissions), create the **cce-admin** user group for David on the IAM console and assign the CCE Administrator role.

### NOTE

**CCE Administrator:** This role has all CCE permissions. You do not need to assign other permissions.

**CCE FullAccess and CCE ReadOnlyAccess:** These policies are related to cluster management permissions and configured only for cluster-related resources (such as clusters and nodes). You must also configure namespace permissions to perform operations on Kubernetes resources (such as workloads and Services).

## O&M Leader: James

James is the O&M team leader of the department. He needs the cluster permissions for all projects and the read-only permissions for all namespaces.

To assign the permissions, create a user group named **cce-sre** on the IAM console and add James to this user group. Then, assign CCE FullAccess to the user group **cce-sre** to allow it to perform operations on clusters in all projects.

### Assigning Read-only Permissions on All Clusters and Namespaces to All Team Leaders and Engineers

You can create a read-only user group named **read\_only** on the IAM console and add users to the user group.

- Although the development engineers Linda and Peter do not require cluster management permissions, they still need to view data on the CCE console. Therefore, the read-only cluster permission is required.
- For the O&M engineer William, assign the read-only permission on clusters to him in this step.
- The O&M team leader James already has the management permissions on all clusters. You can add him to the **read\_only** user group to assign the read-only permission on clusters to him.

Users James, Robert, William, Linda, and Peter are added to the **read\_only** user group.

Assign the read-only permission on clusters to the user group **read\_only**.

Return to the CCE console, and add the read-only permission on namespaces to the user group **read\_only** to which the five users belong. Choose **Permissions** on the CCE console, and assign the read-only policy to the user group **read\_only** for each cluster.

**Figure 15-5** Assigning the read-only permission on namespaces to the user group

**Add Permission** ×

Cluster Name

User/User Group   [Create User Group](#)

Namespace  [Create Namespace](#)

Permission Type  Administrator  O&M  Read-only  Developer  Custom

Description Read-only permissions for most resources in all or selected namespaces.

After the setting is complete, James has the cluster management permissions for all projects and the read-only permissions on all namespaces, and the Robert, William, Linda, and Peter have the read-only permission on all clusters and namespaces.

### Development Team Leader: Robert

In the previous steps, Robert has been assigned the read-only permission on all clusters and namespaces. Now, assign the administrator permissions on all namespaces to Robert.

Therefore, assign the administrator permissions on all namespaces in all clusters to Robert.

**Figure 15-6** Assigning the administrator permissions on namespaces to Robert

✕

### Add Permission

Cluster Name

User/User Group   [Create User](#)

Namespace  [Create Namespace](#)

Permission Type Administrator O&M Read-only Developer Custom

Description Read and write permissions on all resources in all namespaces.

## O&M Engineer: William

In the previous steps, William has been assigned the read-only permission on all clusters and namespaces. He also requires the cluster management permissions in his region. Therefore, you can log in to the IAM console, create a user group named **cce-sre-b4** and assign CCE FullAccess to William for his region.

Now, William has the cluster management permissions for his region and the read-only permission on all namespaces.

## Development Engineers: Linda and Peter

In the previous steps, Linda and Peter have been assigned the read-only permission on clusters and namespaces. Therefore, you only need to assign the edit policy to them.

**Figure 15-7** Assigning the edit policy on namespaces

✕

### Add Permission

Cluster Name

User/User Group   [Create User](#)

Namespace  [Create Namespace](#)

Permission Type Read-only Developer Custom

Description 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.

By now, all the required permissions are assigned to the department members.

# 15.5 Permission Dependency of the CCE Console

Some CCE permissions policies depend on the policies of other cloud services. To view or use other cloud resources on the CCE console, enable the access control feature of IAM and assign dependency policies for the other cloud services.

- Dependency policies are assigned based on the CCE FullAccess or CCE ReadOnlyAccess policy you configure.
- Only users and user groups with namespace permissions can gain the view access to resources in clusters.
  - If a user is granted the view access to all namespaces of a cluster, the user can view all namespace resources (except secrets) in the cluster. To view secrets in the cluster, the user must gain the **admin** or **edit** role in all namespaces of the cluster.
  - The **view** role within a single namespace allows users to view resources only in the specified namespace.

## Dependency Policy Configuration

To grant an IAM user the permissions to view or use resources of other cloud services on the CCE console, you must first grant the CCE Administrator, CCE FullAccess, or CCE ReadOnlyAccess policy to the user group to which the user belongs and then grant the dependency policies listed in [Table 15-3](#) to the user. These dependency policies will allow the IAM user to access resources of other cloud services.

 **NOTE**

CCE supports fine-grained permissions configuration, but has the following restrictions:

- AOM does not support resource-level monitoring. After operation permissions on specific resources are configured using IAM's fine-grained cluster resource management function, IAM users can view cluster monitoring information on the **Dashboard** page of the CCE console, but cannot view the data on non-fine-grained metrics.

**Table 15-3** Dependency policies

Console Function	Dependent Service	Role or Policy Required
Cluster overview	Application Operations Management (AOM)	<ul style="list-style-type: none"> <li>• An IAM user with the CCE Administrator permission assigned can use this function only after the AOM FullAccess permission is assigned.</li> <li>• IAM users with IAM ReadOnlyAccess, CCE FullAccess, or CCE ReadOnlyAccess assigned can directly use this function.</li> </ul>

Console Function	Dependent Service	Role or Policy Required
Workload management	Elastic Load Balance (ELB) Application Performance Management (APM) Application Operations Management (AOM) NAT Gateway Object Storage Service (OBS) Scalable File Service (SFS)	<p>Except in the following cases, the user does not require any additional role to create workloads.</p> <ul style="list-style-type: none"> <li>To create a Service using ELB, you must have the ELB FullAccess or ELB Administrator plus VPC Administrator permissions assigned.</li> <li>To use a Java probe, you must have the AOM FullAccess and APM FullAccess permissions assigned.</li> <li>To create a Service using NAT Gateway, you must have the NAT Gateway Administrator permission assigned.</li> <li>To use OBS, you must have the OBS Administrator permission globally assigned.</li> </ul> <p><b>NOTE</b> Because of the cache, it takes about 13 minutes for the RBAC policy to take effect after being granted to users and user groups. After an OBS-related system policy is granted, it takes about 5 minutes for the policy to take effect.</p> <ul style="list-style-type: none"> <li>To use SFS, you must have the SFS FullAccess permission assigned.</li> </ul>
Cluster management	Application Operations Management (AOM)	<ul style="list-style-type: none"> <li>Auto scale-out or scale-up requires the AOM FullAccess policy.</li> </ul>
Node management	Elastic Cloud Server (ECS)	<p>If the permission assigned to an IAM user is CCE Administrator, creating or deleting a node requires the ECS FullAccess or ECS Administrator policy and the VPC Administrator policy.</p>
Service	Elastic Load Balance (ELB) NAT Gateway	<p>Except in the following cases, the user does not require any additional role to create a Service.</p> <ul style="list-style-type: none"> <li>To create a Service using ELB, you must have the ELB FullAccess or ELB Administrator plus VPC Administrator permissions assigned.</li> <li>To create a Service using NAT Gateway, you must have the NAT Administrator permission assigned.</li> </ul>

Console Function	Dependent Service	Role or Policy Required
Storage	Object Storage Service (OBS) Scalable File Service (SFS) SFS Turbo	<ul style="list-style-type: none"> <li>To use OBS, you must have the OBS Administrator permission globally assigned.</li> </ul> <p><b>NOTE</b> Because of the cache, it takes about 13 minutes for the RBAC policy to take effect after being granted to users and user groups. After an OBS-related system policy is granted, it takes about 5 minutes for the policy to take effect.</p> <ul style="list-style-type: none"> <li>To use SFS, you must have the SFS FullAccess permission assigned.</li> <li>To use SFS Turbo, you must have the SFS Turbo FullAccess permission.</li> </ul> <p>The CCE Administrator role is required for importing storage devices.</p>
Namespace management	/	/
Chart management	/	Cloud accounts and the IAM users with CCE Administrator assigned can use this function.
Add-ons	/	Cloud accounts and the IAM users with CCE Administrator, CCE FullAccess, or CCE ReadOnlyAccess assigned can use this function.
Permissions management	/	<ul style="list-style-type: none"> <li>For cloud accounts, no additional policy/role is required.</li> <li>IAM users with the CCE FullAccess or CCE ReadOnlyAccess permission can use this function. In addition, the IAM users must have the <b>administrator permissions (cluster-admin)</b> on the namespace.</li> </ul>
ConfigMaps and Secrets	/	<ul style="list-style-type: none"> <li>Creating ConfigMaps does not require any additional policy.</li> <li>Viewing secrets requires that the cluster-admin, admin, or edit permission be configured for the namespace. The DEW KeypairFullAccess or DEW KeypairReadOnlyAccess policy must be assigned for dependent services.</li> </ul>

Console Function	Dependent Service	Role or Policy Required
Help center	/	/
Switching to other related services	Software Repository for Container (SWR)	The CCE console provides links to other related services. To view or use these services, an IAM user must be assigned required permissions for the services.

## 15.6 Service Account Token Security Improvement

In clusters earlier than v1.21, a token is obtained by mounting the secret of the service account to a pod. Tokens obtained this way are permanent. This approach is no longer recommended starting from version 1.21. Service accounts will stop auto creating secrets in clusters from version 1.25.

In clusters of version 1.21 or later, you can use the [TokenRequest](#) API to obtain the token and use the projected volume to mount the token to the pod. Such tokens are valid for a fixed period (one hour by default). Before expiration, kubelet refreshes the token to ensure that the pod always uses a valid token. When the mounting pod is deleted, the token automatically becomes invalid. This approach is implemented by the [BoundServiceAccountTokenVolume](#) feature to improve the token security of the service account. Kubernetes clusters of v1.21 and later enable this approach by default.

For smooth transition, the community extends the token validity period to one year by default. After one year, the token becomes invalid, and clients that do not support certificate reloading cannot access the API server. It is recommended that clients of earlier versions be upgraded as soon as possible. Otherwise, service faults may occur.

If you use a Kubernetes client of a to-be-outdated version, the certificate reloading may fail. Versions of officially supported Kubernetes client libraries able to reload tokens are as follows:

- Go:  $\geq$  v0.15.7
- Python:  $\geq$  v12.0.0
- Java:  $\geq$  v9.0.0
- Javascript:  $\geq$  v0.10.3
- Ruby: master branch
- Haskell: v0.3.0.0
- C#:  $\geq$  7.0.5

For details, visit <https://github.com/kubernetes/enhancements/tree/master/keps/sig-auth/1205-bound-service-account-tokens>.



 NOTE

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

## Diagnosis

Perform the following steps to check your CCE clusters of v1.21 or later:

1. Use kubectl to access the cluster and run the **kubectl get --raw "/metrics" | grep stale** command to obtain the metrics. Check the metric named **serviceaccount\_stale\_tokens\_total**.

If the value is greater than 0, some workloads in the cluster may be using an earlier client-go version. In this case, check whether this problem occurs in your deployed applications. If yes, upgrade client-go to the version specified by the community as soon as possible. The version must be at least two major versions of the CCE cluster. For example, if your cluster version is 1.23, the Kubernetes dependency library version must be at least 1.19.

```
[root@ ]# kubectl get --raw "/metrics" | grep stale
# HELP serviceaccount_stale_tokens_total [ALPHA] Cumulative stale projected service account tokens used
# TYPE serviceaccount_stale_tokens_total counter
serviceaccount_stale_tokens_total 52 |
```

# 16 Settings

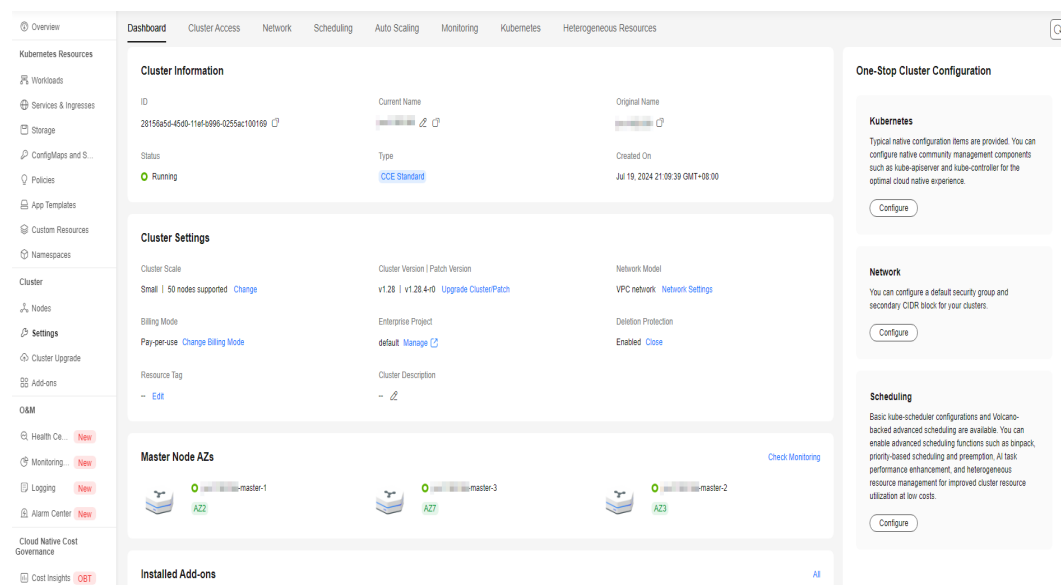
## 16.1 Dashboard

Settings offers you an entry to check and modify the basic settings of clusters. It includes information from dimension like [Cluster Information](#), [Cluster Settings](#), [Master Node AZs](#), and [Installed Add-ons](#).

### Accessing the Settings

- Step 1** Log in to the CCE console and click the name of the target cluster to access the cluster console.
- Step 2** In the navigation pane, choose **Settings** and click the **Dashboard** tab.


Figure 16-1 Checking the cluster dashboard



----End

## Cluster Information

It includes:

- **ID:** uniquely identifies a cluster resource. It is automatically generated after a cluster is created and can be used in scenarios such as API calling.
- **Current Name:** After a cluster is created, you can click  to change its name.
- **Original Name:** specifies a cluster's original name after its current name is changed. The current name of a cluster must be unique.
- **Status:** specifies the status of a cluster. For details, see [Cluster Lifecycle](#).
- **Type:** specifies whether a cluster is a CCE standard or a CCE Turbo cluster. For details about the differences between CCE standard and CCE Turbo clusters, see [Comparison Between Cluster Types](#).
- **Created On:** specifies the time when a cluster was created. You will be billed based on the creation time of clusters.

## Cluster Settings

After a cluster is created, you can modify the following items for it:

- **Cluster Scale:** specifies how many nodes a cluster can manage at most. You can change the cluster scale as needed. For details, see [Changing Cluster Scale](#).
- **Cluster Version | Patch Version:** specifies the Kubernetes version and CCE patch version of a cluster.
- **Network Model:** specifies the network model of a cluster, which cannot be changed. For details about network models, see [Container Network](#).
- **Billing Mode:** specifies the billing mode of a cluster.
- **Enterprise Project:** specifies the enterprise project to which a cluster belongs.
- **Deletion Protection:** After deletion protection is enabled, you need to confirm the operation by using the virtual MFA device, SMS messages, or emails. You can go to the IAM console, choose **Security Settings > Critical Operations**, and enable the protection functions.
- **Resource Tag:** You can add resource tags to classify resources.
- **Cluster Description:** specifies the description that you entered for a cluster. A maximum of 200 characters are allowed.

## Master Node AZs

You can check how many master nodes are supported in a cluster.

## Installed Add-ons

You can check the installed add-ons in a cluster. If there is an add-on to be upgraded in the cluster, click **Go to Upgrade** to go to the **Add-ons** page and see more details.

## 16.2 Cluster Access

### Access Mode

- **kubectl:** You need to download and configure the kubectl and kubeconfig configuration files first, and then use kubectl to access a Kubernetes cluster. For details, see [Connecting to a Cluster Using kubectl](#).
- **EIP:** You can bind an EIP to the cluster API server. Then, the cluster API server can access the public network.

#### NOTE

- Binding an EIP to a cluster could bring the cluster potential access risks from the public network. You are advised to harden the inbound rule of port 5443 for the master node security group.
- This operation will restart kube-apiserver and update the cluster access certificate (kubeconfig). Do not perform any operations on the cluster during this time.
- **Custom SAN:** The Subject Alternative Name (SAN) allows multiple values (including IP addresses and domain names) to be associated with certificates. After the custom SAN is configured in the cluster access certificate, the cluster can be accessed using the domain names or IP addresses specified by the SAN. For details, see [Accessing a Cluster Using a Custom Domain Name](#).

#### NOTE

This operation will restart kube-apiserver and update the cluster access certificate (kubeconfig). Do not perform any operations on the cluster during this time.

### Authentication

CCE allows you to download the X509 certificate, which contains the **client.key**, **client.crt**, and **ca.crt** files. Keep your certificate secure.

For details about how to use a certificate to access clusters, see [Accessing a Cluster Using an X.509 Certificate](#).

## Server Request Settings

**Table 16-1** Parameters

Item	Parameter	Description	Value
Maximum Number of Concurrent Modification API Calls	max-mutating-requests-inflight	<p>Maximum number of concurrent modification API calls. Any requests that exceeding the specified value will be rejected by the server.</p> <p>The value <b>0</b> specifies that there is no limitation on the maximum number of concurrent modification calls. This parameter is related to the cluster scale. You are advised not to change the value.</p>	<p>Manual configuration is no longer supported since clusters of v1.21. The value is automatically specified based on the cluster scale:</p> <ul style="list-style-type: none"> <li>• <b>200</b> for clusters with 50 or 200 worker nodes</li> <li>• <b>500</b> for clusters with 1000 worker nodes</li> <li>• <b>1000</b> for clusters with 2000 worker nodes</li> </ul>

Item	Parameter	Description	Value
Maximum Number of Concurrent Non-Modification API Calls	max-requests-inflight	<p>Maximum number of concurrent non-modification API calls. Any requests that exceeding the specified value will be rejected by the server.</p> <p>The value <b>0</b> specifies that there is no limitation on the maximum number of concurrent non-modification calls. This parameter is related to the cluster scale. You are advised not to change the value.</p>	<p>Manual configuration is no longer supported since clusters of v1.21. The value is automatically specified based on the cluster scale:</p> <ul style="list-style-type: none"> <li>• <b>400</b> for clusters with 50 or 200 worker nodes</li> <li>• <b>1000</b> for clusters with 1000 worker nodes</li> <li>• <b>2000</b> for clusters with 2000 worker nodes</li> </ul>
Request Timeout	request-timeout	<p>Default request timeout interval of kube-apiserver. Exercise caution when changing the value of this parameter. Ensure that the changed value is proper to prevent frequent API timeout or other errors.</p> <p>This parameter is available only in clusters of v1.19.16-r30, v1.21.10-r10, v1.23.8-r10, v1.25.3-r10, and later versions.</p>	<p>Default: 1m0s</p> <p>Options: Min ≥ 1s Max ≤ 1 hour</p>

Item	Parameter	Description	Value
Overload Control	support-overload	<p>Cluster overload control. If enabled, concurrent requests 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 <a href="#">Enabling Overload Control for a Cluster</a>.</p> <p>This parameter is available only in clusters of v1.23 or later.</p> <p><b>NOTE</b> In particular scenarios, such as request burst over a short period of time, a cluster could still be overloaded even though overload control is enabled for it. In such cases, you are advised to manage and control access to the cluster in a timely manner.</p>	None

## 16.3 Network

You can configure a default security group and secondary CIDR block for your clusters.

### Cluster Network

**Table 16-2** Parameters

Parameter	Description
VPC	<p>VPC where a cluster resides</p> <p>VPC enables you to provision logically isolated, configurable, manageable virtual networks for cloud servers, cloud containers, and cloud databases. The VPC gives you complete control over your virtual network, allowing you to select your own IP address range, create subnets, configure security groups, and even assign EIPs and allocate bandwidth in your network, enabling secure and easy access to your business system.</p>
VPC CIDR Block	VPC CIDR Block of a cluster

Parameter	Description
Default Node Subnet	<p>Node subnet of a cluster</p> <p>A subnet is a network that manages ECS network planes. It supports IP address management and DNS. ECSs in a subnet are assigned with its IP addresses.</p> <p>By default, ECSs in all subnets of the same VPC can communicate with one another, but ECSs in different VPCs cannot.</p> <p>You can create a VPC peering connection to enable ECSs in different VPCs to communicate with each other.</p>
Default Node Subnet   IPv4 CIDR Block	Node subnet CIDR block of a cluster
Network Model	<p>Network model of a cluster</p> <p>After a cluster is created, its network model cannot be changed. For details about the comparison between different network models, see <a href="#">Overview</a>.</p>
Default Node Security Group	<p>Default security group of the worker nodes in a cluster</p> <p>You can select a custom security group as the default node security group for a cluster, and you need to allow traffic from specified ports in the security group to ensure normal communications in the cluster.</p> <p>If the custom security group needs some modifications, the modified security group applies only to newly created or accepted nodes. For existing nodes, you need to manually modify the security group rules.</p>



Parameter	Description
Retain the non-spoofed CIDR block of the original pod IP address. (available only in clusters using the VPC network model)	<p>In a cluster using a VPC network, 10.0.0.0/8, 172.16.0.0/12, and 192.168.0.0/16 are regarded as private CIDR blocks of the cluster by default. If the VPC to which the cluster resides uses a secondary CIDR block, operations such as creating or resetting a node will also add the secondary CIDR block to the private CIDR blocks.</p> <p>If a pod tries to access a private CIDR block, the source node will not perform NAT on the pod IP address. Instead, the upper-layer VPC 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.</p> <p>This function is available only in clusters of v1.23.14-r0, v1.25.9-r0, v1.27.6-r0, v1.28.4-r0, or later versions. For details, see <a href="#">Retaining the Original IP Address of a Pod</a>.</p> <p><b>NOTE</b></p> <p>To enable a node to access a pod in another node, the node CIDR block must be added to this parameter.</p> <p>Similarly, to enable an ECS to access the IP address of a pod in a cluster that is in the same VPC as the ECS, the ECS CIDR block must be added to this parameter.</p>
Pod's Access to Metadata (available only in CCE Turbo clusters)	<p>Whether to allow pods in a cluster to access the node metadata, such as AZs and enterprise project IDs.<a href="#">ECS Metadata Types</a>. This function is available only in clusters of v1.23.13-r0, v1.25.8-r0, v1.27.5-r0, v1.28.3-r0, or later versions.</p> <ul style="list-style-type: none"> <li>• If a pod is created when the function is enabled, whether it can access metadata depends on the function status.</li> <li>• If a pod is created when the function is disabled or in a cluster of an earlier version, it cannot access metadata regardless of the function status. To grant a pod access to metadata, it must be rebuilt while the function is enabled.</li> </ul>

## Service Settings

Table 16-3 Parameters

Parameter	Description
Request Forwarding	<p>Forwarding mode of a cluster</p> <p>After a cluster is created, the service forwarding mode cannot be changed. IPVS and iptables are supported. For details, see <a href="#">Comparing iptables and IPVS</a>.</p>

Parameter	Description
Service CIDR Block	Each Service in a cluster has its own IP address. When creating a CCE cluster, you can specify the Service address range (Service CIDR block). The Service CIDR block cannot overlap with the subnet or the container CIDR block. The Service CIDR block can be used only within a cluster.
Service Port Range	<p>NodePort port range</p> <p>The default port range is 30000 to 32767. The port range can be changed to 20106 to 32767. After changing the value, go to the security group page and change the TCP/UDP port range of node security groups 30000 to 32767. Otherwise, ports other than the default port cannot be accessed from external networks.</p> <p><b>NOTE</b></p> <p>If the port number is smaller than 20106, a conflict may occur between the port and the system health check port, which may further lead to unavailable cluster. If the port number is greater than 32767, a conflict may occur between the port and the random port of the OS, which may further affect the performance.</p>

## Container CIDR Blocks (Available only in Clusters Using the VPC Network Model)

If a container CIDR block configured during cluster creation cannot meet service expansion requirements, you can add more container CIDR blocks. For details, see [Adding a Container CIDR Block for a Cluster](#).

### NOTE

- This function is available only for clusters of v1.19 or later using a VPC network.
- An added container CIDR block cannot be deleted.

## Custom Container Network (Available only in CCE Turbo Clusters)

If you want different namespaces or workloads to use different subnet CIDR blocks or security groups, you can create a policy to associate subnets or security groups with namespaces or workloads. For details, see [Binding a Subnet and Security Group to a Namespace or Workload Using a Container Network Configuration](#).

- Associating a pod with a subnet: The pod IP address is restricted in a specific CIDR block. The networks between different namespaces or workloads are isolated from each other.
- Associating a pod with a security group: You can configure security group rules for pods in the same namespace or workload to customize access policies.

### NOTE

This configuration is supported only by CCE Turbo clusters. Pod subnets can be deleted from clusters of v1.23.17-r0, v1.25.12-r0, v1.27.9-r0, v1.28.7-r0, v1.29.3-r0, or later versions.

## Container Network Pre-binding Settings (Available only in CCE Turbo Clusters)

A CCE Turbo cluster requests for and binds an ENI or sub-ENI to each pod. Pods support fast scaling. However, it takes some time to create and bind an ENI to a pod, which slows down the pod startup speed if large-scale ENIs are to be created in batches. Dynamic container ENI pre-binding is enabled by default to speed up pod startup while improving the IP address usage. Cluster pre-binding policies take effect globally. Cluster nodes will pre-bind container ENIs based on the configured policies. To configure a separate pre-binding policy for a group of nodes, enable node pool pre-binding.

### NOTE

This configuration is supported only by CCE Turbo clusters.

### All Container ENI Pre-binding

- After this function is enabled, your cluster nodes will request for and bind the maximum number of ENIs supported by the node flavor. For example, if the maximum number of sub-ENIs supported by s7.large.2 nodes is 16, CCE will dynamically pre-bind 16 sub-ENIs to each node of this flavor.
- After this function is disabled, you can customize the pre-binding parameters on the console.

**Table 16-4** Parameters of the dynamic ENI pre-binding policy

Parameter	Default Value	Description	Suggestion
nic-minimum-target	10	Minimum number of container ENIs bound to a node. The parameter value must be a positive integer. The value <b>10</b> indicates that there are at least 10 container ENIs bound to a node. If the number you entered exceeds the container ENI quota of the node, the ENI quota will be used.	Configure these parameters based on the number of pods.

Parameter	Default Value	Description	Suggestion
nic-maximum-target	0	<p>If the number of ENIs bound to a node exceeds the value of <b>nic-maximum-target</b>, the system does not proactively pre-bind ENIs.</p> <p>If the value of this parameter is greater than or equal to the value of <b>nic-minimum-target</b>, the check on the maximum number of the pre-bound ENIs is enabled. Otherwise, the check is disabled.</p> <p>The parameter value must be a positive integer. The value <b>0</b> indicates that the check on the upper limit of pre-bound container ENIs is disabled. If the number you entered exceeds the container ENI quota of the node, the ENI quota will be used.</p>	Configure these parameters based on the number of pods.
nic-warm-target	2	<p>Minimum number of pre-bound ENIs on a node. The value must be a number.</p> <p>When the value of <b>nic-warm-target</b> + the number of bound ENIs is greater than the value of <b>nic-maximum-target</b>, the system will pre-bind ENIs based on the difference between the value of <b>nic-maximum-target</b> and the number of bound ENIs.</p>	Set this parameter to the number of pods that can be scaled out instantaneously within 10 seconds.

Parameter	Default Value	Description	Suggestion
nic-max-above-warm-target	2	<p>Only when the number of idle ENIs on a node minus the value of <b>nic-warm-target</b> is greater than the threshold, the pre-bound ENIs will be unbound and reclaimed. The value can only be a number.</p> <ul style="list-style-type: none"> <li>Setting a larger value of this parameter slows down the recycling of idle ENIs and accelerates pod startup. However, the IP address usage decreases, especially when IP addresses are insufficient. Therefore, <b>exercise caution when increasing the value of this parameter.</b></li> <li>Setting a smaller value of this parameter accelerates the recycling of idle ENIs and improves the IP address usage. However, when a large number of pods increase instantaneously, the startup of some pods slows down.</li> </ul>	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.

## 16.4 Scheduling

Basic kube-scheduler configurations and Volcano-backed advanced scheduling are available. You can enable advanced scheduling functions such as bin packing, priority-based scheduling and preemption, AI task performance enhancement, and heterogeneous resource management for improved cluster resource utilization at low costs.

### Default Cluster Scheduler Configuration

#### Default cluster scheduler (default-scheduler)

The Kubernetes scheduler discovers newly created pods that have not been accepted by nodes and assigns them to proper nodes. You are allowed to use multiple schedulers in the same cluster. kube-scheduler is the default cluster scheduler provided by the Kubernetes community. CCE also supports the enhanced Volcano scheduler, which offers general computing capabilities like high-performance job scheduling engine, heterogeneous chip management, and task management.

You can use kube-scheduler together with the Volcano scheduler, or use kube-scheduler or Volcano scheduler separately.

**Table 16-5** Default cluster scheduler

Scheduler Name	Description	Configuration
<p>kube-scheduler</p>	<p>kube-scheduler provides the community native, standard scheduling capabilities.</p> <p>If kube-scheduler is set as the default scheduler in a cluster and the Volcano Scheduler add-on (<b>Volcano Scheduler</b>) is also installed in the cluster, the enhanced Volcano capabilities are enabled by default. This ensures that you have access to advanced scheduling capabilities, including resource utilization optimization, AI job performance enhancement, and heterogeneous resource management. They help reduce costs while improving resource utilization. In this case, kube-scheduler schedules common workloads, while the Volcano scheduler schedules some specified workloads. For details, see <a href="#">Scheduling Workloads</a>.</p>	<p>kube-scheduler configurations:</p> <ul style="list-style-type: none"> <li>• <a href="#">Scheduler Performance Configuration</a></li> <li>• <a href="#">Priority-based Scheduling</a></li> </ul> <p>Enhanced configurations after the Volcano scheduler is installed:</p> <ul style="list-style-type: none"> <li>• <a href="#">Resource Utilization Optimization Scheduling (Supported by the Volcano Scheduler)</a></li> <li>• <a href="#">AI Job Performance Enhancement Scheduling (Supported by the Volcano Scheduler)</a></li> <li>• <a href="#">Heterogeneous Resource Scheduling (Supported by the Volcano Scheduler)</a></li> </ul>

Scheduler Name	Description	Configuration
Volcano scheduler (available in clusters v1.27 or later)	<p>If the Volcano scheduler is set as the default scheduler in a cluster, kube-scheduler will no longer work. The Volcano scheduler schedules all workload tasks in the cluster.</p> <p>Volcano provides enhanced scheduling capabilities in addition to the capabilities provided by kube-scheduler. Before using this scheduler, you need to install the Volcano Scheduler add-on first. For details, see <a href="#">Volcano Scheduler</a>.</p>	<p>Enhanced configurations of the Volcano scheduler:</p> <ul style="list-style-type: none"> <li>• <a href="#">Priority-based Scheduling</a></li> <li>• <a href="#">Resource Utilization Optimization Scheduling (Supported by the Volcano Scheduler)</a></li> <li>• <a href="#">AI Job Performance Enhancement Scheduling (Supported by the Volcano Scheduler)</a></li> <li>• <a href="#">Heterogeneous Resource Scheduling (Supported by the Volcano Scheduler)</a></li> </ul>

## kube-scheduler

kube-scheduler provides the community native, standard scheduling capabilities.

Before **enabling volcano enhanced capabilities**, install Volcano Scheduler. Enabling this function will provide advanced scheduling capabilities, including optimizing resource utilization, enhancing AI job performance, and managing heterogeneous resources. This will ultimately improve cluster resource utilization and reduce costs.

Enhanced configurations of the Volcano scheduler:

- [Priority-based Scheduling](#)
- [Resource Utilization Optimization Scheduling \(Supported by the Volcano Scheduler\)](#)
- [AI Job Performance Enhancement Scheduling \(Supported by the Volcano Scheduler\)](#)
- [Heterogeneous Resource Scheduling \(Supported by the Volcano Scheduler\)](#)

## Scheduler Performance Configuration

### NOTE

Only kube-scheduler supports this configuration.

**Table 16-6** Parameters

Item	Parameter	Description	Value
QPS for communicating with kube-apiserver	kube-api-qps	QPS for communication with kube-apiserver	<ul style="list-style-type: none"> <li>If the number of nodes in a cluster is less than 1,000, the default value is <b>100</b>.</li> <li>If the number of nodes in a cluster is 1,000 or more, the default value is <b>200</b>.</li> </ul>
Burst for communicating with kube-apiserver	kube-api-burst	Burst for communication with kube-apiserver	<ul style="list-style-type: none"> <li>If the number of nodes in a cluster is less than 1,000, the default value is <b>100</b>.</li> <li>If the number of nodes in a cluster is 1,000 or more, the default value is <b>200</b>.</li> </ul>

## Priority-based Scheduling

### Scheduling based on priority

This is a basic scheduling capability and cannot be disabled. The scheduler preferentially guarantees the running of high-priority pods, and will not evict low-priority pods that are running. For details, see [Priority-based Scheduling](#).

### Whether to enable preemption (supported by the Volcano scheduler)

After this function is enabled, 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](#).

#### NOTE

- This configuration is supported when Volcano is selected as the default scheduler.
- Enabling both priority-based preemption scheduling and delayed pod creation simultaneously is not possible.



## Resource Utilization Optimization Scheduling (Supported by the Volcano Scheduler)

### Bin packing

With this option enabled, the cluster scheduler schedules pods to nodes that have the most requested resources. This reduces resource fragments on each node and improves the resource utilization of the cluster. For details, see [Bin Packing](#).

For details about the bin packing weight and weights for each resource to score nodes, see [Table 16-7](#).

**Table 16-7** Bin packing weight

Item	Description	Default Value
Binpack Scheduling Strategy Weight	A larger value indicates a higher weight of the bin packing policy in overall scheduling.	10
CPU Weight	A larger value indicates a higher cluster CPU usage.	1
Memory Weight	A larger value indicates a higher cluster memory usage.	1
Custom Resource Type	Other custom resource types requested by pods, for example, <b>nvidia.com/gpu</b> . A larger value indicates a higher usage of the specified cluster resource.	None

### Load-aware scheduling (usage)

This function uses the Cloud Native Cluster Monitoring (kube-prometheus-stack) add-on to obtain the actual CPU and memory load of each node, calculates the average load of each node based on the specified period, and preferentially schedules jobs to the node with the lightest load to balance load. For details, see [Load-aware Scheduling](#).

## AI Job Performance Enhancement Scheduling (Supported by the Volcano Scheduler)

### Fair Scheduling Policy (DRF)

Dominant Resource Fairness (DRF) is a scheduling algorithm based on the dominant resource of a container group. It supports fair allocation of multiple types of resources and is suitable for batch AI training and big data jobs. DRF is suitable for batch process small scale services like single AI model training and single big data computing and query, because it preferentially considers the throughput of services in clusters.

DRF helps you enhance the service throughput of clusters and improve service performance. For details, see [DRF](#).

### Workload Group Scheduling Policy (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](#).

## Heterogeneous Resource Scheduling (Supported by the Volcano Scheduler)

### Support GPU resource scheduling

To use this capability, the CCE AI Suite (NVIDIA GPU) add-on ([CCE AI Suite \(NVIDIA GPU\)](#)) must be installed. With this option enabled, GPUs can be used for AI training jobs, and the scheduler provides full GPU dispatch and GPU sharing to improve resource utilization.

### Support NPU resource scheduling

To use this capability, the CCE AI Suite (Ascend NPU) add-on ([CCE AI Suite \(Ascend NPU\)](#)) must be installed. With this option enabled, Ascend NPUs can be used for AI training jobs, and the scheduler provides NPU topology-aware scheduling to improve training job efficiency.

## 16.5 Auto Scaling

### Auto Scale-out Configuration

CCE Cluster Autoscaler comprehensively checks the resource statuses of an entire cluster. When the load of a microservice is high (for example, the CPU or memory usage is too high), it will add more pods to reduce the load.

#### Node Capacity Expansion Conditions

- Auto scale-out when the workload cannot be scheduled: If a workload pod cannot be scheduled, the system automatically scales out the node pool with auto scaling enabled. If the pod has been configured to be affinity for a node, the system will not automatically add more nodes.

Such auto scaling works with an HPA policy. For details, see [Using HPA and CA for Auto Scaling of Workloads and Nodes](#).

- User-defined policy switch: specifies whether to automatically scale out a node pool based on the [node scaling policies](#). This function is enabled by default.

#### Upper limit of resources to be expanded

- **Total Nodes:** specifies how many nodes can be present in a cluster during scale-out. If there are more nodes in the cluster than the specified value, the cluster will not add nodes. The default value is determined by how many nodes a cluster can manage at most.
- **Total Cores:** specifies how many cores on all nodes can be present in a cluster during scale-out. If there are more cores in the cluster than the specified value, the cluster will not add nodes. By default, the number is not limited.

- **Total Memory** (GiB): specifies the upper limit of the total memory of all nodes in a cluster during scale-out. If the total memory exceeds the specified value, the cluster will not add nodes. By default, the number is not limited.

#### NOTE

When the total number of nodes, CPUs, and memory is collected, unavailable nodes in custom node pools are included but unavailable nodes in the default node pool are not included.

### Scale-out Priority

You can drag and drop node pools in the list to adjust their scale-out priorities.

## Auto Scale-in Configuration

CCE Cluster Autoscaler comprehensively checks the resource statuses of an entire cluster. Once it confirms that workload pods can be scheduled and run properly, it automatically obtains nodes for scale-in.

### Node Scale-in Conditions

- **Node Resource Condition:** When the requested cluster node resources (both CPU and memory) are lower than a certain percentage (50% by default) for a period of time (10 minutes by default), a cluster scale-in is triggered.
- **Node Status Condition:** If a node is unavailable for a specified period of time, the node will be automatically reclaimed. The default value is 20 minutes.
- **Scale-in Exception Scenarios:** When a node meets the following exception scenarios, CCE will not scale in the node even if the node resources or status meets scale-in conditions:
  - a. Resources on other nodes in the cluster are insufficient.
  - b. Scale-in protection is enabled on the node. To enable or disable node scale-in protection, choose **Nodes** in the navigation pane and then click the **Nodes** tab. Locate the target node, choose **More**, and then enable or disable node scale-in protection in the **Operation** column.
  - c. There is a pod with the non-scale label on the node.
  - d. Policies such as reliability have been configured on some containers on the node.
  - e. There are non-DaemonSet containers in the **kube-system** namespace on the node.
  - f. (Optional) A container managed by a third-party pod controller is running on a node. Third-party pod controllers are for custom workloads except Kubernetes-native workloads such as Deployments and StatefulSets. Such controllers can be created using [CustomResourceDefinitions](#).

### Node Scale-in Policy

**Table 16-8** Node scale-in policy configurations

Item	Description	Default Value
Number of Concurrent Scale-In Requests	<p>Maximum number of idle nodes that can be deleted concurrently.</p> <p>Only idle nodes can be concurrently scaled in. Nodes that are not idle can only be scaled in one by one.</p> <p><b>NOTE</b> During a node scale-in, if the pods on the node do not need to be evicted (such as DaemonSet pods), the node is idle. Otherwise, the node is not idle.</p>	10
Node Recheck Timeout	Interval for rechecking a node that could not be removed	5 minutes
Cooldown Time	<p>Cooldown period for starting scale-in evaluation again after auto scale-in is triggered in a cluster</p> <p><b>NOTE</b> If both auto scale-out and scale-in exist in a cluster, set this parameter to 0 minutes. This prevents the node scale-in from being blocked due to continuous scale-out of some node pools or retries upon a scale-out failure, which results in unexpected waste of node resources.</p>	10 minutes
	Cooldown period for starting scale-in evaluation again after auto scale-out is triggered in a cluster	10 minutes
	Cooldown period for starting scale-in evaluation again after auto scale-in triggered in a cluster failed	3 minutes

## 16.6 Kubernetes

Typical native configuration items are provided. You can configure native community management components such as kube-apiserver and kube-controller for the best cloud native experience.

### API Server Configuration (kube-apiserver)

#### Container eviction configuration

By default, the default tolerance time applies to all containers in a cluster. You can also configure different tolerance times for pods. In this case, your custom settings take effect.

 **NOTE**

It is recommended that you properly configure the tolerance times for pods, or certain problems may occur.

- If the parameter is set to a low value, containers may be moved frequently during brief fault scenarios like network jitter, which can impact services.
- If the parameter is set to a high value, containers may not be moved for a long period of time in the event of a node failure, which can impact services.

**Table 16-9** Parameters

Item	Parameter	Description	Value
Toleration time for nodes in NotReady state	default-not-ready-toleration-seconds	Tolerance time when a node is not ready. If a node becomes unavailable, pods running on the node are evicted automatically after the tolerance time elapses. The default value is <b>300s</b> .	Default: 300s
Toleration time for nodes in unreachable state	default-unreachable-toleration-seconds	Tolerance time when a node is unreachable. If the environment is abnormal, for example, a node cannot be accessed (due to reasons such as abnormal node network), pods running on the node are evicted automatically after the tolerance time elapses. The default value is <b>300s</b> .	Default: 300s

**Admission Controller Add-on Configurations**

With Kubernetes, you can enable admission add-ons to limit and manage Kubernetes API objects (like pods, Services, and Deployments) prior to modifying them within a cluster.

 **NOTE**

This parameter is available only in clusters of v1.23.14-r0, v1.25.9-r0, v1.27.6-r0, v1.28.4-r0, or later versions.

**Table 16-10** Parameters

Item	Parameter	Description	Value
Node Restriction Add-on	enable-admission-plugin-node-restriction	This add-on allows the kubelet of a node to operate only the objects of the current node for enhanced isolation in multi-tenant scenarios or the scenarios with high security requirements. For details, see the <a href="#">official documentation</a> .	Enable/Disable
Pod Node Selector Add-on	enable-admission-plugin-pod-node-selector	This add-on allows cluster administrators to configure the default node selector through namespace annotations. In this way, pods run only on specific nodes and configurations are simplified.	Enable/Disable
Pod Toleration Limit Add-on	enable-admission-plugin-pod-toleration-restriction	This add-on allows cluster administrators to configure the default value and limits of pod tolerations through namespaces for fine-grained control over pod scheduling and key resource protection.	Enable/Disable

### Service Account Token Volume Projection

Kubelet can project a service account token into a pod. You can specify the desired properties of the token, such as the API audiences. The token will become invalid against the API when either the pod or the service account is deleted. For details, see the [official document](#).

 **NOTE**

This parameter is available only in clusters of v1.23.16-r0, v1.25.11-r0, v1.27.8-r0, v1.28.6-r0, v1.29.2-r0, or later versions.

**Table 16-11** Parameters

Item	Parameter	Description	Value
API Audience Settings	api-audiences	<p>Audiences for a service account token. The Kubernetes component for authenticating service account tokens checks whether the token used in an API request specifies authorized audiences.</p> <p>Configuration suggestion: Accurately configure audiences according to the communication needs among cluster services. By doing so, the service account token is used for authentication only between authorized services, which enhances security.</p> <p><b>NOTE</b> An incorrect configuration may lead to an authentication communication failure between services or an error during token verification.</p>	<p>Default value: "https://kubernetes.default.svc.cluster.local"</p> <p>Multiple values can be configured, which are separated by commas (,).</p>
Service Account Token Issuer Identity	service-account-issuer	<p>Entity identifier for issuing a service account token, which is the value identified by the <b>iss</b> field in the payload of the service account token.</p> <p>Configuration suggestion: Ensure the configured issuer URL can be accessed in the cluster and trusted by the authentication system in the cluster.</p> <p><b>NOTE</b> If your specified issuer URL is untrusted or inaccessible, the authentication process based on the service account may fail.</p>	<p>Default value: "https://kubernetes.default.svc.cluster.local"</p> <p>Multiple values can be configured, which are separated by commas (,).</p>

## Controller Configuration (kube-controller-manager)

### Common Configurations of the Controller

- **Controller performance configuration:** used to configure performance parameters for the controller to access kube-apiserver.

 **NOTE**

It is recommended that you properly configure the controller performance settings, or certain problems may occur.

- If a parameter is set to a small value, client traffic limiting may be triggered, affecting controller performance.
- If a parameter is set to a large value, kube-apiserver may be overloaded.

**Table 16-12** Parameters

Item	Parameter	Description	Value
QPS for communicating with kube-apiserver	kube-api-qps	QPS for communication with kube-apiserver	<ul style="list-style-type: none"> <li>• If the number of nodes in a cluster is less than 1,000, the default value is <b>100</b>.</li> <li>• If the number of nodes in a cluster is 1,000 or more, the default value is <b>200</b>.</li> </ul>
Burst for communicating with kube-apiserver	kube-api-burst	Burst for communication with kube-apiserver	<ul style="list-style-type: none"> <li>• If the number of nodes in a cluster is less than 1,000, the default value is <b>100</b>.</li> <li>• If the number of nodes in a cluster is 1,000 or more, the default value is <b>200</b>.</li> </ul>



- **Cluster controller concurrent configuration:** specifies the number of resource objects that are allowed to synchronize simultaneously. A larger value indicates a quicker response and higher CPU (and network) load.

 **NOTE**

It is recommended that you properly configure the controller concurrency, or certain problems may occur.

- If a parameter is set to a small value, the controller may respond slowly.
- If a parameter is set to a large value, the cluster management plane will be overloaded.

**Table 16-13** Parameters

Item	Parameter	Description	Value
Number of concurrent processing of deployment	concurrent-deployment-syncs	Number of Deployment objects that can be synchronized concurrently. A larger value indicates a quicker response to Deployments and higher CPU (and network bandwidth) pressure.	Default: 5
Concurrent processing number of endpoint	concurrent-endpoint-syncs	Number of endpoints that can be concurrently synchronized. A larger value indicates faster update of endpoints and higher CPU (and network) pressure.	Default: 5
Concurrent number of garbage collector	concurrent-gc-syncs	Number of garbage collector workers that are allowed to synchronize concurrently.	Default: 20
Number of job objects allowed to sync simultaneously	concurrent-job-syncs	Number of job objects that can be synchronized concurrently. A larger value indicates a quicker response to jobs and higher CPU (and network) usage.	Default: 5
Number of CronJob objects allowed to sync simultaneously	concurrent-cron-job-syncs	Number of CronJob objects that can be synchronized concurrently. A larger value indicates a quicker response to CronJobs and higher CPU (and network) usage.	Default: 5

Item	Parameter	Description	Value
Number of concurrent processing of namespace	concurrent-namespace-syncs	Number of namespace objects that can be synchronized concurrently. A larger value indicates a quicker response to namespaces and higher CPU (and network) usage.	Default: 10
Concurrent processing number of replicaset	concurrent-replicaset-syncs	Number of ReplicaSet objects that can be synchronized concurrently. A larger value indicates a quicker response to ReplicaSet management and higher CPU (and network) usage.	Default: 5
Number of concurrent processing of resource quota	concurrent-resource-quota-syncs	Number of ResourceQuota objects that can be synchronized concurrently. A larger value indicates a faster response to quota management and higher CPU (and network) usage.	Default: 5
Servicepace	concurrent-service-syncs	Number of Service objects that can be synchronized concurrently. A larger value indicates a faster response to Service management and higher CPU (and network) usage.	Default: 10
Concurrent processing number of serviceaccount-token	concurrent-serviceaccount-token-syncs	Number of service account token objects that can be synchronized concurrently. A larger value indicates faster token generation and higher CPU (and network) usage.	Default: 5
Concurrent processing of ttl-after-finished	concurrent-ttl-after-finished-syncs	Number of <b>ttl-after-finished-controller</b> workers that can be synchronized concurrently.	Default: 5

Item	Parameter	Description	Value
RC	concurrent_rc_syncs	Number of replication controllers that can be synchronized concurrently. A larger value indicates faster replica management operations and higher CPU (and network) usage. <b>NOTE</b> This parameter is used only in clusters of v1.19 or earlier.	Default: 5
RC	concurrent-rc-syncs	Number of replication controllers that can be synchronized concurrently. A larger value indicates faster replica management operations and higher CPU (and network) usage. <b>NOTE</b> This parameter is used only in clusters of v1.21 to v1.23. In clusters of v1.25 and later, this parameter is deprecated (officially deprecated from v1.25.3-r0 on).	Default: 5
HPA	concurrent-horizontal-pod-autoscaler-syncs	Maximum number of HPA auto scaling requests that can be processed concurrently. A larger value indicates a faster HPA auto scaling and higher CPU (and network) usage. This parameter is available only in clusters of v1.27 or later.	Default: 5 Value range: 1 to 50

### Node lifecycle controller (node-lifecycle-controller) configuration

 **NOTE**

This parameter is available only in clusters of v1.23.14-r0, v1.25.9-r0, v1.27.6-r0, v1.28.4-r0, or later versions.

**Table 16-14** Parameters

Item	Parameter	Description	Value
Unhealthy AZ Threshold	unhealthy-zone-threshold	<p>When more than a certain proportion of pods in an AZ are unhealthy, the AZ itself will be considered unhealthy, and scheduling pods to nodes in that AZ will be restricted to limit the impacts of the unhealthy AZ.</p> <p><b>NOTE</b> If the parameter is set to a large value, pods in unhealthy AZs will be migrated in a large scale, which may lead to risks such as overloaded clusters.</p>	Default: 0.55
Node Eviction Rate	node-eviction-rate	<p>This parameter specifies the number of nodes that pods are deleted from per second in a cluster when the AZ is healthy. The default value is <b>0.1</b>, indicating that pods can be evicted from at most one node every 10 seconds.</p> <p><b>NOTE</b> Configure this parameter based on the size of the cluster. The number of pods to be evicted in each batch should not exceed 300.</p> <p>If the parameter is set to a large value, the cluster may be overloaded. Additionally, if too many pods are evicted, they cannot be rescheduled, which will slow down fault recovery.</p>	Default: 0.1

Item	Parameter	Description	Value
Secondary Node Eviction Rate	secondary-node-eviction-rate	<p>This parameter specifies the number of nodes that pods are deleted from per second in a cluster when the AZ is unhealthy. The default value is <b>0.01</b>, indicating that pods can be evicted from at most one node every 100 seconds.</p> <p><b>NOTE</b> Configure this parameter with <b>node-eviction-rate</b> and set it to one-tenth of <b>node-eviction-rate</b>.</p> <p>There is no need to set the parameter to a large value for nodes in an unhealthy AZ, and this configuration may result in overloaded clusters.</p>	Default: 0.01
Large Cluster Threshold	large-cluster-size-threshold	<p>If the number of nodes in a cluster is greater than the value of this parameter, this is a large cluster.</p> <p>Configuration suggestion: For the clusters with a large number of nodes, configure a relatively larger value than the default one for higher performance and faster responses of controllers. Retain the default value for small clusters. Before adjusting the value of this parameter in a production environment, check the impact of the change on cluster performance in a test environment.</p> <p><b>NOTE</b> kube-controller-manager automatically adjusts configurations for large clusters to optimize the cluster performance. Therefore, an excessively small threshold for small clusters will deteriorate the cluster performance.</p>	Default: 50

### Load elastic scaling synchronization cycle

**Table 16-15** Parameters

Item	Parameter	Description	Value
Cluster elastic computing period	horizontal-pod-autoscaler-sync-period	<p>Period for the horizontal pod autoscaler to perform elastic scaling on pods. A smaller value will result in a faster auto scaling response and higher CPU load.</p> <p><b>NOTE</b> Make sure to configure this parameter properly as a lengthy period can cause the controller to respond slowly, while a short period may overload the cluster control plane.</p>	Default: 15s
Horizontal Pod Scaling Tolerance	horizontal-pod-autoscaler-tolerance	<p>The configuration determines how quickly the horizontal pod autoscaler will act to auto scaling policies. If the parameter is set to <b>0</b>, auto scaling will be triggered immediately when the related metrics are met.</p> <p>Configuration suggestion: If the service resource usage increases sharply over time, retain a certain tolerance to prevent auto scaling which is beyond expectation in high resource usage scenarios.</p>	Default: 0.1

Item	Parameter	Description	Value
HPA CPU Initialization Period	horizontal-pod-autoscaler-cpu-initialization-period	<p>During the period specified by this parameter, the CPU usage data used in HPA calculation is limited to pods that are both ready and have recently had their metrics collected. You can use this parameter to filter out unstable CPU usage data during the early stage of pod startup. This helps prevent incorrect scaling decisions based on momentary peak values.</p> <p>Configuration suggestion: If you find that HPA is making incorrect scaling decisions due to CPU usage fluctuations during pod startup, increase the value of this parameter to allow for a buffer period of stable CPU usage.</p> <p><b>NOTE</b></p> <p>Make sure to configure this parameter properly as a small value may trigger unnecessary scaling based on peak CPU usage, while a large value may cause scaling to be delayed.</p> <p>This parameter is available only in clusters of v1.23.16-r0, v1.25.11-r0, v1.27.8-r0, v1.28.6-r0, v1.29.2-r0, or later versions.</p>	Default: 5 minutes

Item	Parameter	Description	Value
HPA Initial Readiness Delay	horizontal-pod-autoscaler-initial-readiness-delay	<p>After CPU initialization, this period allows HPA to use a less strict criterion for getting CPU metrics. During this period, HPA will gather data on the CPU usage of the pod for scaling, regardless of any changes in the pod's readiness status. This parameter ensures continuous tracking of CPU usage, even when the pod status changes frequently.</p> <p>Configuration suggestion: If the readiness status of pods fluctuates after startup and you want to prevent HPA misjudgment caused by the fluctuation, increase the value of this parameter to allow HPA to gather more comprehensive CPU usage data.</p> <p><b>NOTE</b></p> <p>Configure this parameter properly. If it is set to a small value, an unnecessary scale-out may occur due to CPU data fluctuations when the pod is just ready. If it is set to a large value, HPA may not be able to make a quick decision when a rapid response is needed.</p> <p>This parameter is available only in clusters of v1.23.16-r0, v1.25.11-r0, v1.27.8-r0, v1.28.6-r0, v1.29.2-r0, or later versions.</p>	Default: 30s

**Threshold configuration of the number of terminal state pods that trigger recycling**



**Table 16-16** Parameters

Item	Parameter	Description	Value
The maximum number of terminated pods that can be kept before the Pod GC deletes the terminated pod	terminated-pod-gc-threshold	Number of terminated pods that can exist before the terminated pod garbage collector starts deleting terminated pods  <b>NOTE</b> It is recommended that you properly configure this parameter. If the value is too large, there may be a large number of terminated pods in the cluster, which will further affect the performance of list queries and result in an overloaded cluster.	Default: 1000 Value range: 10 to 12500

### Resource quota controller (resource-quota-controller) configuration

 **NOTE**

In high-concurrency scenarios (for example, creating pods in batches), the resource quota management may cause some requests to fail due to conflicts. Do not enable this function unless necessary. To enable this function, ensure that there is a retry mechanism in the request client.

**Table 16-17** Parameters

Item	Parameter	Description	Value
Enable resource quota management	enable-resource-quota	With resource quota management, you are allowed to control the number of workloads (such as Deployments and pods) and the upper limits of resources (such as CPUs and memory) in namespaces or related dimensions. Namespaces control quotas through the ResourceQuota objects.  <ul style="list-style-type: none"> <li>• <b>false</b>: Auto creation is disabled.</li> <li>• <b>true</b>: Auto creation is enabled. For details about the resource quota defaults, see <a href="#">Configuring Resource Quotas</a>.</li> </ul>	Default: false

## 16.7 Heterogeneous Resources

### GPU Settings

#### GPU virtualization

CCE uses the proprietary xGPU virtualization technologies to dynamically divide the GPU memory and compute. A single GPU can be virtualized into up to 20 virtual GPUs. Virtualization is more flexible than static allocation. You can specify the number of GPUs on the basis of stable services to improve GPU utilization. For details, see [Overview](#).

#### GPU Driver

- **Default Cluster Driver:** specifies the default GPU driver version used by the GPU nodes in a cluster. To use a custom driver, enter the download link of the NVIDIA driver. For details, see [Obtaining the Driver Link from Public Network](#).
- **Custom Node Pool Driver:** If you do not want all GPU nodes in a cluster to use the same driver, CCE allows you to install a different GPU driver for each node pool. After you customize a GPU driver for a node pool, nodes in the node pool will preferentially use the custom driver. Nodes for which no driver is specified will use the cluster's default driver.

#### NOTE

- The system installs the driver of the version specified for the node pool. The driver applies only to new nodes in the node pool.
- After the driver version is updated, it takes effect on the nodes newly added to the node pool. Existing nodes must restart to apply the changes.

### NPU Settings

If driver selection is disabled, you cannot specify driver versions or maintain drivers using the CCE AI Suite (Ascend NPU) add-on. When you add an NPU node on the console, the system adds the command to install an NPU driver (version and type decided by the system) and automatically restarts the node after the driver installation is complete. Adding an NPU node in another way, such as using an API, requires you to add the driver installation command to the post-installation command.

Once driver selection is enabled, CCE AI Suite (Ascend NPU) will automatically install the driver during its startup based on the driver configuration of the corresponding model. This makes driver maintenance more flexible. You are advised to use the default driver version. You can also enter the complete driver address to use custom drivers.

# 17 Old Console

---

## 17.1 What Is Cloud Container Engine?

CCE allows you to create highly scalable, high-performance, and enterprise-class Kubernetes clusters and run Docker containers. With CCE, you can easily deploy, manage, and scale containerized applications on HUAWEI CLOUD.

### Defined Terms

This section helps you obtain a deeper understanding of how CCE works.

- **Cluster:** A cluster is a collection of cloud resources required for running containers, such as cloud servers and load balancers.
- **Pod:** A pod consists of one or more related containers that share the same storage and network space.
- **Workload:** A workload is a Kubernetes resource object, which is used to manage the creation and scheduling of pod replicas and automatically control the entire lifecycle of pod replicas.
- **Service:** A Service is an abstraction which defines a logical set of pods and a policy by which to access them (sometimes this pattern is called a microservice).
- **Ingress:** An ingress is a set of rules used to route external HTTP(S) traffic to Services.
- **Helm:** Helm is a package manager for Kubernetes. You can use Helm charts to define, install, and upgrade even the most complex Kubernetes application.
- **Image repository:** An image repository stores Docker images that can be used to deploy containerized services.

For more information, visit <https://kubernetes.io/docs/concepts/>.

### Main Functions

CCE supports full lifecycle management of containerized applications.

#### Cluster management

- You can create Kubernetes clusters with just a few clicks on the HUAWEI CLOUD CCE console. Cross-AZ high-availability is supported.

**One-stop container management**

- Containerized application lifecycle management
- High-performance container networking: tunnel network and VPC network.
- Persistent storage using cloud services, such as Elastic Volume Service (EVS), Scalable File Service (SFS), and Object Storage Service (OBS).
- Multi-dimensional monitoring of resources, applications, and containers
- Diversified logs and statistics
- Role-based access control (RBAC) and container runtime security

**Developer services**

- Open APIs and native APIs from the community
- kubectl-related add-on and native kubectl from the community

## 17.2 High-Risk Operations and Solutions

During service deployment or running, you may trigger high-risk operations at different levels, causing service faults or interruption. To help you better estimate and avoid operation risks, this section introduces the consequences and solutions of high-risk operations from multiple dimensions, such as clusters, nodes, networking, load balancing, logs, and EVS disks.

### Clusters and Nodes

**Table 17-1** High-risk operations and solutions

Category	Operation	Impact	Solution
Master nodes	Modifying the security group of a node in a cluster	The master node may be unavailable. <b>NOTE</b> Naming rule of a master node: <i>Cluster name-cce-control-Random number</i>	Restore the security group by referring to <a href="#">Creating a Cluster</a> and allow traffic from the security group to pass through.
	Letting the node expire or destroying the node	The master node will be unavailable.	This operation cannot be undone.
	Reinstalling the OS	Components on the master node will be deleted.	This operation cannot be undone.
	Upgrading components on the master or etcd node	The cluster may be unavailable.	Roll back to the original version.

Category	Operation	Impact	Solution
	Deleting or formatting core directory data such as <b>/etc/kubernetes</b> on the node	The master node will become unavailable.	This operation cannot be undone.
	Changing the node IP address	The master node will become unavailable.	Change the IP address back to the original one.
	Modifying parameters of core components (such as etcd, kube-apiserver, and docker)	The master node may be unavailable.	Restore the parameter settings to the recommended values. For details, see <a href="#">Configuring Kubernetes Parameters</a> .
	Replacing the master or etcd certificate	The cluster may become unavailable.	This operation cannot be undone.
Worker nodes	Modifying the security group of a node in a cluster	The node may be unavailable. <b>NOTE</b> Naming rule of a worker node: <i>Cluster name-cce-node-Random number</i>	Restore the security group by referring to <a href="#">Creating a Cluster</a> and allow traffic from the security group to pass through.
	Deleting the node	The node will become unavailable.	This operation cannot be undone.
	Reinstalling the OS	Node components are deleted, and the node becomes unavailable.	Reset the node. For details, see <a href="#">Resetting a Node</a> .
	Upgrading the node kernel	The node may be unavailable or the network may be abnormal. <b>NOTE</b> Node running depends on the system kernel version. Do not use <b>yum update</b> to update or reinstall the operating system kernel of a node unless necessary. (Reinstalling the operating system kernel using the original image or other images is a risky operation.)	Reset the node. For details, see <a href="#">Resetting a Node</a> .

Category	Operation	Impact	Solution
	Changing the node IP address	The node will become unavailable.	Change the IP address back to the original one.
	Modifying parameters of core components (such as kubelet and kube-proxy)	The node may become unavailable, and components may be insecure if security-related configurations are modified.	Restore the parameter settings to the recommended values. For details, see <a href="#">Configuring Kubernetes Parameters</a> .
	Modifying OS configuration	The node may be unavailable.	Restore the configuration items or reset the node. For details, see <a href="#">Resetting a Node</a> .
	Deleting the <b>opt</b> directory, <b>/var/</b> <b>paas</b> directory, or a data disk	The node will become unready.	Reset the node. For details, see <a href="#">Resetting a Node</a> .
	Modifying the node directory permission and the container directory permission	The permissions will be abnormal.	You are not advised to modify the permissions. Restore the permissions if they are modified.
	Formatting or partitioning disks on cluster nodes	The node will become unready.	Reset the node. For details, see <a href="#">Resetting a Node</a> .
	Installing other software on nodes	This may cause exceptions on Kubernetes components installed on the node, and make the node unavailable.	Uninstall the software that has been installed and restore or reset the node. For details, see <a href="#">Resetting a Node</a> .

## Networking and Load Balancing

**Table 17-2** High-risk operations and solutions

Operation	Impact	How to Avoid/Fix
Changing the value of the kernel parameter <b>net.ipv4.ip_forward</b> to <b>0</b>	The network becomes inaccessible.	Change the value to <b>1</b> .

Operation	Impact	How to Avoid/Fix
Changing the value of the kernel parameter <b>net.ipv4.tcp_tw_recycle</b> to <b>1</b> .	The NAT service becomes abnormal.	Change the value to <b>0</b> .
Not configuring the node security group to allow UDP packets to pass through port 53 of the container CIDR block	The DNS in the cluster cannot work properly.	Restore the security group by referring to <a href="#">Creating a Cluster</a> and allow traffic from the security group to pass through.
Creating a custom listener on the ELB console for the load balancer managed by CCE	The modified items are reset by CCE or the ingress is faulty.	Use the YAML file of the Service to automatically create a listener.
Binding a user-defined backend on the ELB console to the load balancer managed by CCE.		Do not manually bind any backend.
Changing the ELB certificate on the ELB console for the load balancer managed by CCE.		Use the YAML file of the ingress to automatically manage certificates.
Changing the listener name on the ELB console for the ELB listener managed by CCE.		Do not change the name of the ELB listener managed by CCE.
Changing the description of load balancers, listeners, and forwarding policies managed by CCE on the ELB console.		Do not modify the description of load balancers, listeners, or forwarding policies managed by CCE.
Delete CRD resources of network-attachment-definitions of default-network.		The container network is disconnected, or the cluster fails to be deleted.

## Logs

**Table 17-3** High-risk operations and solutions

Operation	Impact	Solution
Deleting the <code>/tmp/ccs-log-collector/pos</code> directory on the host machine	Logs are collected repeatedly.	None
Deleting the <code>/tmp/ccs-log-collector/buffer</code> directory of the host machine	Logs are lost.	None

## EVS Disks

**Table 17-4** High-risk operations and solutions

Operation	Impact	Solution	Remarks
Manually unmounting an EVS disk on the console	An I/O error is reported when the pod data is being written into the disk.	Delete the mount path from the node and schedule the pod again.	The file in the pod records the location where files are to be collected.
Unmounting the disk mount path on the node	Pod data is written into a local disk.	Remount the corresponding path to the pod.	The buffer contains log cache files to be consumed.
Operating EVS disks on the node	Pod data is written into a local disk.	None	None

## 17.3 Clusters

### 17.3.1 Cluster Overview

**Kubernetes** is a containerized application software system that can be easily deployed and managed. It facilitates container scheduling and orchestration.

For application developers, Kubernetes can be regarded as a cluster operating system. Kubernetes provides functions such as service discovery, scaling, load balancing, self-healing, and even leader election, freeing developers from infrastructure-related configurations.



When using Kubernetes, it is like you run a large number of servers as one on which your applications run. Regardless of the number of servers in a Kubernetes cluster, the method for deploying applications in Kubernetes is always the same.

## Kubernetes Cluster Architecture

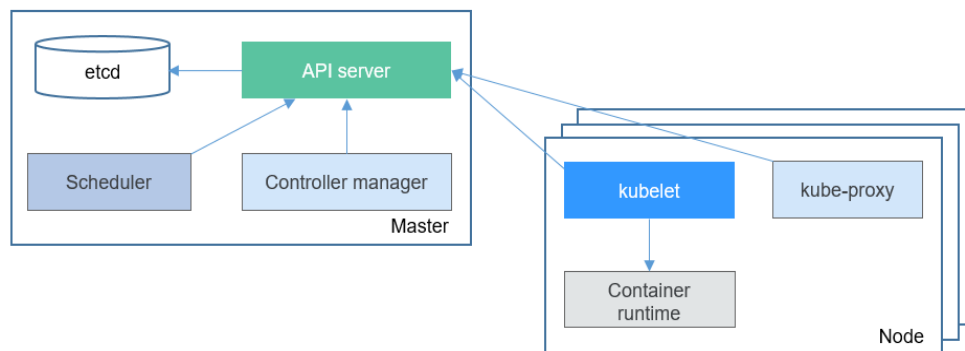
A Kubernetes cluster consists of master nodes (Masters) and worker nodes (Nodes). Applications are deployed on worker nodes, and you can specify the nodes for deployment.

### NOTE

For CCE clusters, master nodes are hosted by CCE. You only need to create worker nodes.

The following figure shows the architecture of a Kubernetes cluster.

**Figure 17-1** Kubernetes cluster architecture



### Master node

A master node is the machine where the control plane components run, including API server, scheduler, controller manager, and etcd.

- API server: a transit station for components to communicate with each other. It receives external requests and writes data into etcd.
- Controller manager: carries out cluster-level functions, such as component replication, node tracing, and node fault fixing.
- Scheduler: schedules containers to nodes based on various conditions (such as available resources and node affinity).
- etcd: provides distributed data storage for cluster configurations.

In a production environment, multiple master nodes are deployed to ensure high cluster availability. For example, you can deploy three master nodes for your CCE cluster.

### Worker node

A worker node is a compute node for running containerized applications in a cluster. A worker node consists of the following components:

- kubelet: communicates with the container runtime, interacts with the API server, and manages containers on the node.

- kube-proxy: an access proxy between application components.
- Container runtime: an engine such as Docker software for downloading images and running containers.

## Number of Master Nodes and Cluster Scale

When you create a cluster on CCE, the number of master nodes can be set to 1 or 3. Three master nodes can be deployed to create a cluster in HA mode.

The master node specifications determine the number of nodes that can be managed by a cluster. When creating a cluster, you can select the cluster management scale, for example, 50 or 200 nodes.

## Cluster Network

From the perspective of the network, all nodes in a cluster are located in a VPC, and containers are running on the nodes. You need to configure node-node, node-container, and container-container communication.

A cluster network can be divided into three network types:

- Node network: IP addresses are assigned to nodes in a cluster.
- Container network: IP addresses are assigned to containers in a cluster for communication between them. Currently, multiple container network models are supported, and each model has its own working mechanism.
- Service network: A service is a Kubernetes object used to access containers. Each Service has a fixed IP address.

When you create a cluster, select a proper CIDR block for each network to ensure that the CIDR blocks do not conflict with each other and each CIDR block has sufficient available IP addresses. **After a cluster is created, the container network model cannot be modified.** Plan the container network model properly before creating a cluster.

You are advised to learn about the cluster network and container network models before creating a cluster. For details, see [Overview](#).

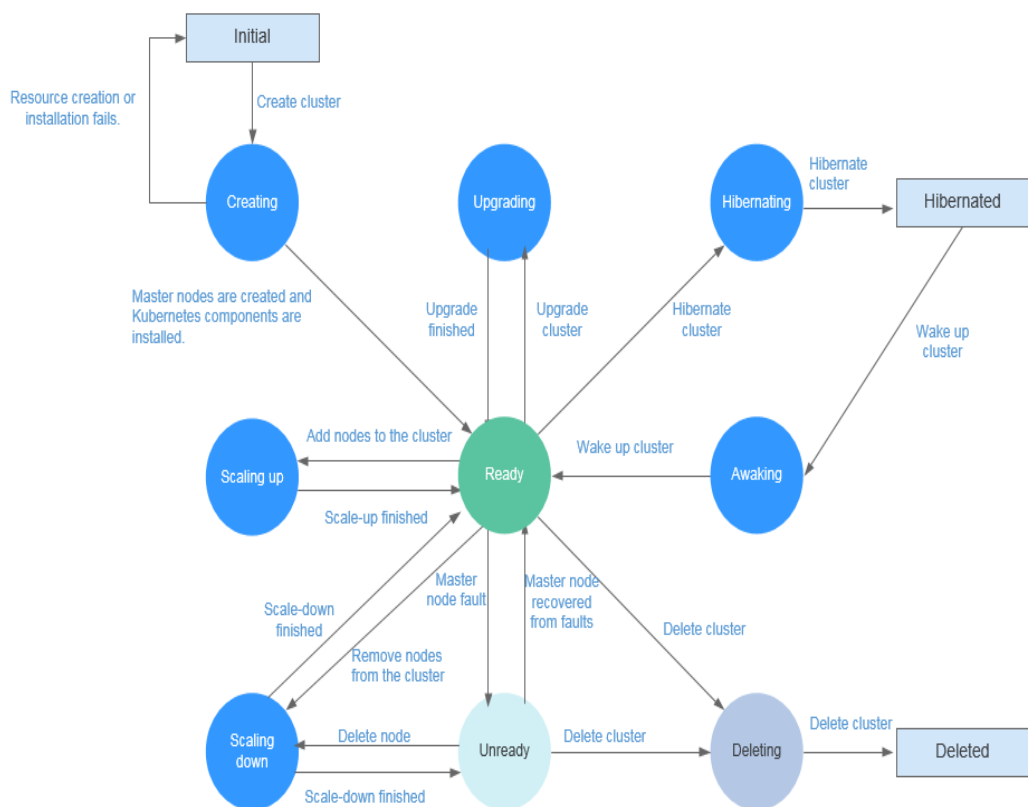
## Cluster Lifecycle

**Table 17-5** Cluster status

Status	Description
Creating	A cluster is being created and is requesting for cloud resources.
Normal	A cluster is running properly.
Scaling-out	A node is being added to a cluster.
Scaling-in	A node is being deleted from a cluster.
Hibernating	A cluster is hibernating.
Awaking	A cluster is being woken up.

Status	Description
Upgrading	A cluster is being upgraded.
Unavailable	A cluster is unavailable.
Deleting	A cluster is being deleted.

Figure 17-2 Cluster status transition



### 17.3.2 Buying a CCE Cluster

On the CCE console, you can easily create Kubernetes clusters. Kubernetes can manage container clusters at scale. A cluster manages a group of node resources.

In CCE, you can create a CCE cluster to manage VMs as nodes. By using high-performance network models, hybrid clusters provide a multi-scenario, secure, and stable runtime environment for containers.

#### Notes and Constraints

- During the node creation, software packages are downloaded from OBS using the domain name. You need to use a private DNS server to resolve the OBS domain name, and configure the subnet where the node resides with a **private DNS server address**. When you create a subnet, the private DNS server is used by default. If you change the subnet DNS, ensure that the DNS server in use can resolve the OBS domain name.

- You can create a maximum of 50 clusters in a single region.
- After a cluster is created, the following items cannot be changed:
  - Number of master nodes in the cluster.
  - AZ of a master node.
  - Network configuration of the cluster, such as the VPC, subnet, container CIDR block, Service CIDR block, and kube-proxy (forwarding) settings.
  - Network model. For example, change the **tunnel network** to the **VPC network**.

For more information, see [Notes and Constraints](#).

## Procedure

- Step 1** Log in to the CCE console. On the **Dashboard** page, click **Buy Cluster**. Alternatively, choose **Resource Management > Clusters** in the navigation pane and click **Buy** next to **CCE Cluster**.
- Step 2** Set cluster parameters by referring to [Table 17-6](#). Pay attention to the parameters marked with an asterisk (\*).

**Table 17-6** Parameters for creating a cluster

Parameter	Description
Billing Mode	<ul style="list-style-type: none"> <li>• <b>Yearly/Monthly</b>: a prepaid billing mode suitable in scenarios where you have a good idea of what resources you will need during the billing period. Fees need to be paid in advance, but services will be less expensive. Yearly/monthly billed clusters cannot be deleted after creation. To stop using these clusters, go to the <a href="#">Billing Center</a> and unsubscribe them.</li> <li>• <b>Pay-per-use</b>: a postpaid billing mode suitable in scenarios where resources will be billed based on usage frequency and duration. You can provision or delete resources at any time.</li> </ul> <p>This section uses the <b>pay-per-use</b> billing mode as an example.</p>
Region	Select a region near you to ensure the lowest latency possible.
Enterprise project	<p>This parameter is displayed only for enterprise users who have enabled the enterprise project function.</p> <p>After an enterprise project (for example, <b>default</b>) is selected, the cluster, nodes in the cluster, cluster security groups, node security groups, and elastic IPs (EIPs) of the automatically created nodes will be created in this enterprise project. After a cluster is created, you are advised not to modify the enterprise projects of nodes, cluster security groups, and node security groups in the cluster.</p> <p>An enterprise project facilitates project-level management and grouping of cloud resources and users.</p>

Parameter	Description
*Cluster Name	<p>Name of the new cluster, which cannot be changed after the cluster is created.</p> <p>A cluster name contains 4 to 128 characters starting with a letter and not ending with a hyphen (-). Only lowercase letters, digits, and hyphens (-) are allowed.</p>
Version	<p>Kubernetes community baseline version. The latest version is recommended.</p> <p>If a <b>Beta</b> version is available, you can use it for trial. However, it is not recommended for commercial use.</p>
Management Scale	<p>Maximum number of worker nodes that can be managed by the master nodes of the current cluster. You can select 50 nodes, 200 nodes, or 1,000 nodes for your cluster, or 2,000 nodes if you are buying a cluster of v1.15.11 or later.</p> <p>If you select <b>1000 nodes</b>, the master nodes of the cluster can manage a maximum of 1000 worker nodes. The configuration fee varies depending on the specifications of master nodes for different management scales.</p>

Parameter	Description
Number of master nodes	<p><b>3:</b> Three master nodes will be created to make the cluster highly available. If a master node is faulty, the cluster can still be available without affecting service functions. Click <b>Change</b>. In the dialog box displayed, you can configure the following parameters:</p> <p><b>Disaster recovery level</b></p> <ul style="list-style-type: none"> <li>• <b>AZ:</b> Master nodes are deployed in different AZs for disaster recovery.</li> <li>• <b>Fault domain:</b> Master nodes are deployed in different failure domains in the same AZ for disaster recovery. This option is displayed only when the environment supports failure domains.</li> <li>• <b>Host computer:</b> Master nodes are deployed on different hosts in the same AZ for disaster recovery.</li> <li>• <b>Customize:</b> You can select different locations to deploy different master nodes. In the fault domain mode, master nodes must be in the same AZ.</li> </ul> <p><b>1:</b> Only one master node is created in the cluster, which cannot ensure SLA for the cluster. Single-master clusters (non-HA clusters) are not recommended for commercial scenarios. Click <b>Change</b>. In the <b>AZ Settings</b> dialog box, select an AZ for the master node.</p> <p><b>NOTE</b></p> <ul style="list-style-type: none"> <li>• You are advised to create multiple master nodes to improve the cluster DR capability in commercial scenarios.</li> <li>• The multi-master mode cannot be changed after the cluster is created. A single-master cluster cannot be upgraded to a multi-master cluster. For a single-master cluster, if a master node is faulty, services will be affected.</li> <li>• To ensure reliability, the multi-master mode is enabled by default for a cluster with 1,000 or more nodes.</li> </ul>
*VPC	<p>VPC where the cluster is located. The value cannot be changed after the cluster is created.</p> <p>A VPC provides a secure and logically isolated network environment.</p> <p>If no VPC is available, click <b>Create a VPC</b> to create a VPC. After the VPC is created, click the refresh icon.</p>

Parameter	Description
*Subnet	<p>Subnet where the node VM runs. The value cannot be changed after the cluster is created.</p> <p>A subnet provides dedicated network resources that are logically isolated from other networks for network security.</p> <p>If no subnet is available, click <b>Create Subnet</b> to create a subnet. After the subnet is created, click the refresh icon. For details about the relationship between VPCs, subnets, and clusters, see <a href="#">Cluster Overview</a>.</p> <p>During the node creation, software packages are downloaded from OBS using the domain name. You need to use a private DNS server to resolve the OBS domain name, and configure the subnet where the node resides with a <a href="#">private DNS server address</a>. When you create a subnet, the private DNS server is used by default. If you change the subnet DNS, ensure that the DNS server in use can resolve the OBS domain name.</p> <p><b>The selected subnet cannot be changed after the cluster is created.</b></p>

Parameter	Description
Network Model	<p>After a cluster is created, the network model cannot be changed. Exercise caution when selecting a network model. For details about how to select a network model, see <a href="#">Overview</a>.</p> <p><b>VPC network</b></p> <p>In this network model, each node occupies one VPC route. The number of VPC routes supported by the current region and the number of container IP addresses that can be allocated to each node (that is, the maximum number of pods that can be created) are displayed on the console.</p> <ul style="list-style-type: none"> <li>• The container network uses VPC routes to integrate with the underlying network. This network model is applicable to performance-intensive scenarios. However, each node occupies one VPC route, and the maximum number of nodes allowed in a cluster depends on the VPC route quota.</li> <li>• Each node is assigned a CIDR block of a fixed size. VPC networks are free from packet encapsulation overheads and outperform container tunnel networks. In addition, as VPC routing includes routes to node IP addresses and the container CIDR block, container pods in the cluster can be directly accessed from outside the cluster.</li> </ul> <p><b>NOTE</b></p> <ul style="list-style-type: none"> <li>- In the VPC network model, extended CIDR blocks and network policies are not supported.</li> <li>- When creating multiple clusters using the VPC network model in one VPC, select a CIDR block for each cluster that does not overlap with the VPC address or other container CIDR blocks.</li> </ul> <p><b>Tunnel network</b></p> <ul style="list-style-type: none"> <li>• 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.</li> <li>• 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.</li> </ul>



Parameter	Description
Container Network Segment	<p>An IP address range that can be allocated to container pods. After the cluster is created, the value cannot be changed.</p> <ul style="list-style-type: none"> <li>• If <b>Automatically select</b> is deselected, enter a CIDR block manually. If the CIDR block you specify conflicts with a subnet CIDR block, the system prompts you to select another CIDR block. The recommended CIDR blocks are 10.0.0.0/8-18, 172.16.0.0/16-18, and 192.168.0.0/16-18. <b>If different clusters share a container CIDR block, an IP address conflict will occur and access to applications may fail.</b></li> <li>• If <b>Automatically select</b> is selected, the system automatically assigns a CIDR block that does not conflict with any subnet CIDR block.</li> </ul> <p>The mask of the container CIDR block must be appropriate. It determines the number of available nodes in a cluster. A too small mask value will cause the cluster to soon fall short of nodes. After the mask is set, the estimated maximum number of containers supported by the current CIDR block will be displayed.</p>
Service Network Segment	<p>An IP address range that can be allocated to Kubernetes Services. After the cluster is created, the value cannot be changed. The Service CIDR block cannot conflict with the created route. If they conflict, select another CIDR block.</p> <ul style="list-style-type: none"> <li>• <b>Default:</b> The default CIDR block 10.247.0.0/16 will be used.</li> <li>• <b>Custom:</b> Manually set a CIDR block and mask based on service requirements. The mask determines the maximum number of Service IP addresses available in the cluster.</li> </ul>
Authorization Mode	<p><b>RBAC</b> is selected by default and cannot be deselected. After RBAC is enabled, IAM users access resources in the cluster according to fine-grained permissions policies. For details, see <a href="#">Namespace Permissions (Kubernetes RBAC-based)</a>.</p>

Parameter	Description
Authentication Mode	<p>The authentication mechanism controls user permission on resources in a cluster.</p> <p>The X.509-based authentication mode is enabled by default. X.509 is a commonly used certificate format.</p> <p>If you want to perform permission control on the cluster, select <b>Enhanced authentication</b>. The cluster will identify users based on the header of the request for authentication.</p> <p>You need to upload your own <b>CA certificate, client certificate, and client certificate private key</b> (for details about how to create a certificate, see <a href="#">Certificates</a>), and select <b>I have confirmed that the uploaded certificates are valid</b>.</p> <p><b>CAUTION</b></p> <ul style="list-style-type: none"> <li>• Upload a file <b>smaller than 1 MB</b>. The CA certificate and client certificate can be in <b>.crt</b> or <b>.cer</b> format. The private key of the client certificate can only be uploaded <b>unencrypted</b>.</li> <li>• The validity period of the client certificate must be longer than five years.</li> <li>• The uploaded CA certificate is used for both the authentication proxy and the kube-apiserver aggregation layer configuration. <b>If the certificate is invalid, the cluster cannot be created.</b></li> </ul>
Cluster Description	Optional. Enter the description of the new container cluster.

Parameter	Description
Advanced Settings	<p>Click <b>Advanced Settings</b> to expand the details page. The following functions are supported (unsupported functions in current AZs are hidden):</p> <p><b>Service Forwarding Mode</b></p> <ul style="list-style-type: none"> <li>• <b>iptables:</b> Traditional kube-proxy uses iptables rules to implement Service load balancing. In this mode, too many iptables rules will be generated when many Services are deployed. In addition, non-incremental updates will cause a latency and even obvious performance issues in the case of heavy service traffic.</li> <li>• <b>ipvs:</b> optimized kube-proxy mode to achieve higher throughput and faster speed, ideal for large-sized clusters. This mode supports incremental updates and can keep connections uninterrupted during Service updates. In this mode, when the ingress and Service use the same ELB instance, the ingress cannot be accessed from the nodes and containers in the cluster.</li> </ul> <p><b>NOTE</b></p> <ul style="list-style-type: none"> <li>• ipvs provides better scalability and performance for large clusters.</li> <li>• Compared with iptables, ipvs supports more complex load balancing algorithms such as least load first (LLF) and weighted least connections (WLC).</li> <li>• ipvs supports server health checking and connection retries.</li> </ul> <p><b>CPU Policy</b></p> <p>This parameter is displayed only for clusters of v1.13.10-r0 and later.</p> <ul style="list-style-type: none"> <li>• <b>On:</b> Exclusive CPU cores can be allocated to workload pods. Select <b>On</b> if your workload is sensitive to latency in CPU cache and scheduling.</li> <li>• <b>Off:</b> Exclusive CPU cores will not be allocated to workload pods. Select <b>Off</b> if you want a large pool of shareable CPU cores.</li> </ul> <p>For details about CPU management policies, see <a href="#">Feature Highlight: CPU Manager</a>.</p> <p>After <b>CPU Policy</b> is enabled, workloads cannot be started or created on nodes after the node specifications are changed.</p>
Validity Period	For a <b>yearly/monthly billed</b> cluster, set the required duration.

**Step 3** Click **Next: Create Node** and set the following parameters.

- **Create Node**
  - **Create now:** Create a node when creating a cluster. Currently, only VM nodes are supported. If a node fails to be created, the cluster will be rolled back.

- **Create later:** No node will be created. Only an empty cluster will be created.
- **Billing Mode:** Select **Yearly/Monthly** or **Pay-per-use**.
  - **Yearly/Monthly:** a prepaid billing mode, in which a resource is billed based on the purchase period. This mode is more cost-effective than the pay-per-use mode and applies if the resource usage period can be estimated.
  - **Pay-per-use:** a postpaid billing mode suitable in scenarios where resources will be billed based on usage duration. You can provision or delete resources at any time.

Nodes created along with the cluster must inherit the billing mode from the cluster. For example, if the billing mode of the cluster is pay-per-use, then nodes created along with the cluster must be billed on the pay-per-use basis. For details, see [Buying a Node](#).

Yearly/monthly billed nodes cannot be deleted after creation. To stop using these nodes, go to the [Billing Center](#) and unsubscribe them.

- **Current Region:** geographic location of the nodes to be created.
- **AZ:** Set this parameter based on the site requirements. An AZ is a physical region where resources use independent power supply and networks. AZs are physically isolated but interconnected through an internal network.

You are advised to deploy worker nodes in different AZs after the cluster is created to make your workloads more reliable. When creating a cluster, you can deploy nodes only in one AZ.

**Figure 17-3** Worker nodes in different AZs

Node Name	Status	Node Pool	Specifications	Allocatable	IP	AZ	Billing Mode	Node Type	Operation
	Available	DefaultPool	2 cores   4 GB   513.large.2	CPU: 1.07 Core Memory: 0.46 GiB	(Private)	AZ2	Pay-per-use Aug 06, 2020 11:40:...	Node created	Monitoring   More
	Available	DefaultPool	2 cores   4 GB   513.large.2	CPU: 0.92 Core Memory: 0.21 GiB	(Private)	AZ3	Pay-per-use Jun 21, 2020 02:14:...	Node created	Monitoring   More
	Available	DefaultPool	2 cores   4 GB   513.large.2	CPU: 0.92 Core Memory: 0.39 GiB	(Private) (EIP)	AZ3	Pay-per-use Jun 20, 2020 17:01:...	Node created	Monitoring   More

- **Node Type**
  - **VM node:** A VM node will be created in the cluster.
- **Node Name:** Enter a node name. A node name contains 1 to 56 characters starting with a lowercase letter and not ending with a hyphen (-). Only lowercase letters, digits, and hyphens (-) are allowed.
- **Specifications:** Select the node specifications based on service requirements. The available node specifications vary depending on AZs.

To ensure node stability, CCE automatically reserves some resources to run necessary system components. For details, see [Formula for Calculating the Reserved Resources of a Node](#).

- **OS:** Select an OS for the node to be created.
  - **Public image:** Select an OS for the node.  
A public image is a standard, widely used image. It contains an OS and preinstalled public applications and is available to all users.
  - **Private image (OBT):** A private image contains an OS or service data, preinstalled public applications, and the owner's private applications. It is

available only to the user who created it. **Private images are supported only for clusters of v1.15 or later.**

If no private image is available, create one by following the instructions provided in .

Reinstalling the OS or modifying OS configurations could make the node unavailable. Exercise caution when performing these operations.

- **System Disk:** Set the system disk space of the worker node. The value ranges from 40GB to 1024 GB. The default value is 40GB.

By default, system disks support High I/O (SAS) and Ultra-high I/O (SSD) EVS disks.

- **Data Disk:** Set the data disk space of the worker node. The value ranges from 100 GB to 32,768 GB. The default value is 100 GB. The EVS disk types provided for the data disk are the same as those for the system disk.

---

 **CAUTION**


If the data disk is uninstalled or damaged, the Docker service becomes abnormal and the node becomes unavailable. You are advised not to delete the data disk.

- 
- **LVM:** If this option is selected, CCE data disks are managed by the Logical Volume Manager (LVM). On this condition, you can adjust the disk space allocation for different resources. This option is selected for the first disk by default and cannot be unselected. You can choose to enable or disable LVM for new data disks.
    - This option is selected by default, indicating that LVM management is enabled.
    - You can deselect the check box to disable LVM management.

---

 **CAUTION**

- Disk space of the data disks managed by LVM will be allocated according to the ratio you set.
  - When creating a node in a cluster of v1.13.10 or later, if LVM is not selected for a data disk, follow instructions in [Adding a Second Data Disk to a Node in a CCE Cluster](#) to fill in the pre-installation script and format the data disk. Otherwise, the data disk will still be managed by LVM.
  - When creating a node in a cluster earlier than v1.13.10, you must format the data disks that are not managed by LVM. Otherwise, either these data disks or the first data disk will be managed by LVM.
- 
- **Add Data Disk:** Currently, a maximum of two data disks can be attached to a node. After the node is created, you can go to the ECS console to attach more data disks. This function is available only to clusters of certain versions.

- **Data disk space allocation:** Click  [Change Configuration](#) to specify the resource ratio for **Kubernetes Space** and **User Space**. Disk space of the data disks managed by LVM will be allocated according to the ratio you set. This function is available only to clusters of certain versions.
  - **Kubernetes Space:** You can specify the ratio of the data disk space for storing Docker and kubelet resources. Docker resources include the Docker working directory, Docker images, and image metadata. kubelet resources include pod configuration files, secrets, and emptyDirs.

The Docker space cannot be less than 10%, and the space size cannot be less than 60 GB. The kubelet space cannot be less than 10%.


The Docker space size is determined by your service requirements. For details, see [Data Disk Space Allocation](#).
  - **User Space:** You can set the ratio of the disk space that is not allocated to Kubernetes resources and the path to which the user space is mounted.

 **NOTE**

Note that the mount path cannot be `/`, `/home/paas`, `/var/paas`, `/var/lib`, `/var/script`, `/var/log`, `/mnt/paas`, or `/opt/cloud`, and cannot conflict with the system directories (such as `bin`, `lib`, `home`, `root`, `boot`, `dev`, `etc`, `lost+found`, `mnt`, `proc`, `sbin`, `srv`, `tmp`, `var`, `media`, `opt`, `selinux`, `sys`, and `usr`). Otherwise, the system or node installation will fail.

---

**NOTICE**

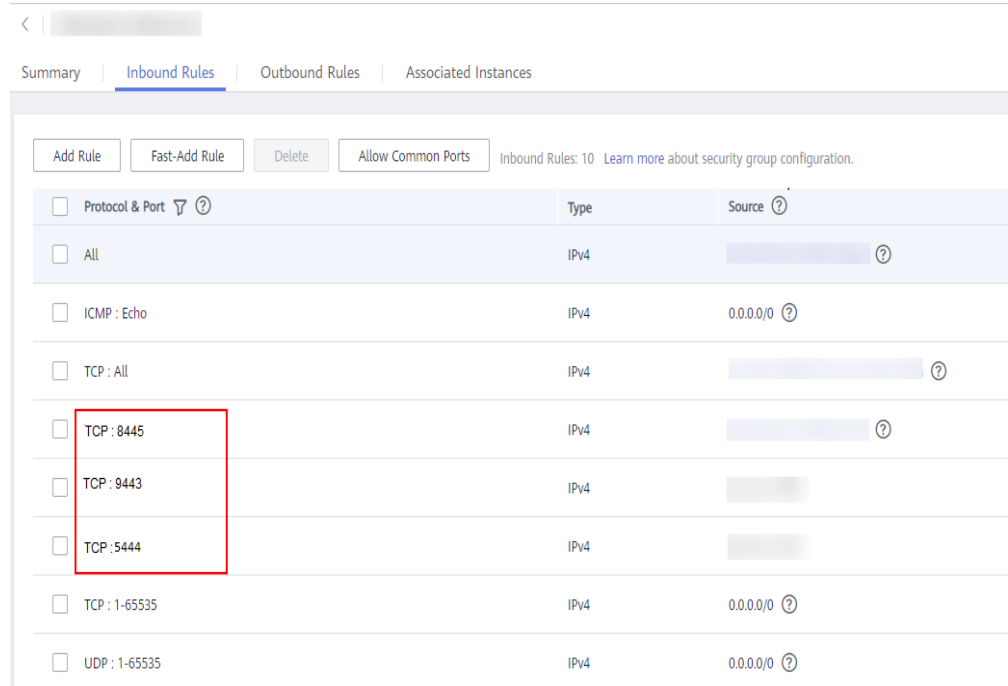
- The ratio of disk space allocated to the Kubernetes space and user space must be equal to 100% in total. You can click  to refresh the data after you have modified the ratio.
- By default, disks run in the direct-lvm mode. If data disks are removed, the loop-lvm mode will be used and this will impair system stability.

- **VPC:** A VPC where the current cluster is located. This parameter cannot be changed and is displayed only for clusters of v1.13.10-r0 or later.
- **Subnet:** A subnet improves network security by providing exclusive network resources that are isolated from other networks. You can select any subnet in the cluster VPC. Cluster nodes can belong to different subnets.

During the node creation, software packages are downloaded from OBS using the domain name. You need to use a private DNS server to resolve the OBS domain name, and configure the subnet where the node resides with a [private DNS server address](#). When you create a subnet, the private DNS server is used by default. If you change the subnet DNS, ensure that the DNS server in use can resolve the OBS domain name.

When a node is added to an existing cluster, if an extended CIDR block is added to the VPC corresponding to the subnet and the subnet is an extended CIDR block, you need to add the following three security group rules to the master node security group (the group name is in the format of **Cluster**

*name-cce-control-Random number*). These rules ensure that the nodes added to the cluster are available. (This step is not required if an extended CIDR block has been added to the VPC during cluster creation.)



- **EIP:** an independent public IP address. If the nodes to be created require public network access, select **Automatically assign** or **Use existing**.  
An EIP bound to the node allows public network access. EIP bandwidth can be modified at any time. An ECS without a bound EIP cannot access the Internet or be accessed by public networks.
  - **Do not use:** A node without an EIP cannot be accessed from public networks. It can be used only as a cloud server for deploying services or clusters on a private network.
  - **Automatically assign:** An EIP with specified configurations is automatically assigned to each node. If the number of EIPs is smaller than the number of nodes, the EIPs are randomly bound to the nodes. Configure the EIP specifications, billing factor, bandwidth type, and bandwidth size as required. When creating an ECS, ensure that the elastic IP address quota is sufficient.
  - **Use existing:** Existing EIPs are assigned to the nodes to be created.

**NOTE**

By default, VPC's SNAT feature is disabled for CCE. If SNAT is enabled, you do not need to use EIPs to access public networks. For details about SNAT, see [Custom Policies](#).

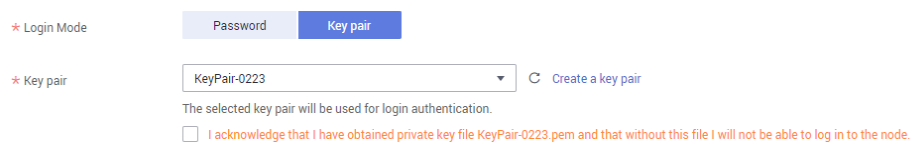
- **Login Mode:** You can use a password or key pair.
  - **Password:** The default username is **root**. Enter the password for logging in to the node and confirm the password.  
Be sure to remember the password as you will need it when you log in to the node.
  - **Key pair:** Select the key pair used to log in to the node. You can select a shared key.

A key pair is used for identity authentication when you remotely log in to a node. If no key pair is available, click **Create a key pair**.

### NOTICE

When creating a node using a key pair, IAM users can select only the key pairs created by their own, regardless of whether these users are in the same group. For example, user B cannot use the key pair created by user A to create a node, and the key pair is not displayed in the drop-down list on the CCE console.

**Figure 17-4** Key pair



- **Advanced ECS Settings** (optional): Click ^ to show advanced ECS settings.
  - **ECS Group**: An ECS group logically groups ECSs. The ECSs in the same ECS group comply with the same policy associated with the ECS group.
    - **Anti-affinity**: ECSs in an ECS group are deployed on different physical hosts to improve service reliability.
  - Select an existing ECS group, or click **Create ECS Group** to create one. After the ECS group is created, click the refresh button.
  - **Resource Tags**: By adding tags to resources, you can classify resources. You can create predefined tags in Tag Management Service (TMS). Predefined tags are visible to all service resources that support the tagging function. You can use predefined tags to improve tag creation and migration efficiency.

CCE will automatically create the "CCE-Dynamic-Provisioning-Node=node id" tag. A maximum of 5 tags can be added.
  - **Agency**: An agency is created by a tenant administrator on the IAM console. By creating an agency, you can share your cloud server resources with another account, or entrust a more professional person or team to manage your resources. To authorize an ECS or BMS to call cloud services, select **Cloud service** as the agency type, click **Select**, and then select **ECS BMS**.
  - **Pre-installation Script**: Enter a maximum of 1,000 characters.

The script will be executed before Kubernetes software is installed. Note that if the script is incorrect, Kubernetes software may fail to be installed. The script is usually used to format data disks.
  - **Post-installation Script**: Enter a maximum of 1,000 characters.


The script will be executed after Kubernetes software is installed and will not affect the installation. The script is usually used to modify Docker parameters.




- **Subnet IP Address:** Select **Automatically assign IP address** (recommended) or **Manually assigning IP addresses**.

 NOTE

When you **manually assign IPs**, the master IP is randomly specified. Therefore, it may conflict with the worker node IP. If you prefer the manual operation, you are advised to select a subnet CIDR block different from that of the master node when setting worker node **subnet**.

- **Advanced Kubernetes Settings:** (Optional) Click  to show advanced cluster settings.
  - **Max Pods:** maximum number of pods that can be created on a node, including the system's default pods. If the cluster uses the **VPC network model**, the maximum value is determined by the number of IP addresses that can be allocated to containers on each node.

This limit prevents the node from being overloaded by managing too many pods. For details, see [Maximum Number of Pods That Can Be Created on a Node](#).
  - **Maximum Data Space per Container:** maximum data space that can be used by a container. The value ranges from 10 GB to 500 GB. If the value of this field is larger than the data disk space allocated to Docker resources, the latter will override the value specified here. Typically, 90% of the data disk space is allocated to Docker resources. This parameter is displayed only for clusters of v1.13.10-r0 and later.
- **Nodes:** The value cannot exceed the management scale you select when configuring cluster parameters. Set this parameter based on service requirements and the remaining quota displayed on the page. Click  to view the factors that affect the number of nodes to be added (depending on the factor with the minimum value).
- **Validity Period:** If the cluster billing mode is **yearly/monthly**, set the number of months or years for which you will use the new node.

**Step 4** Click **Next: Install Add-on**, and select the add-ons to be installed in the **Install Add-on** step.

System resource add-ons must be installed. Advanced functional add-ons are optional.

You can also install all add-ons after the cluster is created. To do so, choose **Add-ons** in the navigation pane of the CCE console and select the add-on you will install. For details, see [Add-ons](#).

**Step 5** Click **Next: Confirm**. Read the **product instructions** and select **I am aware of the above limitations**. Confirm the configured parameters, specifications, and fees.

**Step 6** Click **Submit**.

If the cluster is billed on a yearly/monthly basis, click **Pay Now** and follow on-screen prompts to pay the order.

It takes about 6 to 10 minutes to create a cluster. You can click **Back to Cluster List** to perform other operations on the cluster or click **Go to Cluster Events** to view the cluster details. If the cluster status is Available, the cluster is successfully created.

----End

## Related Operations

- After creating a cluster, you can use the Kubernetes command line (CLI) tool `kubectl` to connect to the cluster. For details, see [Connecting to a Cluster Using kubectl](#).
- Add more nodes to the cluster. For details, see [Buying a Node](#).
- Log in to a node. For details, see [Logging In to a Node](#).
- Create a namespace. You can create multiple namespaces in a cluster and organize resources in the cluster into different namespaces. These namespaces serve as logical groups and can be managed separately. For more information about how to create a namespace for a cluster, see [Namespaces](#).
- Create a workload. Once the cluster is created, you can use an image to create an application that can be accessed from public networks. For details, see [Creating a Deployment](#) or [Creating a StatefulSet](#).
- Click the cluster name to view cluster details.

**Table 17-7** Cluster details

Tab	Description
Cluster Details	View the details and operating status of the cluster.
Monitoring	You can view the CPU and memory allocation rates of all nodes in the cluster (that is, the maximum allocated amount), as well as the CPU usage, memory usage, and specifications of the master node(s).
Events	<ul style="list-style-type: none"> <li>• View cluster events on the <b>Events</b> tab page.</li> <li>• Set search criteria. For example, you can set the time segment or enter an event name to view corresponding events.</li> </ul>
Auto Scaling	<p>You can configure auto scaling to add or reduce worker nodes in a cluster to meet service requirements. For details, see <a href="#">Setting Cluster Auto Scaling</a>.</p> <p>Clusters of v1.17 do not support auto scaling using AOM. You can use node pools for auto scaling. For details, see <a href="#">Node Pool Overview</a>.</p>
kubectl	To access a Kubernetes cluster from a PC, you need to use the Kubernetes command line tool <code>kubectl</code> . For details, see <a href="#">Connecting to a Cluster Using kubectl</a> .

### 17.3.3 Using kubectl to Run a Cluster

### 17.3.3.1 Connecting to a Cluster Using kubectl

#### Scenario

This section uses a CCE cluster as an example to describe how to connect to a CCE cluster using kubectl.

#### Permission Description

When you access a cluster using kubectl, CCE uses the **kubeconfig.json** file generated on the cluster for authentication. This file contains user information, based on which CCE determines which Kubernetes resources can be accessed by kubectl. The permissions recorded in a **kubeconfig.json** file vary from user to user.

For details about user permissions, see [Cluster Permissions \(IAM-based\) and Namespace Permissions \(Kubernetes RBAC-based\)](#).

#### Using kubectl

##### Background

To connect a client to a Kubernetes cluster, you can use kubectl. For details, see [Install Tools](#).

##### Prerequisites

CCE allows you to access a cluster through a **VPC network** or a **public network**.

- VPC internal access: Clusters in the same VPC can access each other.
- Public network access: You need to prepare an ECS that can connect to a public network.

---

#### NOTICE

If public network access is used, the kube-apiserver of the cluster will be exposed to the public network and may be attacked. You are advised to configure Advanced Anti-DDoS for the EIP of the node where the kube-apiserver is located.

---

##### Downloading kubectl

You need to download kubectl and configuration file, copy the file to your client, and configure kubectl. After the configuration is complete, you can use kubectl to access your Kubernetes clusters.

On the [Kubernetes release](#) page, click the corresponding link based on the cluster version, click **Client Binaries**, and download the corresponding platform software package.

**Figure 17-5** Downloading kubectl v1.21.14

Downloads for v1.21.14

Source Code

filename	sha512 hash
<a href="#">kubernetes.tar.gz</a>	d1f2bdf2e7f18fe019156955b5361d6ea1ebb92039696d20d475dc7bb4ae0a74058d4fd64a5aa783d6332d16ca19196c71f8f
<a href="#">kubernetes-src.tar.gz</a>	47cfe98ce5189e70e9147a8df9cd6b2cb3d3048ec0b91843dd2bc21506f662a42ec26eb7f7de671b863f9bd5ca2835524527c

Client Binaries

filename	sha512 hash
<a href="#">kubernetes-client-darwin-amd64.tar.gz</a>	a59b3d33bf77f196b351b1582ef9c5293dba8a6b358b8f75a4dc857e5ce39d106d4f4435c80ce71e1f60b4c0ed361c444ff81836f
<a href="#">kubernetes-client-darwin-arm64.tar.gz</a>	a1cd396ef0b0a6881cc68608c61c1736629326e80dc709d21745783295d06bebeb4239ac7d8fc0aad2dfad6ade3908340803fb0
<a href="#">kubernetes-client-linux-386.tar.gz</a>	da2a71c57f6dda6ca7d2b6ee7c243b791d2f9e514a96b4b35c528c873d2e7839fc24409680107808816b269d38142eebf3db1d2

**Installing and configuring kubectl**

**Step 1** Log in to the CCE console, click **Resource Management > Clusters**, and choose **Command Line Tool > Kubectl** under the cluster to be connected.

**Step 2** On the **Kubectl** tab page of the cluster details page, connect to the cluster as prompted.

**NOTE**

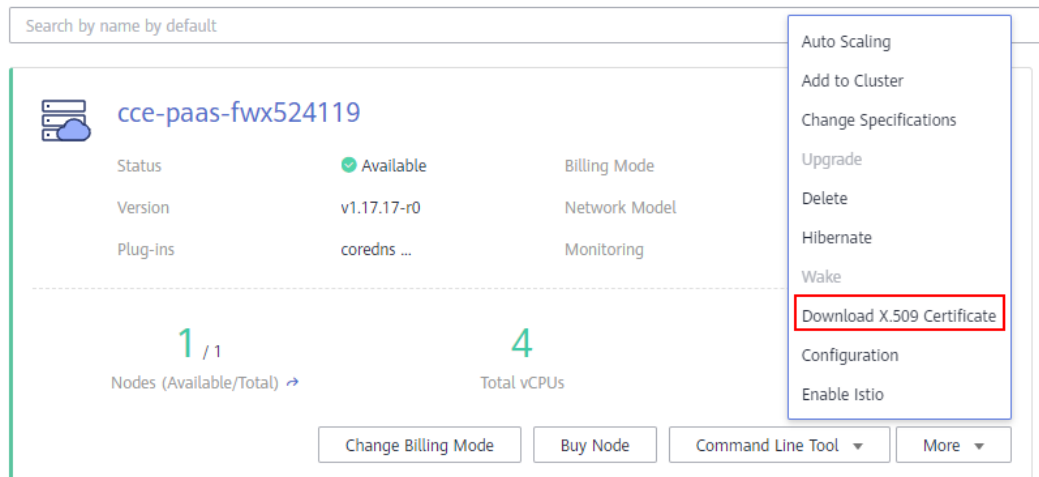
- You can download the kubectl configuration file (**kubeconfig.json**) on the **kubectl** tab page. This file is used for user cluster authentication. If the file is leaked, your clusters may be attacked.
- If two-way authentication is enabled for the current cluster and an EIP has been bound to the cluster, when the authentication fails (x509: certificate is valid), you need to bind the EIP and download the **kubeconfig.json** file again.
- The Kubernetes permissions assigned by the configuration file downloaded by IAM users are the same as those assigned to the IAM users on the CCE console.
- If the KUBECONFIG environment variable is configured in the Linux OS, kubectl preferentially loads the KUBECONFIG environment variable instead of **\$home/.kube/config**.

----End

**Calling Kubernetes Native APIs Through the API Server**

You can use the API server of a Kubernetes cluster to call Kubernetes native APIs.

**Step 1** Log in to the CCE console. In the navigation pane, choose **Resource Management > Clusters**. Choose **More > Download X.509 Certificate** for the cluster to call APIs.



Download the following certificates:

- ca.crt
- client.crt
- client.key

**Step 2** On the cluster details page, obtain the API server address, as shown in the following figure.

### Network

Network Model	VPC network
VPC	vpc-ss
Subnet	subnet-ss
Service Forwarding Mode	iptables
Service Network Segment	10.247.0.0/16
Container Network Segment	172.16.0.0/16
Internal API Server Address	https://192.168.0.188:5443
Public API Server Address	Bind EIP

With the certificates and API server address, you can call Kubernetes native APIs.

For example, if you run the **curl** command to call the API to view the pod information, you only need to carry the certificate in the command as follows:

```
curl --cert ./client.crt --key ./client.key https://192.168.0.198:5443/api/v1/namespaces/default/pods/
```

----End

## Common Issue (Error from server Forbidden)

When you use kubectl to create or query Kubernetes resources, the following output is returned:

```
# 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"
```

The cause is that the user does not have the permissions to operate the Kubernetes resources. For details about how to assign permissions, see [Namespace Permissions \(Kubernetes RBAC-based\)](#).

## 17.3.3.2 Common kubectl Commands

### Getting Started

#### get

The **get** command displays one or many resources of a cluster.

This command prints a table of the most important information about all resources, including cluster nodes, running pods, Deployments, and Services.

---

#### NOTICE

A cluster can have multiple namespaces. If no namespace is specified, this command will run with the **--namespace=default** flag.

---

Examples:

To list all pods with detailed information:

```
kubectl get po -o wide
```

To display pods in all namespaces:

```
kubectl get po --all-namespaces
```

To list labels of pods in all namespaces:

```
kubectl get po --show-labels
```

To list all namespaces of the node:

```
kubectl get namespace
```

#### NOTE

To list information of other nodes, run this command with the **-s** flag. To list a specified type of resources, add the resource type to this command, for example, **kubectl get svc**, **kubectl get nodes**, and **kubectl get deploy**.

To list a pod with a specified name in YAML output format:

```
kubectl get po <podname> -o yaml
```

To list a pod with a specified name in JSON output format:

```
kubectl get po <podname> -o json
kubectl get po rc-nginx-2-btv4j -o=custom-columns=LABELS:metadata.labels.app
```

#### NOTE

**LABELS** indicates a comma separated list of user-defined column titles.  
**metadata.labels.app** indicates the data to be listed in either YAML or JSON output format.

## create

The **create** command creates a cluster resource from a file or input.

If there is already a resource descriptor (a YAML or JSON file), you can create the resource from the file by running the following command:

```
kubectl create -f filename
```

## expose

The **expose** command exposes a resource as a new Kubernetes service. Possible resources include a pod, Service, and Deployment.

```
kubectl expose deployment deployname --port=81 --type=NodePort --target-port=80 --name=service-name
```

### NOTE

The example command creates a service of NodePort type for the deployment with the name specified in **deployname**. The service will serve on port 81 specified in **-port** and connect to the containers on port 80 specified in **-target-port**. More specifically, the service is reachable at <cluster-internal IP address>:<port>, and containers are reachable at <node IP address>:<target-port>.

## run

Examples:

To run a particular image in the cluster:

```
kubectl run deployname --image=nginx:latest
```

To run a particular image using a specified command:

```
kubectl run deployname -image=busybox --command -- ping baidu.com
```

## set

The **set** command configures object resources.

Example:

To change the image of a deployment with the name specified in **deployname** to image 1.0:

```
kubectl set image deploy deployname containername=containername:1.0
```

## edit

The **edit** command edits a resource from the default editor.

Examples:

To update a pod:

```
kubectl edit po po-nginx-btv4j
```

The example command yields the same effect as the following command:

```
kubectl get po po-nginx-btv4j -o yaml >> /tmp/nginx-tmp.yaml  
vim /tmp/nginx-tmp.yaml  
/*do some changes here */  
kubectl replace -f /tmp/nginx-tmp.yaml
```

## explain

The **explain** command views documents or reference documents.

Example:

To get documentation of pods:

```
kubectl explain pod
```

### **delete**

The **delete** command deletes resources by resource name or label.

Example:

To delete a pod with minimal delay:

```
kubectl delete po podname --now  
kubectl delete -f nginx.yaml  
kubectl delete deployment deployname
```

## Deployment Commands

### **rolling-update\***

**rolling-update** is a very important command. It updates a running service with zero downtime. Pods are incrementally replaced by new ones. One pod is updated at a time. The old pod is deleted only after the new pod is up. New pods must be distinct from old pods by name, version, and label. Otherwise, an error message will be reported.

```
kubectl rolling-update poname -f newfilename  
kubectl rolling-update poname -image=image:v2
```

If any problem occurs during the rolling update, run the command with the **-rollback** flag to abort the rolling update and revert to the previous pod.

```
kubectl rolling-update poname -rollback
```

### **rollout**

The **rollout** command manages the rollout of a resource.

Examples:

To check the rollout status of a particular deployment:

```
kubectl rollout status deployment/deployname
```

To view the rollout history of a particular deployment:

```
kubectl rollout history deployment/deployname
```

To roll back to the previous deployment: (by default, a resource is rolled back to the previous version)

```
kubectl rollout undo deployment/test-nginx
```

### **scale**

The **scale** command sets a new size for a resource by adjusting the number of resource replicas.

```
kubectl scale deployment deployname --replicas=newnumber
```

### **autoscale**

The **autoscale** command automatically chooses and sets the number of pods. This command specifies the range for the number of pod replicas maintained by a



replication controller. If there are too many pods, the replication controller terminates the extra pods. If there is too few, the replication controller starts more pods.

```
kubectl autoscale deployment deployname --min=minnumber --max=maxnumber
```

## Cluster Management Commands

### **cordons, drain, uncordon\***

If a node to be upgraded is running many pods or is already down, perform the following steps to prepare the node for maintenance:

- Step 1** Run the **cordons** command to mark a node as unschedulable. This means that new pods will not be scheduled onto the node.

```
kubectl cordon nodename
```

Note: In CCE, **nodename** indicates the private network IP address of a node.

- Step 2** Run the **drain** command to smoothly migrate the running pods from the node to another node.

```
kubectl drain nodename --ignore-daemonsets --ignore-emptydir
```

**ignore-emptydir** ignores the pods that use emptyDirs.

- Step 3** Perform maintenance operations on the node, such as upgrading the kernel and upgrading Docker.

- Step 4** After node maintenance is completed, run the **uncordon** command to mark the node as schedulable.

```
kubectl uncordon nodename
```

----End

### **cluster-info**

To display the add-ons running in the cluster:

```
kubectl cluster-info
```

To dump current cluster information to stdout:

```
kubectl cluster-info dump
```

### **top\***

The **top** command displays resource (CPU/memory/storage) usage. This command requires Heapster to be correctly configured and working on the server.

### **taint\***

The **taint** command updates the taints on one or more nodes.

### **certificate\***

The **certificate** command modifies the certificate resources.

## Fault Diagnosis and Debugging Commands

### **describe**

The **describe** command is similar to the **get** command. The difference is that the **describe** command shows details of a specific resource or group of resources, whereas the **get** command lists one or more resources in a cluster. The **describe** command does not support the **-o** flag. For resources of the same type, resource details are printed out in the same format.

#### NOTE

If the information about a resource is queried, you can use the **get** command to obtain more detailed information. If you want to check the status of a specific resource, for example, to check if a pod is in the running state, run the **describe** command to show more detailed status information.

```
kubectl describe po <podname>
```

### logs

The **logs** command prints logs for a container in a pod or specified resource to stdout. To display logs in the **tail -f** mode, run this command with the **-f** flag.

```
kubectl logs -f podname
```

### exec

The **kubectl exec** command is similar to the Docker **exec** command and executes a command in a container. If there are multiple containers in a pod, use the **-c** flag to choose a container.

```
kubectl exec -it podname bash  
kubectl exec -it podname -c containername bash
```

### port-forward\*

The **port-forward** command forwards one or more local ports to a pod.

Example:

To listen on ports 5000 and 6000 locally, forwarding data to/from ports 5000 and 6000 in the pod:

```
kubectl port -forward podname 5000:6000
```

### proxy\*

The **proxy** command creates a proxy server between localhost and the Kubernetes API server.

Example:

To enable the HTTP REST APIs on the master node:

```
kubectl proxy -accept-hosts= '*' -port=8001 -address= '0.0.0.0'
```

### cp

The **cp** command copies files and directories to and from containers.

```
cp filename newfilename
```

### auth\*

The **auth** command inspects authorization.

### attach\*

The **attach** command is similar to the **logs -f** command and attaches to a process that is already running inside an existing container. To exit, run the **ctrl-c** command. If a pod contains multiple containers, to view the output of a specific container, use the **-c** flag and *containername* following *podname* to specify a container.

```
kubectl attach podname -c containername
```

## Advanced Commands

### replace

The **replace** command updates or replaces an existing resource by attributes including the number of replicas, labels, image versions, and ports. You can directly modify the original YAML file and then run the **replace** command.

```
kubectl replace -f filename
```

#### NOTICE

Resource names cannot be updated.

### apply\*

The **apply** command provides a more strict control on resource updating than **patch** and **edit** commands. The **apply** command applies a configuration to a resource and maintains a set of configuration files in source control. Whenever there is an update, the configuration file is pushed to the server, and then the **kubectl apply** command applies the latest configuration to the resource. The Kubernetes compares the new configuration file with the original one and updates only the changed configuration instead of the whole file. The configuration that is not contained in the **-f** flag will remain unchanged. Unlike the **replace** command which deletes the resource and creates a new one, the **apply** command directly updates the original resource. Similar to the git operation, the **apply** command adds an annotation to the resource to mark the current apply.

```
kubectl apply -f
```

### patch

If you want to modify attributes of a running container without first deleting the container or using the **replace** command, the **patch** command is to the rescue. The **patch** command updates field(s) of a resource using strategic merge patch, a JSON merge patch, or a JSON patch. For example, to change a pod label from **app=nginx1** to **app=nginx2** while the pod is running, use the following command:

```
kubectl patch pod podname -p '{"metadata":{"labels":{"app":"nginx2"}}}'
```

### convert\*

The **convert** command converts configuration files between different API versions.

## Configuration Commands

### label

The **label** command update labels on a resource.

```
kubectl label pods my-pod new-label=newlabel
```

### **annotate**

The **annotate** command update annotations on a resource.

```
kubectl annotate pods my-pod icon-url=http://.....
```

### **completion**

The **completion** command provides autocompletion for shell.

## **Other Commands**

### **api-versions**

The **api-versions** command prints the supported API versions.

```
kubectl api-versions
```

### **api-resources**

The **api-resources** command prints the supported API resources.

```
kubectl api-resources
```

### **config\***

The **config** command modifies kubeconfig files. An example use case of this command is to configure authentication information in API calls.

### **help**

The **help** command gets all command references.

### **version**

The **version** command prints the client and server version information for the current context.

```
kubectl version
```

## **17.3.3.3 kubectl Usage Guide**

Before running kubectl commands, you should have the kubectl development skills and understand the kubectl operations. For details, see [Kubernetes API](#) and [kubectl CLI](#).

Go to the [Kubernetes release page](#) to download kubectl corresponding to the cluster version or a later version.

## **Cluster Connection**

- [Connecting to a Kubernetes cluster using kubectl](#)

## **Workload Creation**

- [Creating a Deployment using kubectl](#)
- [Creating a StatefulSet using kubectl](#)

## Workload Affinity/Anti-affinity Scheduling

- [Example YAML for workload-node affinity](#)
- [Example YAML for workload-node anti-affinity](#)
- [Example YAML for workload-workload affinity](#)
- [Example YAML for workload-workload anti-affinity](#)
- [Example YAML for workload-AZ affinity](#)
- [Example YAML for workload-AZ anti-affinity](#)

## Workload Access Mode Settings

- [Implementing intra-cluster access using kubectl](#)
- [Implementing node access using kubectl](#)
- [Implementing Layer 4 load balancing using kubectl](#)
- [Implementing Layer 7 load balancing using kubectl](#)

## Advanced Workload Settings

- [Example YAML for setting the container lifecycle](#)

## Job Management

- [Creating a job using kubectl](#)
- [Creating a cron job using kubectl](#)

## Configuration Center

- [Creating a ConfigMap using kubectl](#)
- [Creating a secret using kubectl](#)

## Storage Management

- [Creating a PV using kubectl](#)
- [Creating a PVC using kubectl](#)

## 17.3.4 Setting Cluster Auto Scaling

### Scenario

The Cluster Auto Scaling feature allows CCE to automatically scale out a cluster (adding worker nodes to a cluster) according to custom policies when workloads cannot be scheduled into the cluster due to insufficient cluster resources.

### Notes and Constraints

- Currently, master nodes cannot be automatically added to or removed from clusters.
- If both auto scale-in and auto scale-out are required, use the autoscaler add-on. For details, see [autoscaler](#).
- Clusters of v1.17 do not support auto scaling using AOM. You can use node pools for auto scaling. For details, see [Node Pool Overview](#).

## Automatic Cluster Scale-out

- Step 1** Log in to the CCE console. Choose **Resource Management > Clusters** in the navigation pane. In the card view of the cluster to be scaled, choose **More > Auto Scaling**.
- Step 2** Click the **Scale-out Settings** tab and then **Edit**. Set the maximum number of nodes, minimum number of nodes, cooldown period, and node configuration.

**Table 17-8** Scale-out settings

Parameter	Description
Cooldown Period	Interval between consecutive scale-out operations, in the unit of second. The cooldown period ensures that a scale-out operation is initiated only when previous scaling operation is finished and the system is running stably.  The value ranges from 60 to 3600, in seconds. The default value is 900. If the cooling interval is less than 900 seconds (15 minutes), the auto scaling may not work well, because creating a node may take 2 to 10 minutes.
Maximum Nodes	Maximum number of nodes to which the cluster can scale out.  $1 \leq \text{Maximum Nodes} < \text{cluster node quota}$ <b>NOTE</b> The cluster node quota depends on the cluster size (maximum number of nodes that can be managed by a cluster) and the node quota of the account. The cluster node quota used here is the smaller of the two.
Node Configuration	If scale-out is required after the scale-out policy is executed, the system creates a node.  1. Click <b>Set</b> and set the node parameters. For details about how to set the node parameters, see <a href="#">Buying a Node</a> .  2. After the parameters are configured, click <b>Submit</b> .

- Step 3** After confirming the scale-out configuration and node parameters, click **OK**.
- Step 4** Set the scale-out policy for the cluster. Click the **Scale-out Policies** tab and click **Add Policy**.
- **Policy Name:** Enter a policy name, for example, **policy01**.
  - **Policy Type:** Currently, the following types of auto scale-out policies are supported:
    - **Metric-based policy:** Scale-out is performed based on the CPU or memory settings.

**Table 17-9** Parameters for adding a metric-based policy

Parameter	Description
*Metric	Select <b>Allocated CPU</b> or <b>Allocated Memory</b> .
*Trigger Condition	Set a condition for triggering a scale-out policy, that is, when the average CPU or memory allocation value is greater than or less than a specified percentage.
*Monitoring Window	Size of the data aggregation window. Select a value from the drop-down list. If you select <b>15min</b> , the selected metric is measured every 15 minutes.
*Threshold Crossings	Number of consecutive times that the threshold is reached within the monitoring window. The calculation cycle is fixed at one minute. If you set this parameter to <b>3</b> , the configured action will be triggered when the metrics meet the specified threshold for three consecutive times.
*Action	Action executed after a policy is triggered.

- **Scheduled policy:** Scale-out is performed at a specified time.

**Table 17-10** Parameters for adding a scheduled policy

Parameter	Description
*Policy Type	Set this parameter to <b>Scheduled policy</b> .
*Trigger Time	Time at which the policy is triggered.
*Action	Action executed after a policy is triggered.

- **Periodic policy:** Scale-out can be performed by day, week, or month.

**Table 17-11** Parameters for adding a periodic policy

Parameter	Description
*Policy Type	Set the parameter to <b>Periodic policy</b> .
*Time Range	Specify the time for triggering the policy.
*Action	Action executed after a policy is triggered.

**Step 5** Click **OK**.

After the auto scale-out is completed, choose **Resource Management > Nodes** in the navigation pane. On the node list, you can view the worker nodes added during cluster auto scaling.

----End

## 17.3.5 Upgrading a Cluster

### 17.3.5.1 Overview

To enable interoperability from one Kubernetes installation to the next, you must upgrade your Kubernetes clusters before the maintenance period ends.

After the latest Kubernetes version is available in CCE, CCE will describe the changes in this version.

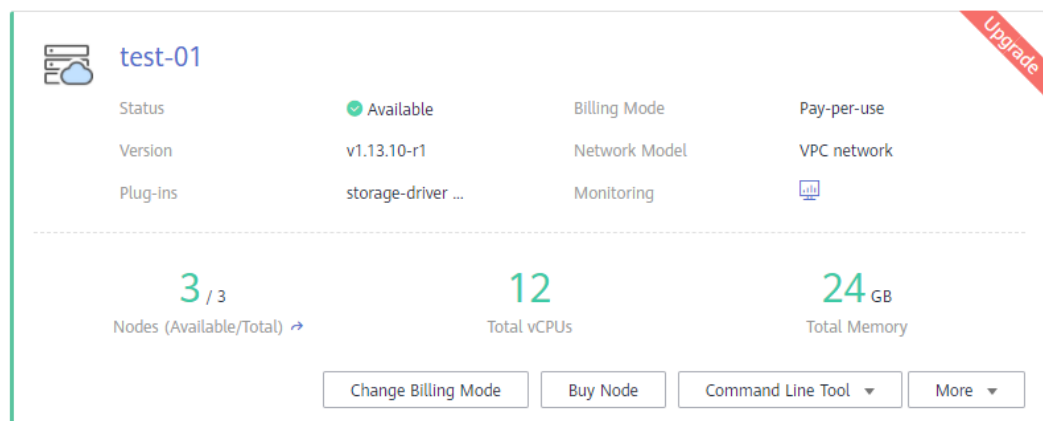
You can use the CCE console to upgrade the Kubernetes version of a cluster.

An upgrade flag will be displayed on the cluster card view if there is a new version for the cluster to upgrade.

#### How to check:

Choose **Resource Management > Clusters** and check whether there is an upgrade flag in the upper right corner of the cluster card view. If yes, the cluster can be upgraded.

**Figure 17-6** Cluster with the upgrade flag



## Cluster Upgrade

The following table describes the target version to which each cluster version can be upgraded, the supported upgrade modes, and upgrade impacts.



**Table 17-12** Cluster upgrade paths and impacts

Source Version	Target Version	Upgrade Modes	Impacts
v1.19	v1.21	In-place upgrade	You need to identify the differences between versions.
v1.17 v1.15	v1.19	In-place upgrade	You need to identify the differences between versions.
v1.13	v1.15	Rolling upgrade Replace upgrade	<ul style="list-style-type: none"> <li>The <b>proxy</b> configuration item in the coredns add-on configuration is not supported and needs to be replaced with <b>forward</b>.</li> <li>The storage add-on is changed from storage-driver to everest.</li> </ul>

## Upgrade Modes

CCE provides the following upgrade modes based on the cluster version and deployment site. The upgrade processes are the same for master nodes. The differences between the upgrade modes of worker nodes are described as follows:

**Table 17-13** Differences between upgrade modes and their advantages and disadvantages

Upgrade Mode	Method	Advantage	Disadvantage
<b>In-place upgrade</b>	Kubernetes components, network components, and CCE management components are upgraded on the node. During the upgrade, service pods and networks are not affected. The <b>SchedulingDisabled</b> label will be added to all existing nodes. After the upgrade is complete, you can properly use existing nodes.	You do not need to migrate services, ensuring service continuity.	In-place upgrade does not upgrade the OS of a node. If you want to upgrade the OS, clear the corresponding node after the node upgrade is complete and reset the node to upgrade the OS to a new version.
<b>Rolling upgrade</b>	Only the Kubernetes components and certain network components are upgraded on the node. The <b>SchedulingDisabled</b> label will be added to all existing nodes to ensure that the running applications are not affected. <b>After the upgrade is complete, you need to manually create nodes and gradually release the old nodes</b> , thereby migrating your applications to the new nodes. In this mode, you can control the upgrade process.	Services are not interrupted.	-
<b>Replace upgrade</b>	The latest worker node image is used to reset the node OS.	This is the fastest upgrade mode and requires few manual interventions.	Data or configurations on the node will be lost, and services will be interrupted for a period of time.

### 17.3.5.2 Before You Start

Before the upgrade, you can check whether your cluster can be upgraded and which versions are available on the CCE console. For details, see [Overview](#).

#### Precautions

- **Upgraded clusters cannot be rolled back. Therefore, perform the upgrade during off-peak hours to minimize the impact on your services.**
- Do not shut down or restart nodes during cluster upgrade. Otherwise, the upgrade fails.
- Before upgrading a cluster, disable auto scaling policies to prevent node scaling during the upgrade. Otherwise, the upgrade fails.
- If you locally modify the configuration of a cluster node, the cluster upgrade may fail or the configuration may be lost after the upgrade. Therefore, modify the configurations on the CCE console (cluster or node pool list page) so that they will be automatically inherited during the upgrade.
- During the cluster upgrade, the running workload services will not be interrupted, but access to the API server will be temporarily interrupted.
- Before upgrading the cluster, check whether the cluster is healthy.
- To ensure data security, you are advised to back up data before upgrading the cluster. During the upgrade, you are not advised to perform any operations on the cluster.

#### Notes and Constraints

- Currently, only CCE clusters consisting of VM nodes can be upgraded.
- If initContainer or Istio is used in the in-place upgrade of a cluster of v1.15, pay attention to the following restrictions:

In kubelet 1.16 and later versions, [QoS classes](#) are different from those in earlier versions. In kubelet 1.15 and earlier versions, only containers in **spec.containers** are counted. In kubelet 1.16 and later versions, containers in both **spec.containers** and **spec.initContainers** are counted. The QoS class of a pod will change after the upgrade. As a result, the container in the pod restarts. You are advised to modify the QoS class of the service container before the upgrade to avoid this problem. For details, see [Table 17-14](#).

**Table 17-14** QoS class changes before and after the upgrade

Init Container (Calculated Based on spec.initContainers)	Service Container (Calculated Based on spec.containers)	Pod (Calculated Based on spec.containers and spec.initContainers)	Impacted or Not
Guaranteed	Besteffort	Burstable	Yes
Guaranteed	Burstable	Burstable	No
Guaranteed	Guaranteed	Guaranteed	No

Init Container (Calculated Based on spec.initContainers)	Service Container (Calculated Based on spec.containers)	Pod (Calculated Based on spec.containers and spec.initContainers)	Impacted or Not
Besteffort	Besteffort	Besteffort	No
Besteffort	Burstable	Burstable	No
Besteffort	Guaranteed	Burstable	Yes
Burstable	Besteffort	Burstable	Yes
Burstable	Burstable	Burstable	No
Burstable	Guaranteed	Burstable	Yes

## Performing Pre-upgrade Check

Before upgrading a cluster, check the health status of the cluster and nodes and ensure that they are available.

### Method 1: Use the console.

On the CCE console, click **Resource Management** in the navigation pane, and click **Clusters** and **Nodes** separately to check whether the cluster and nodes are normal.

### Method 2: Run kubectl commands.

**Step 1** Run the following command to verify that all cluster modules are in the Healthy state:

#### kubectl get cs

Information similar to the following is displayed:

```

NAME          STATUS  MESSAGE  ERROR
scheduler     Healthy ok
controller-manager Healthy ok
etcd-0        Healthy {"health": "true"}
etcd-1        Healthy {"health": "true"}
etcd-2        Healthy {"health": "true"}
    
```

#### NOTE

In the command output, the value of **STATUS** must be **Healthy** for all items.

**Step 2** Run the following command to verify that all nodes are in the Ready state:

#### kubectl get nodes

#### NOTE

All nodes must be in the **Ready** state.

```

NAME          STATUS  ROLES  AGE  VERSION
xxx.xxx.xx.xx Ready  <none> 38d  v1.9.7-r1
    
```

```
xxx.xxx.xx.xx Ready <none> 38d v1.9.7-r1
xxx.xxx.xx.xx Ready <none> 38d v1.9.7-r1
```

----End

## Pre-upgrade Checklist

Before upgrading a cluster, follow the pre-upgrade checklist to identify risks and problems in advance.

**Table 17-15** Cluster upgrade check items

Module	Item
Cluster	Check whether the node IP addresses (including EIPs) of the current cluster are used in other configurations or whitelists.
	Perform the pre-upgrade check.
Workload	Record the number and status of workloads for comparison after the upgrade.
	For the databases you use (such as Direct Connect, Redis, and MongoDB), you need to consider the changes in their whitelists, routes, or security group policies in advance.
Storage	Record the storage status to check whether storage resources are lost after the upgrade.
Networking	Check and back up the load balancing services and ingresses.
	If Direct Connect is used, check whether the upgrade causes changes in the IP addresses of nodes or pods where services are deployed. To handle changes, you need to enable routes on Direct Connect in advance.
O&M	Private configurations: Check whether data plane passwords, certificates, and environment variables are configured for nodes or containers in the cluster before the upgrade. If a container is restarted (for example, the node is abnormal and the pod is re-scheduled), the configurations will be lost and your service will be abnormal.
	Check and back up kernel parameters or system configurations.

## Upgrade Backup

Currently, there are two backup modes for cluster upgrade:

- etcd database backup: CCE automatically backs up the etcd database during the cluster upgrade.
- Master node backup (recommended, **manual confirmation required**): On the upgrade confirmation page, click **Backup** to back up the entire master node of the cluster. The backup process uses the Cloud Backup and Recovery

(CBR) service and takes about 20 minutes. If there are many cloud backup tasks at the current site, the backup time may be prolonged.

### 17.3.5.3 Performing Replace/Rolling Upgrade (v1.13 and Earlier)

#### Scenario

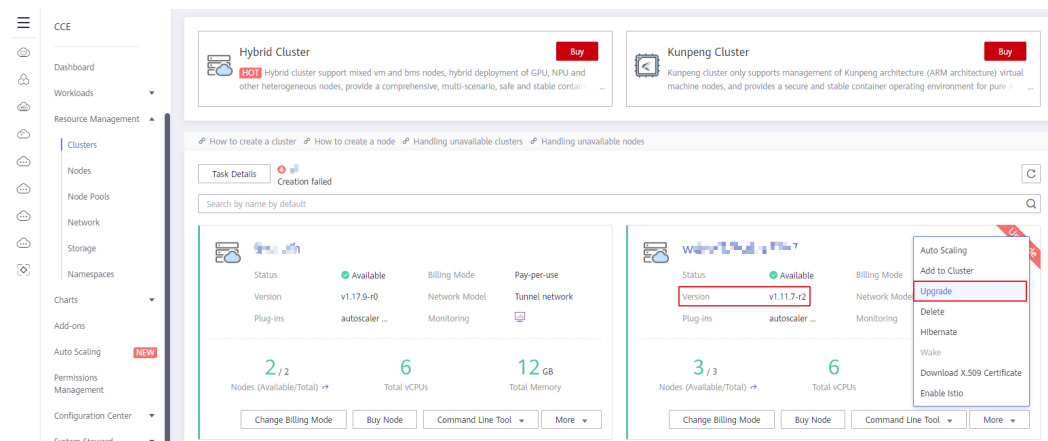
You can upgrade your clusters to a newer Kubernetes version on the CCE console.

Before the upgrade, learn about the target version to which each CCE cluster can be upgraded in what ways, and the upgrade impacts. For details, see [Overview](#) and [Before You Start](#).

#### Procedure

- Step 1** Log in to the CCE console. In the navigation pane, choose **Resource Management > Clusters**. In the cluster list, check the cluster version.
- Step 2** Click **More** for the cluster you want to upgrade, and select **Upgrade** from the drop-down menu.

**Figure 17-7** Upgrading a cluster

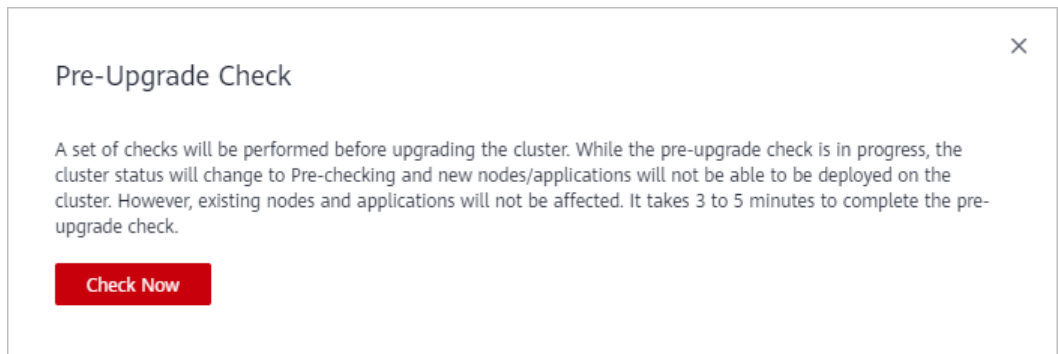


#### **NOTE**

- If your cluster version is up-to-date, the **Upgrade** button is grayed out.
- If the cluster status is **Unavailable**, the upgrade flag in the upper right corner of the cluster card view will be grayed out. Check the cluster status by referring to [Before You Start](#).

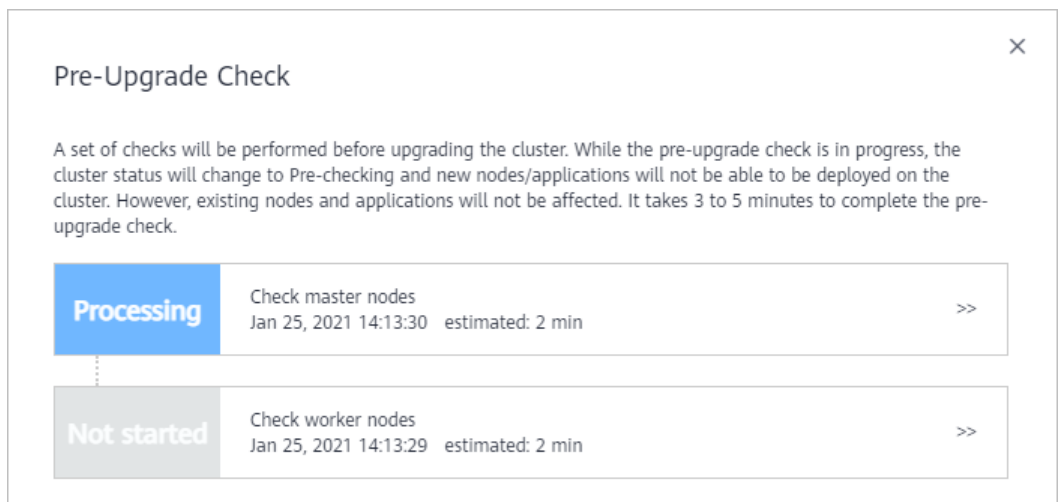
- Step 3** In the displayed **Pre-upgrade Check** dialog box, click **Check Now**.

**Figure 17-8** Pre-upgrade check



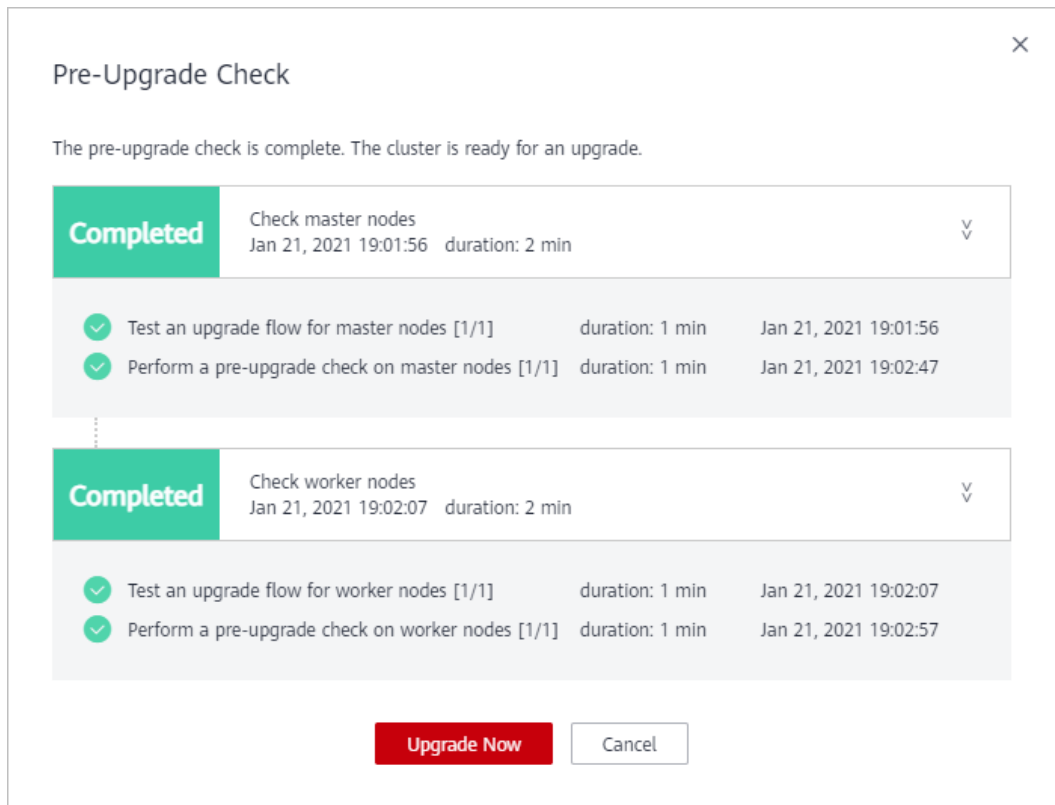
**Step 4** The pre-upgrade check starts. While the pre-upgrade check is in progress, the cluster status will change to **Pre-checking** and new nodes/applications will not be able to be deployed on the cluster. However, existing nodes and applications will not be affected. It takes 3 to 5 minutes to complete the pre-upgrade check.

**Figure 17-9** Pre-upgrade check in process



**Step 5** When the status of the pre-upgrade check is **Completed**, click **Upgrade**.

**Figure 17-10** Pre-upgrade check completed



**Step 6** On the cluster upgrade page, review or configure basic information by referring to [Table 17-16](#).

**Table 17-16** Basic information

Parameter	Description
Cluster Name	Review the name of the cluster to be upgraded.
Current Version	Review the version of the cluster to be upgraded.
Target Version	Review the target version after the upgrade.



Parameter	Description
Node Upgrade Policy	<p><b>Replace</b> (replace upgrade): Worker nodes will be reset. Their OSs will be reinstalled, and data on the system and data disks will be cleared. Exercise caution when performing this operation.</p> <p><b>NOTE</b></p> <ul style="list-style-type: none"> <li>• The lifecycle management function of the nodes and workloads in the cluster is unavailable.</li> <li>• APIs cannot be called temporarily.</li> <li>• Running workloads will be interrupted because nodes are reset during the upgrade.</li> <li>• Data in the system and data disks on the worker nodes will be cleared. Back up important data before resetting the nodes.</li> <li>• Data disks without LVM mounted to worker nodes need to be mounted again after the upgrade, and data on the disks will not be lost during the upgrade.</li> <li>• The EVS disk quota must be greater than 0.</li> <li>• The container IP addresses change, but the communication between containers is not affected.</li> <li>• Custom labels on the worker nodes will be cleared.</li> <li>• It takes about 20 minutes to upgrade a master node and about 30 to 120 minutes to upgrade worker nodes (about 3 minutes for each worker node), depending on the number of worker nodes and upgrade batches.</li> </ul>
Login Mode	<p>You can use a password or key pair.</p> <ul style="list-style-type: none"> <li>• If the login mode is <b>Password</b>: The default username is <b>root</b>. Enter the password for logging in to the node and confirm the password. Remember the password as you will need it when you log in to the node.</li> <li>• <b>Key pair</b>: Select the key pair used to log in to the node. You can select a shared key. A key pair is used for identity authentication when you remotely log in to a node. If no key pair is available, click <b>Create a key pair</b>.</li> </ul>

**Step 7** Click **Next**. In the dialog box displayed, click **OK**.

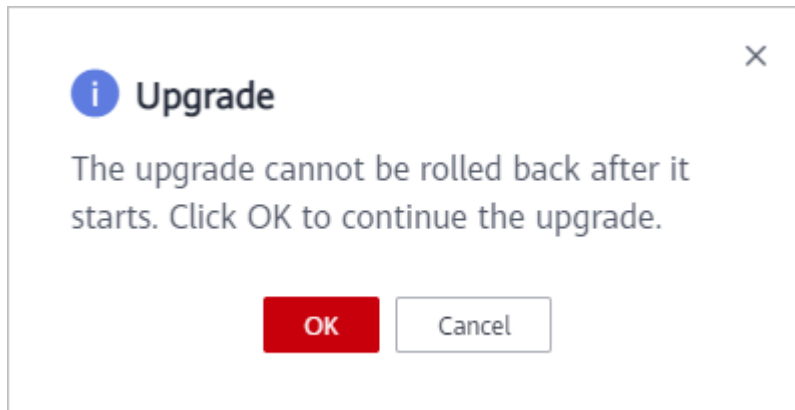
**Step 8** Upgrade add-ons. If an add-on needs to be upgraded, a red dot is displayed. Click the **Upgrade** button in the lower left corner of the add-on card view. After the upgrade is complete, click **Upgrade** in the lower right corner of the page.

 **NOTE**

- Master nodes will be upgraded first, and then the worker nodes will be upgraded concurrently. If there are a large number of worker nodes, they will be upgraded in different batches.
- Select a proper time window for the upgrade to reduce impacts on services.
- Clicking **OK** will start the upgrade immediately, and the upgrade cannot be canceled. Do not shut down or restart nodes during the upgrade.

**Step 9** In the displayed **Upgrade** dialog box, read the information and click **OK**. Note that the cluster cannot be rolled back after the upgrade.

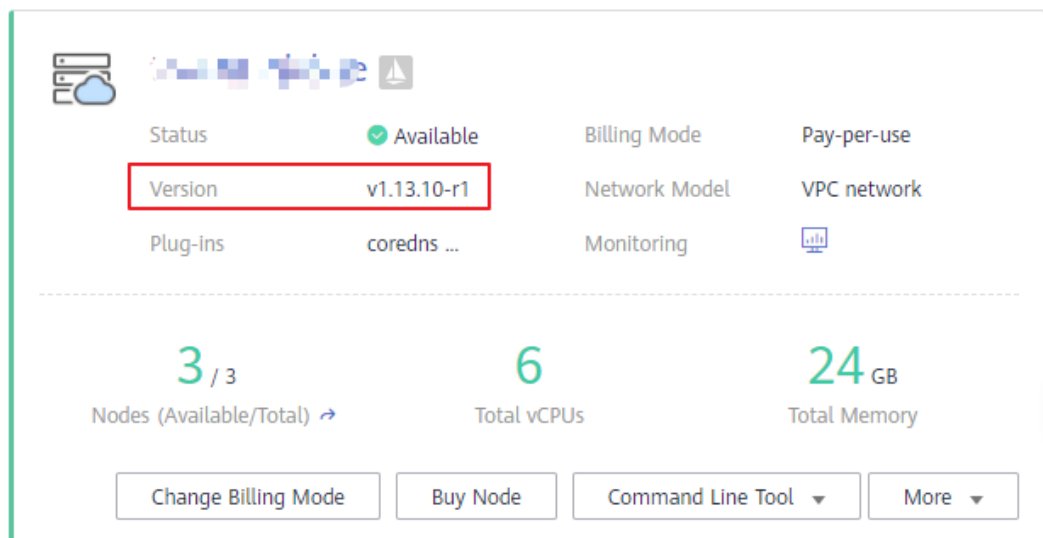
**Figure 17-11** Confirming cluster upgrade



**Step 10** Back to the cluster list, you can see that the cluster status is **Upgrading**. Wait until the upgrade is completed.

After the upgrade is successful, you can view the cluster status and version on the cluster list or cluster details page.

**Figure 17-12** Verifying the upgrade success



----End

### 17.3.5.4 Performing In-place Upgrade (v1.15 and Later)

#### Scenario

On the CCE console, You can perform an in-place cluster upgrade to use new cluster features.

Before the upgrade, learn about the target version to which each CCE cluster can be upgraded in what ways, and the upgrade impacts. For details, see [Overview](#) and [Before You Start](#).

## Description

- An in-place upgrade updates the Kubernetes components on cluster nodes, without changing their OS version.
- Data plane nodes are upgraded in batches. By default, they are prioritized based on their CPU, memory, and [PodDisruptionBudgets \(PDBs\)](#). You can also set the priorities according to your service requirements.

## Precautions

- During the cluster upgrade, the system will automatically upgrade add-ons to a version compatible with the target cluster version. Do not uninstall or reinstall add-ons during the cluster upgrade.
- Before the upgrade, ensure that all add-ons are running. If an add-on fails to be upgraded, rectify the fault and try again.
- During the upgrade, CCE checks the add-on running status. Some add-ons (such as coredns) require at least two nodes to run normally. In this case, at least two nodes must be available for the upgrade.

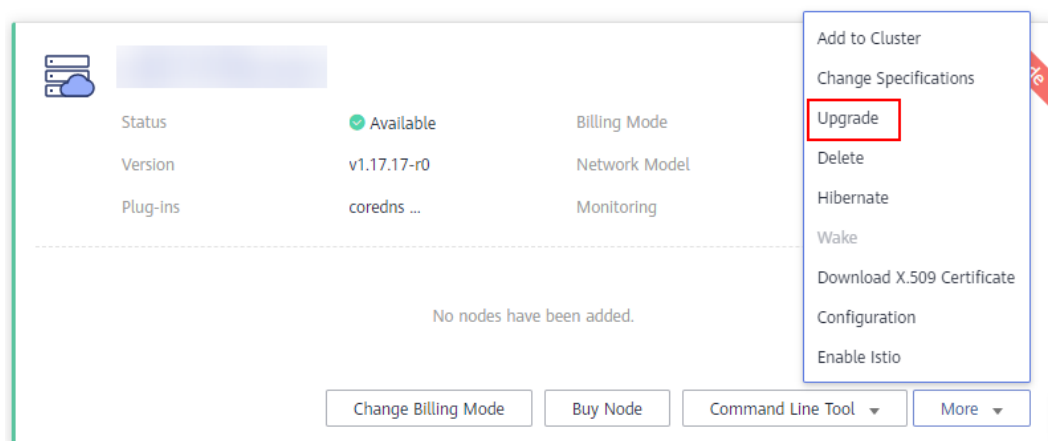
For more information, see [Before You Start](#).

## Procedure

This section describes how to upgrade a CCE cluster of v1.15 or later. For other versions, see [Performing Replace/Rolling Upgrade \(v1.13 and Earlier\)](#).

- Step 1** Log in to the CCE console. In the navigation pane, choose **Resource Management > Clusters**. In the cluster list, view the cluster version.
- Step 2** Click **More** for the cluster you want to upgrade, and select **Upgrade** from the drop-down menu.

Figure 17-13 Upgrading a cluster




 **NOTE**

- If your cluster version is up-to-date, the **Upgrade** button is grayed out.
- If the cluster status is **Unavailable**, the upgrade flag in the upper right corner of the cluster card view will be grayed out. Check the cluster status by referring to [Before You Start](#).

**Step 3** Set the upgrade parameters.

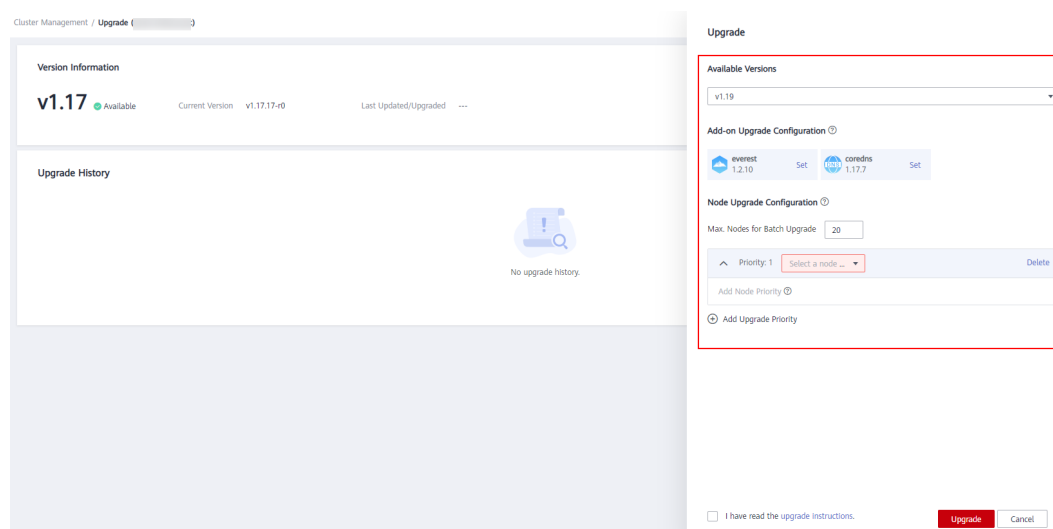
- **Available Versions:** Select v1.19 in this example.
- **Cluster Backup:** A manual confirmation is required for backing up the entire master node. The backup process uses the Cloud Backup and Recovery (CBR) service and takes about 20 minutes. If there are many cloud backup tasks at the current site, the backup time may be prolonged.
- **Add-on Upgrade Configuration:** Add-ons that have been installed in your cluster are listed. During the cluster upgrade, the system automatically upgrades the add-ons to be compatible with the target cluster version. You can click **Set** to re-define the add-on parameters.

 **NOTE**

If a red dot  is displayed on the right of an add-on, the add-on is incompatible with the target cluster version. During the upgrade, the add-on will be uninstalled and then re-installed. Ensure that the add-on parameters are correctly configured.

- **Node Upgrade Configuration:** Before setting the node upgrade priority, you need to select a node pool. Nodes and node pools will be upgraded according to the priorities you specify. You can set the maximum number of nodes to be upgraded in batch, or set priorities for nodes to be upgraded. If you do not set this parameter, the system will determine the nodes to upgrade in batches based on specific conditions.
  - **Add Upgrade Priority:** Add upgrade priorities for node pools.
  - **Add Node Priority:** After adding a node pool priority, you can set the upgrade sequence of nodes in the node pool. The system upgrades nodes in the sequence you specify. If you skip this setting, the system upgrades nodes based on the default policy.

**Figure 17-14** Configuring upgrade parameters



**Step 4** Click **Upgrade**.

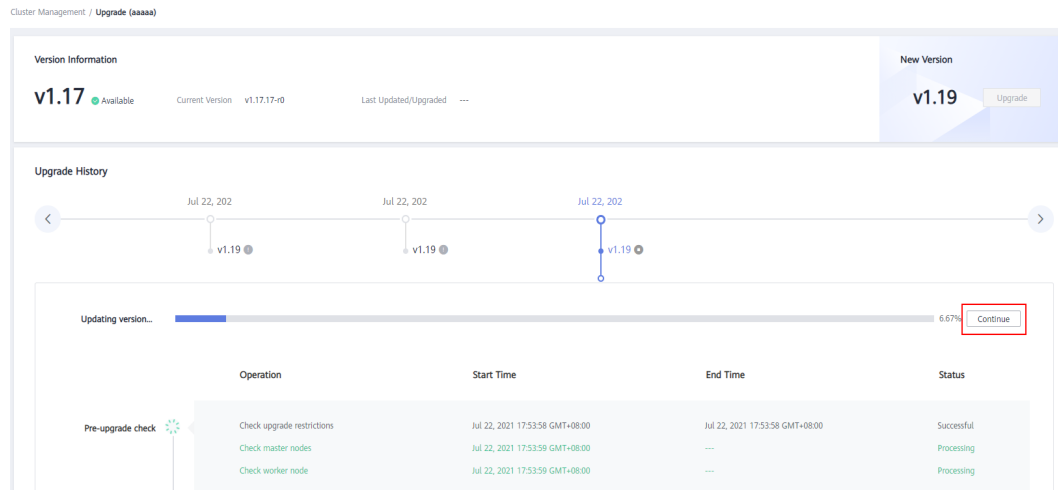
**Figure 17-15** Final step before upgrade



**Step 5** After you click **Upgrade**, the cluster upgrade starts. You can view the upgrade process in the lower part of the page.

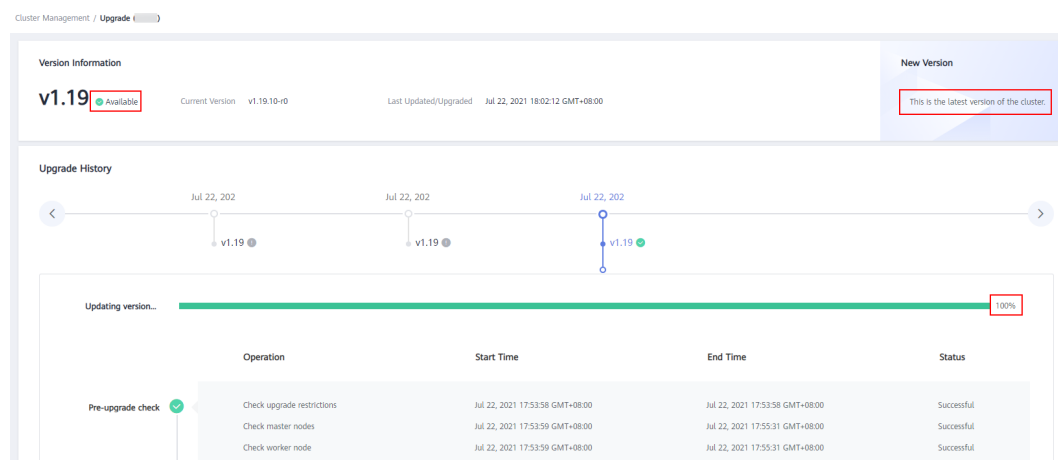
During the upgrade, you can click **Suspend** on the right to suspend the cluster upgrade. To continue the upgrade, click **Continue**.

**Figure 17-16** Cluster upgrade in process



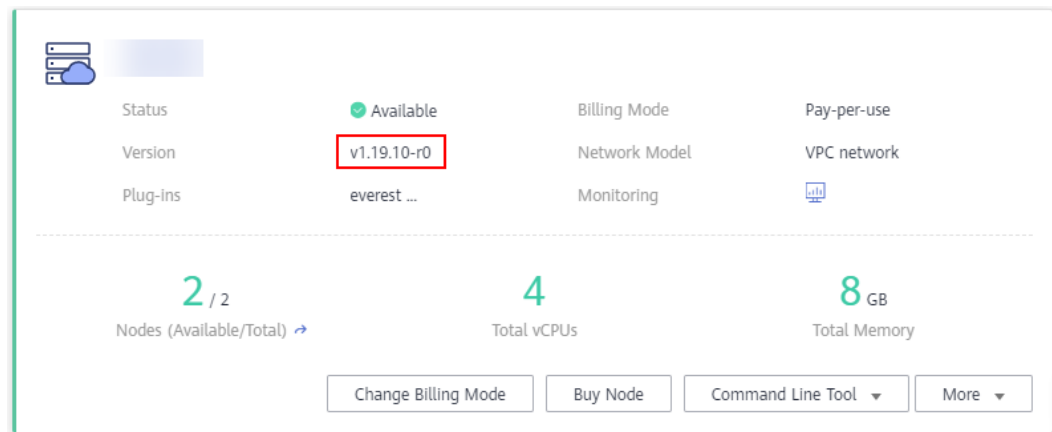
**Step 6** When the upgrade progress reaches 100%, the cluster is upgraded. The version information will be properly displayed, and no upgrade is required.

**Figure 17-17** Upgrade completed



**Step 7** After the upgrade is complete, verify the cluster Kubernetes version on the **Clusters** page.

**Figure 17-18** Verifying the upgrade success



----End

### 17.3.5.5 Migrating Services Across Clusters of Different Versions

#### Application Scenarios

This section describes how to migrate services from a cluster of an earlier version to a cluster of a later version in CCE.

This operation is applicable when a cross-version cluster upgrade is required (for example, upgrade from v1.7.\* or v1.9.\* to 1.17.\*) and new clusters can be created for service migration.

#### Prerequisites

**Table 17-17** Checklist before migration

Category	Description
Cluster	NodeIP-related: Check whether node IP addresses (including EIPs) of the cluster before the migration have been used in other configurations or whitelists.
Workloads	Record the number of workloads for post-migration check.
Storage	<ol style="list-style-type: none"> <li>1. Check whether the storage resources in use are provisioned by the cloud or by your organization.</li> <li>2. Change the automatically created storage to the existing storage in the new cluster.</li> </ol>
O&M	Private configuration: Check whether kernel parameters or system data have been configured on nodes in the cluster.

#### Procedure

##### Step 1 Create a CCE cluster.

Create a cluster with the same specifications and configurations as the cluster of the earlier version. For details, see [Buying a CCE Cluster](#).

**Step 2 Add a node.**

Add nodes with the same specifications and manual configuration items. For details, see [Buying a Node](#).

**Step 3 Create a storage volume in the new cluster.**

Use an existing storage volume to create a PVC in the new cluster. The PVC name remains unchanged. For details, see [PersistentVolumeClaims \(PVCs\)](#).

 **NOTE**

Storage switching supports only OBS buckets, SFS file systems, and shared EVS disks. If a non-shared EVS disk is used, you need to suspend the workloads in the old cluster to switch the storage resources. As a result, services will be interrupted.

**Step 4 Create a workload in the new cluster.**

The workload name and specifications remain unchanged. For details about how to create a workload, see [Creating a Deployment](#) or [Creating a StatefulSet](#). For details about how to mount a storage volume to the workload, see [Creating a Pod Mounted with an EVS Volume](#).

**Step 5 Create a Service in the new cluster.**

The Service name and specifications remain unchanged. For details about how to create a Service, see [Services](#).

**Step 6 Commission services.**

After all resources are created, commission the containerized services. If the commissioning is successful, migrate the services to the new cluster.

**Step 7 Delete or unsubscribe from the old cluster.**

When all functions of the new cluster are stable, unsubscribe from or delete the old cluster. For details about how to delete a cluster, see [Deleting a Cluster \(Pay-per-Use\)](#).

----End

### 17.3.5.6 CCE Kubernetes Release Notes

To enable interoperability from one Kubernetes installation to the next, you must upgrade your Kubernetes clusters before the maintenance period ends.

After the latest Kubernetes version is released, CCE will provide you the changes in this version. For details, see [Table 17-18](#).

**Table 17-18** Cluster version differences

Source Version	Target Version	Description
v1.19	v1.21	<ul style="list-style-type: none"> <li>Changelog from v1.19 to v1.21 Changelog from v1.20 to v1.21: <a href="https://github.com/kubernetes/kubernetes/blob/master/CHANGELOG/CHANGELOG-1.21.md">https://github.com/kubernetes/kubernetes/blob/master/CHANGELOG/CHANGELOG-1.21.md</a></li> <li>Changelog from v1.19 to v1.20: <a href="https://github.com/kubernetes/kubernetes/blob/master/CHANGELOG/CHANGELOG-1.20.md">https://github.com/kubernetes/kubernetes/blob/master/CHANGELOG/CHANGELOG-1.20.md</a></li> </ul>
v1.17	v1.19	<ul style="list-style-type: none"> <li>Changelog from v1.17 to v1.19 Changelog from v1.18 to v1.19: <a href="https://github.com/kubernetes/kubernetes/blob/master/CHANGELOG/CHANGELOG-1.19.md">https://github.com/kubernetes/kubernetes/blob/master/CHANGELOG/CHANGELOG-1.19.md</a></li> <li>Changelog from v1.17 to v1.18: <a href="https://github.com/kubernetes/kubernetes/blob/master/CHANGELOG/CHANGELOG-1.18.md">https://github.com/kubernetes/kubernetes/blob/master/CHANGELOG/CHANGELOG-1.18.md</a></li> </ul>
v1.15	v1.17	<ul style="list-style-type: none"> <li>Changelog from v1.15 to v1.17 Changelog from v1.16 to v1.17: <a href="https://github.com/kubernetes/kubernetes/blob/master/CHANGELOG/CHANGELOG-1.17.md">https://github.com/kubernetes/kubernetes/blob/master/CHANGELOG/CHANGELOG-1.17.md</a></li> <li>Changelog from v1.15 to v1.16: <a href="https://github.com/kubernetes/kubernetes/blob/master/CHANGELOG/CHANGELOG-1.16.md">https://github.com/kubernetes/kubernetes/blob/master/CHANGELOG/CHANGELOG-1.16.md</a></li> </ul>

## 17.3.6 Managing a Cluster

### 17.3.6.1 Deleting a Cluster (Pay-per-Use)

#### Scenario

This section describes how to delete a cluster billed on a pay-per-use basis.

You can unsubscribe from a yearly/monthly-billed cluster. Deletion without unsubscription will not cut your bills. For details, see [Deleting, Unsubscribing From, or Releasing a Yearly/Monthly-Billed Cluster](#).

#### Precautions

- Deleting a cluster will not delete the yearly/monthly-billed resources in the cluster, and their billing continues.
- Deleting a cluster will delete the nodes in the cluster (excluding accepted nodes), data disks attached to the nodes, workloads, and Services. Related services cannot be restored. Before performing this operation, ensure that data has been backed up or migrated. Deleted data cannot be restored.



Resources that are not created in CCE will not be deleted:

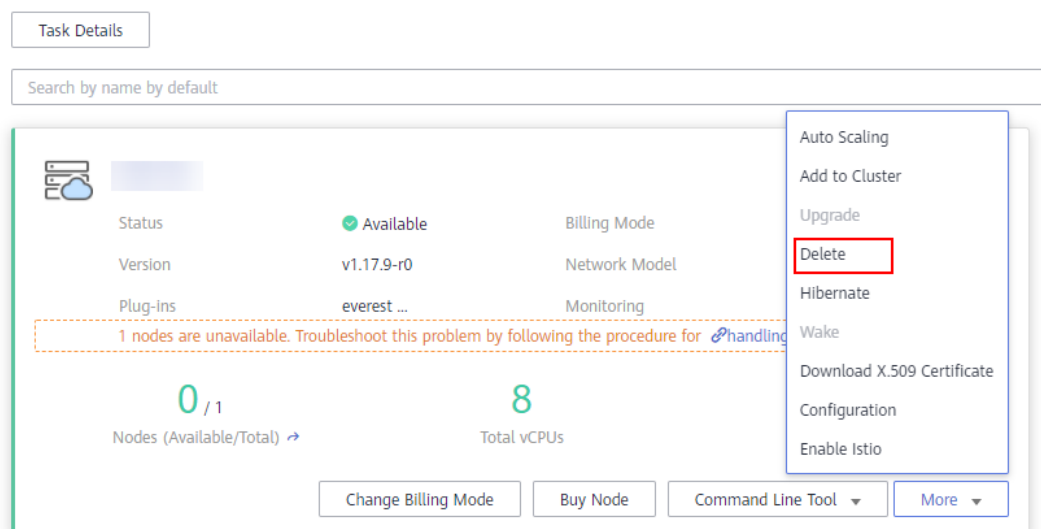
- Accepted nodes (only the nodes created in CCE are deleted);
  - ELB load balancers associated with Services and ingresses (only the automatically created load balancers are deleted);
  - Manually created cloud storage resources associated with PVs or imported cloud storage resources (only the cloud storage resources automatically created by PVCs are deleted)
- A hibernated cluster cannot be deleted. Wake up the cluster and try again.
  - If a cluster whose status is Unavailable is deleted, some storage resources of the cluster may need to be manually deleted.

## Procedure

**Step 1** Log in to the CCE console. In the navigation pane, choose **Resource Management > Clusters**.

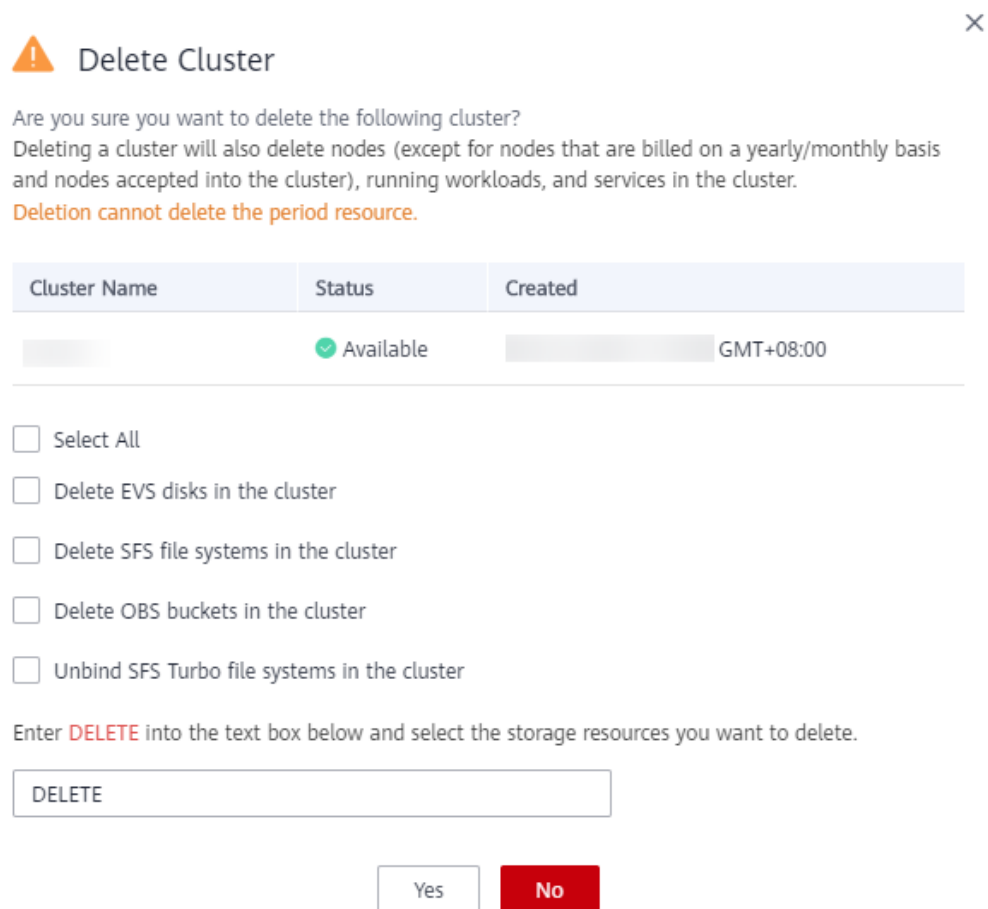
**Step 2** Choose **More > Delete**.

**Figure 17-19** Deleting a cluster



**Step 3** Delete the cluster.

**Figure 17-20** Deleting a cluster



**Step 4** Click **Yes** to start deleting the cluster.

The delete operation takes 1 to 3 minutes to complete.

----End

### 17.3.6.2 Deleting, Unsubscribing From, or Releasing a Yearly/Monthly-Billed Cluster

You can delete, unsubscribe from, or release a yearly/monthly-billed cluster.

You can directly delete pay-per-use clusters. For details, see [Deleting a Cluster \(Pay-per-Use\)](#).

#### Precautions

- Unsubscribing from or releasing a cluster will delete the nodes in the cluster (excluding accepted nodes), data disks attached to the nodes, workloads, and Services. Related services cannot be restored. Before performing this operation, ensure that data has been backed up or migrated. Removed data cannot be restored.

Resources that are not created in CCE will not be deleted:

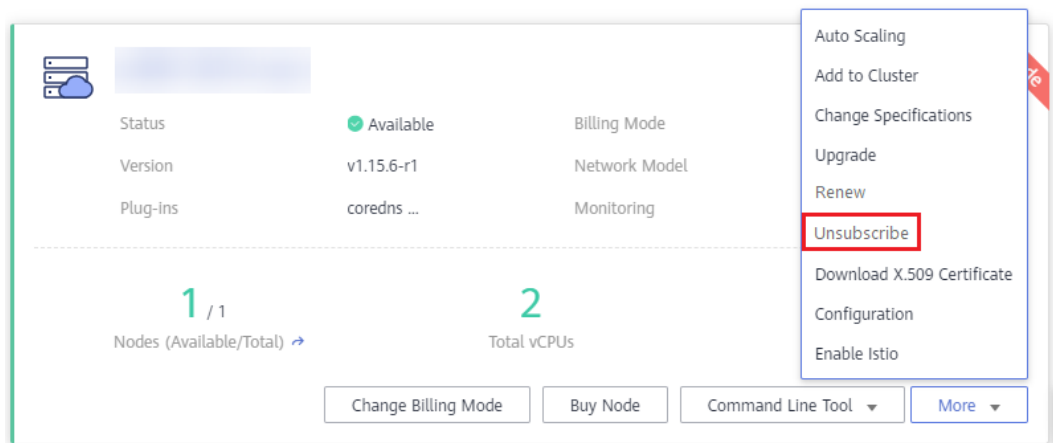
- Accepted nodes (only the nodes created in CCE are deleted);
- ELB load balancers associated with Services and ingresses (only the automatically created load balancers are deleted);
- Manually created cloud storage resources associated with PVs or imported cloud storage resources (only the cloud storage resources automatically created by PVCs are deleted)
- When you unsubscribe from or release a cluster, only the resources associated with the order are unsubscribed from. Non-associated resources are retained and their billing continues.
- For a yearly/monthly cluster, if the retention expires, the cluster will be automatically released. For the nodes in the cluster, if their retention expires at the same time, they will also be released; if not, the node OS will be reinstalled. Pay attention to the expired clusters under your account and renew them in a timely manner to prevent data loss caused by node reinstallation.
- Cluster resources include master node resources and IaaS resources used by worker nodes. For details, see [Pricing Details](#).
- If an order contains resources in a primary-secondary relationship, you need to unsubscribe from the resources separately.

## Unsubscribing From a Cluster

This section describes how to unsubscribe from a **yearly/monthly-billed** cluster that has not expired.

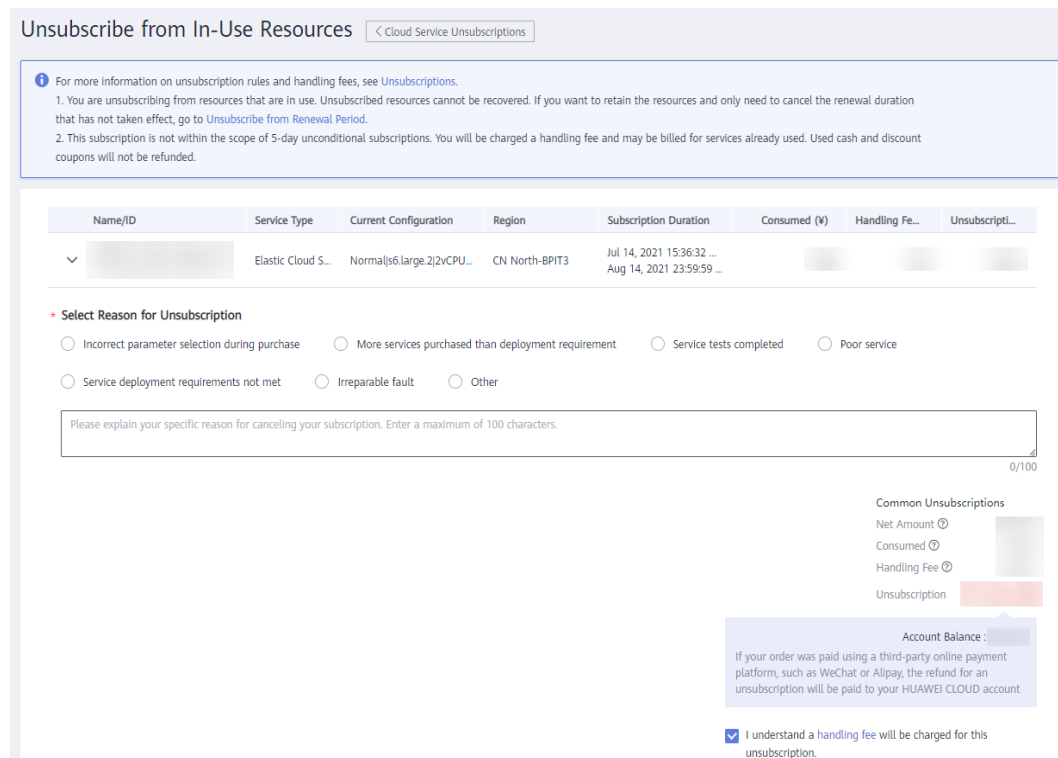
- Step 1** Log in to the CCE console. In the navigation pane, choose **Resource Management > Clusters**.
- Step 2** Choose **More > Unsubscribe** in the card view of the cluster you want to unsubscribe from.

**Figure 17-21** Unsubscribing from a cluster



- Step 3** On the **Unsubscribe from In-Use Resources** page, unsubscribe from the resources as prompted.

**Figure 17-22** Unsubscribing from cluster resources

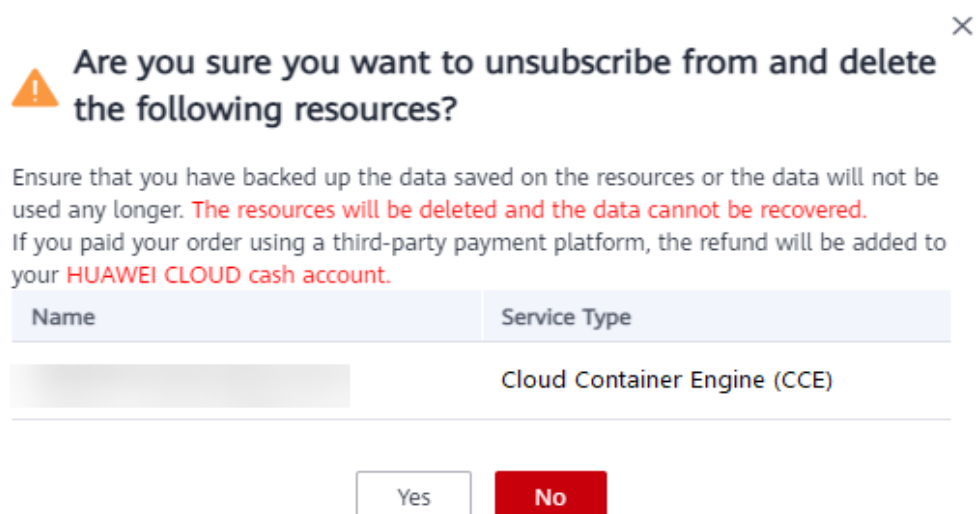


**Step 4** Select the check box "Data will be deleted and cannot be recovered. I confirm I have backed up or I no longer need the data". Click **Confirm**.

**Step 5** In the dialog box displayed, read the instructions and review the information about the resources to be unsubscribed from. Click **Yes** to unsubscribe from the cluster.

The release takes 1 to 3 minutes to complete.

**Figure 17-23** Unsubscription review



 **NOTE**

- Ensure that you have backed up or no longer need the data on the resources. Unsubscribed resources will be deleted and the data cannot be recovered.
- If you paid your order using a third-party payment platform, the refund will be added to your Huawei Cloud cash account.

----End

## Releasing a Cluster

This section describes how to release a **yearly/monthly-billed** cluster that has expired without renewal.

**Step 1** Log in to the CCE console. In the navigation pane, choose **Resource Management > Clusters**.

**Step 2** Choose **More > Release** in the card view of the cluster to be released.

**Step 3** Release the cluster as prompted.

 **NOTE**

- Releasing the cluster will destroy all nodes (except those accepted into the cluster), running workloads, and services in the cluster. Released resources cannot be restored.
- The release takes 1 to 3 minutes to complete.

**Step 4** Click **Release**. Read the instructions, review the resource information, click **OK** to release the resource.

The release takes 1 to 3 minutes to complete.

----End

### 17.3.6.3 Renewing a Yearly/Monthly-Billed Cluster

You can renew a yearly/monthly-billed cluster.

#### Procedure

This section describes how to renew a **yearly/monthly-billed** CCE cluster.

---

**NOTICE**

A yearly/monthly-billed cluster will be deleted if it is not renewed after expiration, and all nodes and the running services in the cluster will be destroyed. CCE strongly recommends that you renew the cluster before it expires or **enable auto renewal**.

---

**Step 1** Log in to the CCE console. In the navigation pane, choose **Resource Management > Clusters**.

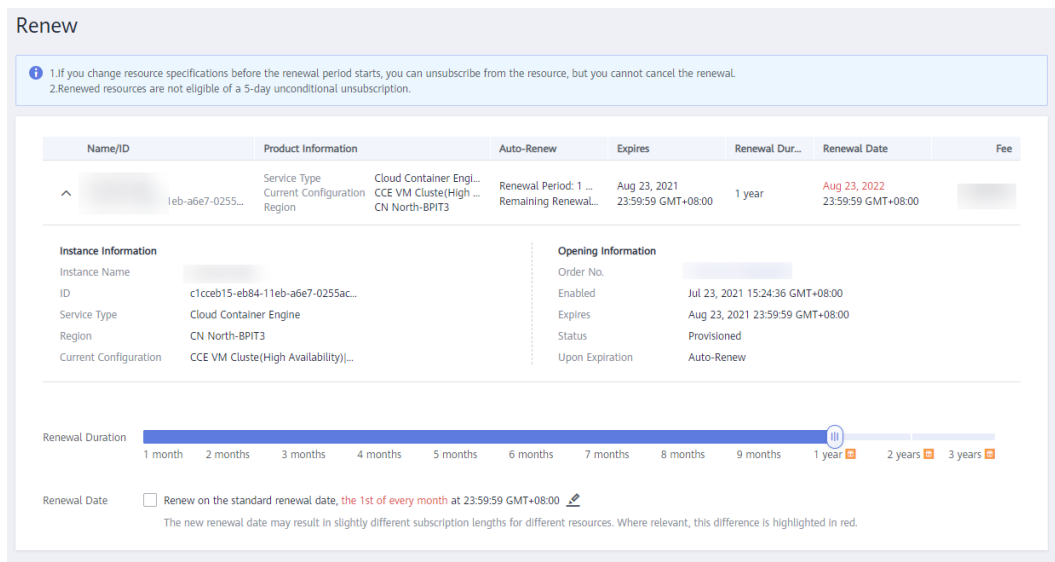
**Step 2** Choose **More > Renew** in the card view of the cluster to be renewed.

**Step 3** On the displayed page, renew the service as prompted.

**NOTE**

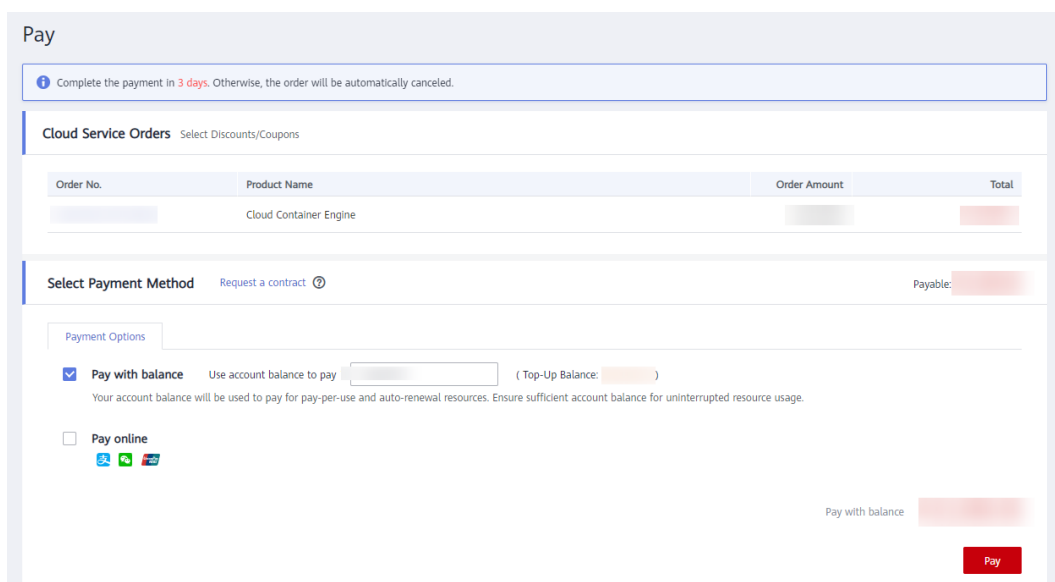
- If the selected resource (highlighted) is associated with other resources, you can decide whether you want to perform the operation on all these resources at the same time.
- If you change resource specifications before the renewal period starts, you can unsubscribe from the resource, but not the renewal period.
- Renewed resources are not eligible of a 5-day unconditional unsubscription.

**Figure 17-24** Renewing a cluster



**Step 4** Click **Pay**. On the page displayed, review the order amount, select a payment method, and click **Pay**.

**Figure 17-25** Paying for the renewal



**Step 5** After the payment is complete, you can go back to the **Orders** or **Renewals** page to view and manage your order.

----End

### 17.3.6.4 Hibernating and Waking Up a Cluster (Pay-per-Use)

#### Scenario

If you do not need to use a cluster temporarily, you are advised to hibernate the cluster to save cluster management costs.

After a cluster is hibernated, resources such as workloads cannot be created or managed in the cluster.

A hibernated cluster can be quickly woken up and used normally.

#### Notes and Constraints

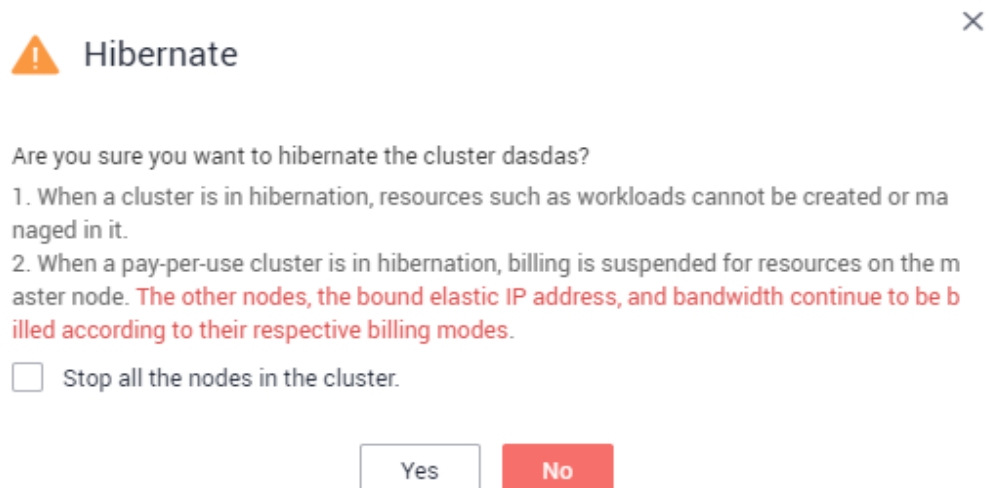
Yearly/Monthly-billed clusters cannot be hibernated.

During cluster wakeup, the master node may fail to be started due to insufficient resources. As a result, the cluster fails to be woken up. Wait for a while and wake up the cluster again.

#### Hibernating a Cluster

- Step 1** Log in to the CCE console. In the navigation pane, choose **Resource Management > Clusters**.
- Step 2** Choose **More > Hibernate** for the target cluster.
- Step 3** In the dialog box displayed, check the precautions and click **Yes**. Wait until the cluster is hibernated.

Figure 17-26 Hibernating a cluster



#### NOTICE

- After a pay-per-use cluster is hibernated, the billing of master node resources will stop.
- After a cluster is hibernated, resources, such as worker nodes (ECSs), bound EIPs, and bandwidth, are still billed based on their own billing modes. To shut down nodes, select **Stop all nodes in the cluster** in the dialog box or see [Stopping a Node](#).

**Step 4** When the cluster status changes from **Hibernating** to **Hibernation**, the cluster is hibernated.

----End

## Waking Up a Cluster

**Step 1** Log in to the CCE console. In the navigation pane, choose **Resource Management > Clusters**.

**Step 2** Choose **More > Wake**.

**Step 3** In the dialog box displayed, click **Yes** and wait until the cluster is woken up.

**Step 4** When the cluster status changes from **Waking** to **Available**, the cluster is woken up.

#### NOTE

After the cluster is woken up, billing will be resumed for the resources on the master node.

----End

### 17.3.6.5 Changing the Billing Mode from Pay-per-Use to Yearly/Monthly

Currently, clusters support **pay-per-use** and **yearly/monthly** billing modes. A pay-per-use cluster can be converted to a yearly/monthly-billed cluster.

## Notes and Constraints

- You are not advised to perform any operation on a node on the ECS console. If you perform operations on the ECS console, the PersistentVolumeClaim (PVC) is bound to the node. As a result, the PVC cannot be used by other nodes.
- Only nodes in the default node pool **DefaultPool** can be changed to yearly/monthly billing mode.
- Nodes whose billing mode is changed to yearly/monthly do not support auto scaling.

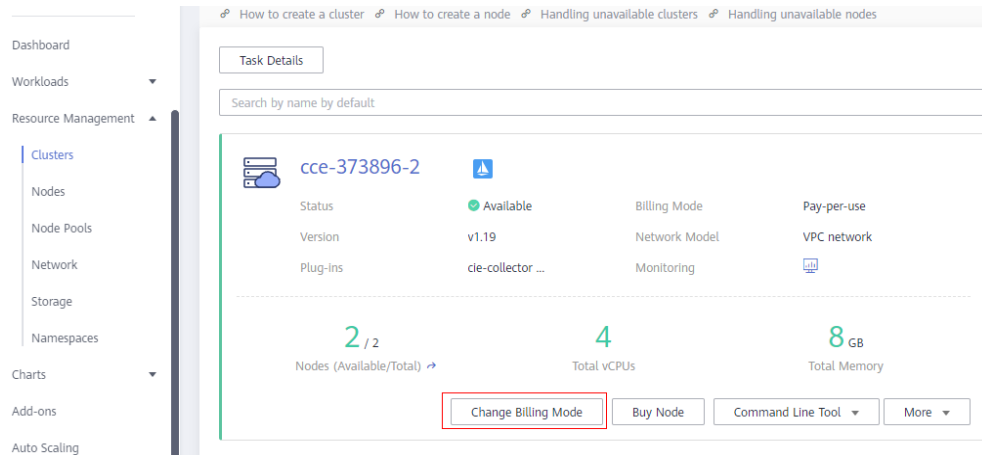
## Changing to Yearly/Monthly Billing

To change the billing mode of the clusters you have purchased from pay-per-use to yearly/monthly, perform the following steps:



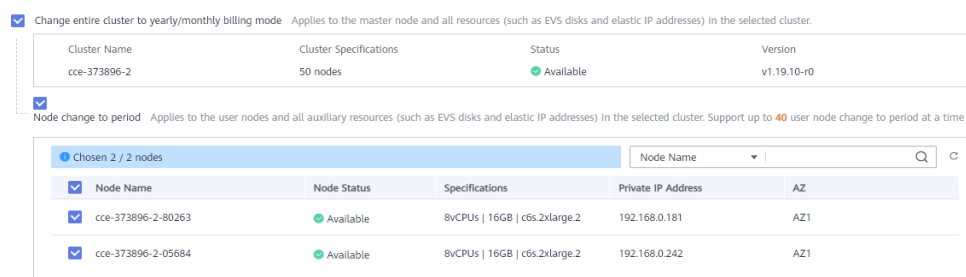
**Step 1** Log in to the CCE console. In the navigation pane, choose **Resource Management > Clusters**. In the card view of the clusters for which you will change the billing mode, click **Change Billing Mode**.

**Figure 17-27** Changing to the yearly/monthly billing mode



**Step 2** On the **Change Billing Mode** page, choose the master and worker nodes that will be changed to the yearly/monthly billing mode.

**Figure 17-28** Changing billing mode for master and worker nodes



**Step 3** Click **OK**. Wait until the order is processed and the payment is complete.

----End

### 17.3.6.6 Configuring Kubernetes Parameters

#### Scenario

CCE clusters allow you to manage Kubernetes parameters, through which you can let core components work under your very requirements.

#### Notes and Constraints

This function is supported only in clusters of **v1.15 and later**. It is not displayed for versions earlier than v1.15.

#### Procedure

**Step 1** Log in to the CCE console. In the navigation pane, choose **Resource Management > Clusters**.

**Step 2** Choose **More > Configuration**.

**Step 3** On the **Configuration** page on the right, change the values of the following Kubernetes parameters:

**Table 17-19** Kubernetes parameters

Component	Parameter	Description	Value
kube-apiserver	default-not-ready-toleration-seconds	notReady tolerance time, in seconds. NoExecute that is added by default to every pod that does not already have such a toleration.	Default: 300
	default-unreachable-toleration-seconds	unreachable tolerance time, in seconds. NoExecute that is added by default to every pod that does not already have such a toleration.	Default: 300
	max-mutating-requests-inflight	<p>Maximum number of concurrent mutating requests. When the value of this parameter is exceeded, the server rejects requests.</p> <p>The value <b>0</b> indicates no limitation.</p> <p>Manual configuration is no longer supported since cluster version 1.21. The value is automatically specified based on the cluster scale.</p> <ul style="list-style-type: none"> <li>• <b>200</b> for clusters with 50 or 200 nodes</li> <li>• <b>500</b> for clusters with 1000 nodes</li> <li>• <b>1000</b> for clusters with 2000 nodes</li> </ul>	Default: 1000

Component	Parameter	Description	Value
	max-requests-inflight	<p>Maximum number of concurrent non-mutating requests. When the value of this parameter is exceeded, the server rejects requests.</p> <p>The value <b>0</b> indicates no limitation.</p> <p>Manual configuration is no longer supported since cluster version 1.21. The value is automatically specified based on the cluster scale.</p> <ul style="list-style-type: none"> <li>• <b>400</b> for clusters with 50 or 200 nodes</li> <li>• <b>1000</b> for clusters with 1000 nodes</li> <li>• <b>2000</b> for clusters with 2000 nodes</li> </ul>	Default: 2000
	service-node-port-range	Range of node port numbers.	Default: 30000-32767 Options: min>20105 max<32768
kube-controller-manager	concurrent-deployment-syncs	Number of Deployments that are allowed to synchronize concurrently.	Default: 5
	concurrent-endpoint-syncs	Number of endpoints that are allowed to synchronize concurrently.	Default: 5
	concurrent-gc-syncs	Number of garbage collector workers that are allowed to synchronize concurrently.	Default: 20
	concurrent-job-syncs	Number of jobs that can be synchronized at the same time.	Default: 5
	concurrent-namespace-syncs	Number of namespaces that are allowed to synchronize concurrently.	Default: 10

Component	Parameter	Description	Value
	concurrent-replicaset-syncs	Number of ReplicaSets that are allowed to synchronize concurrently.	Default: 5
	concurrent-resource-quota-syncs	Number of resource quotas that are allowed to synchronize concurrently.	Default: 5
	concurrent-service-syncs	Number of Services that are allowed to synchronize concurrently.	Default: 10
	concurrent-serviceaccount-token-syncs	Number of service account tokens that are allowed to synchronize concurrently.	Default: 5
	concurrent-ttl-after-finished-syncs	Number of TTL-after-finished controller workers that are allowed to synchronize concurrently.	Default: 5
	concurrent_rc_syncs	Number of replication controllers that are allowed to synchronize concurrently.	Default: 5
	horizontal-pod-autoscaler-sync-period	How often HPA audits metrics in a cluster.	Default: 15 seconds
	kube-api-qps	Query per second (QPS) to use while talking with kube-apiserver.	Default: 100
	kube-api-burst	Burst to use while talking with kube-apiserver.	Default: 100
kube-scheduler	kube-api-qps	Query per second (QPS) to use while talking with kube-apiserver.	Default: 100
	kube-api-burst	Burst to use while talking with kube-apiserver.	Default: 100

**Step 4** Click **OK**.

----End

## References

- [kube-apiserver](#)
- [kube-controller-manager](#)

- [kube-scheduler](#)

## 17.3.7 Obtaining a Cluster Certificate

### Scenario

Before accessing cluster resources through open-source Kubernetes APIs, obtain the cluster's certificate.

### Procedure

- Step 1** Log in to the CCE console. In the navigation pane, choose **Resource Management > Clusters**.
- Step 2** In the card view of the target cluster, choose **More > Download X.509 Certificate**.
- Step 3** In the **Download X.509 Certificate** dialog box displayed, select the certificate expiration time and download the X.509 certificate of the cluster as prompted.

**Figure 17-29** Downloading a certificate



#### NOTICE

- The downloaded certificate contains three files: **client.key**, **client.crt**, and **ca.crt**. Keep these files secure.
- Certificates are not required for mutual access between containers in a cluster.

----End

## 17.3.8 Changing Cluster Scale

### Scenario

CCE allows you to change the number of nodes managed in a cluster.

### Notes and Constraints

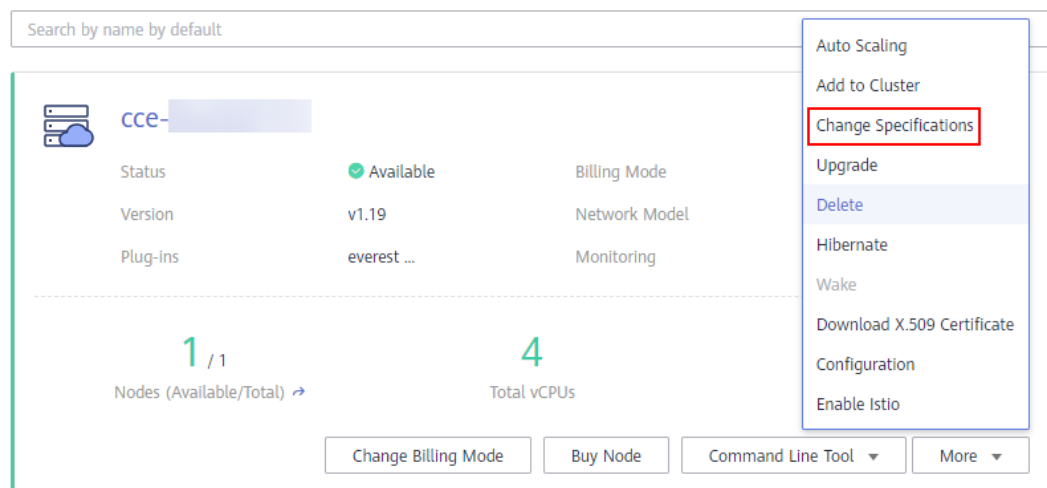
- This function is supported for clusters of v1.15 and later versions.
- Starting from v1.15.11, the number of nodes in a cluster can be changed to 2000. The number of nodes in a single master node cannot be changed to 1000 or more.

- Currently, a cluster can only be scaled out to a larger specification, but cannot be scaled in.
- Cluster scale changes cannot be rolled back. An alarm is reported if the change fails.
- Changing the cluster scale does not affect the services running in the cluster. However, the control plane (master nodes) will be interrupted for a short period of time. You are advised not to perform any other operations (such as creating workloads) during the change.

## Procedure

**Step 1** Log in to the CCE console. In the navigation pane, choose **Resource Management > Clusters**.

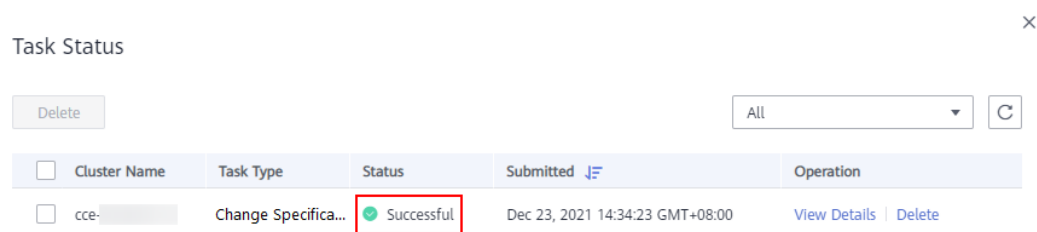
**Step 2** In the cluster card, choose **More > Change Specifications**.



**Step 3** On the displayed page, set **Management Scale** as required.

**Step 4** Click **Submit**.

You can click **Back to Cluster List** and click **Task Status** in the upper left corner to view the cluster change history. The status changes from **Processing** to **Successful**, indicating that the cluster specifications are successfully changed.



**Step 5** (Optional) If the cluster specifications fail to be changed, choose **More > Change Specifications** to change the cluster specifications again.

----End

## 17.3.9 Controlling Cluster Permissions

### Scenario

This section describes how to control permissions on resources in a cluster, for example, allow user A to read and write application data in a namespace, and user B to only read resource data in a cluster.

### Procedure

- Step 1** If you need to perform permission control on the cluster, select **Enhanced authentication** for **Authentication Mode** during cluster creation, upload your own **CA certificate**, **client certificate**, and **client certificate private key** (for details about how to create a certificate, see [Certificates](#)), and select **I have confirmed that the uploaded certificates are valid**. For details, see [Table 17-6](#).

---

**CAUTION**

- Upload a file **smaller than 1 MB**. The CA certificate and client certificate can be in **.crt** or **.cer** format. The private key of the client certificate can only be uploaded **unencrypted**.
- The validity period of the client certificate must be longer than five years.
- The uploaded CA certificate is used for both the authentication proxy and the kube-apiserver aggregation layer configuration. **If the certificate is invalid, the cluster cannot be created.**

- Step 2** Create a role using kubectl.

The following example shows how to create a **role** and allow the role to read all pods in the default namespace. For details about the parameters, see the [official Kubernetes documentation](#).

```
kind: Role
apiVersion: rbac.authorization.k8s.io/v1
metadata:
  namespace: default
  name: pod-reader
rules:
- apiGroups: [""]
  resources: ["pods"]
  verbs: ["get", "watch", "list"]
```

- Step 3** Bind the role to a user by using kubectl.

In the following example, the **RoleBinding** assigns the role of **pod-reader** in the default namespace to user **jane**. This policy allows user **jane** to read all pods in the default namespace. For details about the parameters, see the [official Kubernetes documentation](#).

```
kind: RoleBinding
apiVersion: rbac.authorization.k8s.io/v1
metadata:
  name: read-pods
  namespace: default
subjects:
- kind: User
  name: jane #User name
  apiGroup: rbac.authorization.k8s.io
```

```
roleRef:
  kind: Role
  name: pod-reader #Name of the role that is created
  apiGroup: rbac.authorization.k8s.io
```

**Step 4** After a role is created and bound to a user, call a Kubernetes API by initiating an API request message where headers carry user information and the certificate uploaded during cluster creation. For example, to call the pod query API, run the following command:

```
curl -k -H "X-Remote-User: jane" --cacert /root/tls-ca.crt --key /root/tls.key --cert /root/tls.crt https://192.168.23.5:5443/api/v1/namespaces/default/pods
```

If **200** is returned, user **jane** is authorized to read pods in the cluster's default namespace. If **403** is returned, user **jane** is not authorized to read pods in the cluster's default namespace.

**NOTE**

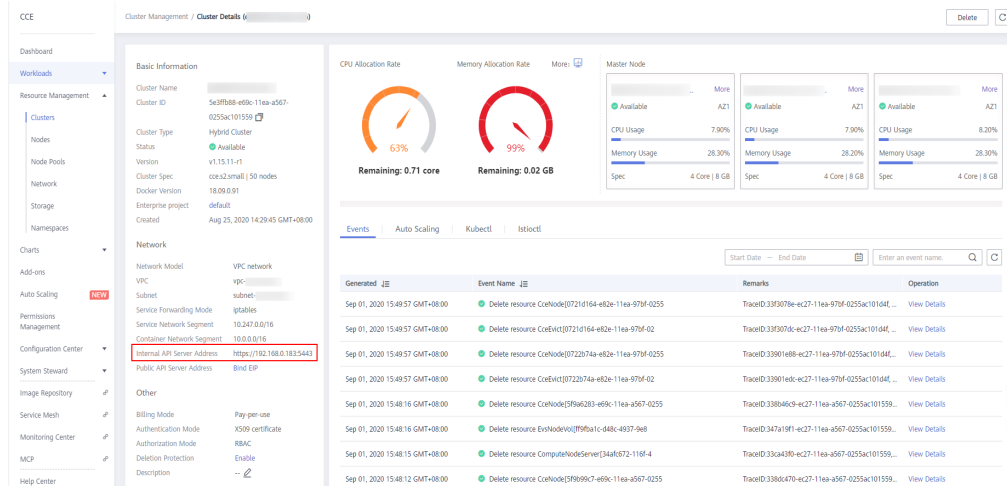
To prevent the command execution failure, upload the certificate to the **/root** directory in advance.

The parameter descriptions are as follows:

- **X-Remote-User: jane:** The request header is fixed at **X-Remote-User**, and **jane** is the username.
- **tls-ca.crt:** CA root certificate uploaded during cluster creation.
- **tls.crt:** client certificate that matches the CA root certificate uploaded during cluster creation.
- **tls.key:** client key corresponding to the CA root certificate uploaded during cluster creation.
- **192.168.23.5:5443:** address for connecting to the cluster. To obtain the address, perform the following steps:

Log in to the CCE console. In the navigation pane, choose **Resource Management > Clusters**. Click the name of the cluster to be connected and obtain the IP address and port number from **Internal API Server Address** on the cluster details page.

**Figure 17-30** Obtaining the access address





In addition, the **X-Remote-Group** header field, that is, the user group name, is supported. During role binding, a role can be bound to a group and carry user group information when you access the cluster.

----End

## 17.3.10 Cluster Parameters

### 17.3.10.1 Maximum Number of Pods That Can Be Created on a Node

The maximum number of pods that can be created on a node is determined by the following parameters:

- Number of container IP addresses that can be allocated on a node (alpha.cce/fixPoolMask): Set this parameter when creating a CCE cluster. This parameter is available only when **Network Model** is **VPC network**.
- Maximum number of pods of a node (maxPods): Set this parameter when creating a node. It is a configuration item of kubelet.

The maximum number of pods that can be created on a node depends on the minimum value of these parameters.

- For a cluster using the container tunnel network model, the value depends only on [the maximum number of pods on a node](#).
- For a cluster using the VPC network model, the value depends on the minimum value between [the maximum number of pods on a node](#) and [the number of container IP addresses that can be allocated to a node](#), that is,  $\min(\text{maximum number of pods on a node, number of container IP addresses that can be allocated to a node})$ .

## Container Network vs. Host Network

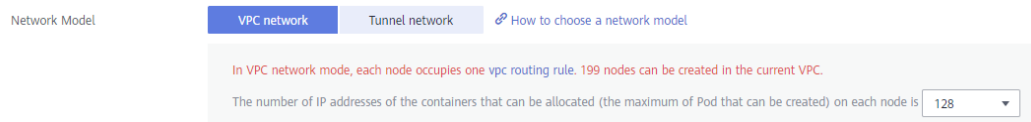
When creating a pod, you can select the container network or host network for the pod.

- Container network (default): **Each pod is assigned an IP address by the cluster networking add-ons, which occupies the IP addresses of the container network.**
- Host network: The pod uses the host network (**hostNetwork: true** needs to be configured for the pod) and occupies the host port. The pod IP address is the host IP address. The pod does not occupy the IP addresses of the container network. To use the host network, you must confirm whether the container ports conflict with the host ports. Do not use the host network unless you know exactly which host port is used by which container.

## Number of Container IP Addresses That Can Be Allocated on a Node

If you select **VPC network** for **Network Model** when creating a CCE cluster, you also need to set the number of container IP addresses that can be allocated to each node, as shown in the following figure.

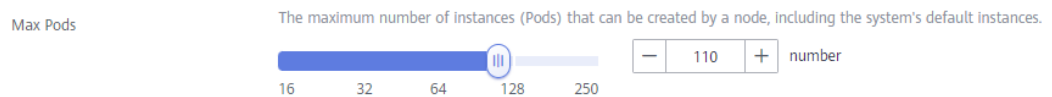
This parameter affects the maximum number of pods that can be created on a node. Each pod occupies an IP address (when the container network is used). If the number of available IP addresses is insufficient, pods cannot be created.



By default, a node occupies three container IP addresses (network address, gateway address, and broadcast address). Therefore, the number of container IP addresses that can be allocated to a node equals the number of selected container IP addresses minus 3. For example, in the preceding figure, **the number of container IP addresses that can be allocated to a node is 125 (128 - 3).**

## Maximum Number of Pods on a Node

When creating a node, you can configure the maximum number of pods that can be created on the node. This parameter is a configuration item of kubelet and determines the maximum number of pods that can be created by kubelet.



### 17.3.10.2 Comparing iptables and IPVS

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 two forwarding modes: iptables and IPVS.

- IPVS allows higher throughput and faster forwarding. This mode 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 a large number of short connections are concurrently sent on the client.

## Notes and Constraints

In IPVS mode, when an ingress and Service use the same load balancer, the ingress cannot be accessed from the nodes and containers in the cluster.

### iptables

iptables is a Linux kernel function that provides a large number of data packet processing and filtering capabilities. It allows flexible sequences of rules to be attached to various hooks in the packet processing pipeline. When iptables is used, kube-proxy implements NAT and load balancing in the NAT pre-routing hook.

kube-proxy is an  $O(n)$  algorithm, in which  $n$  increases with the cluster scale. The cluster scale refers to the number of Services and backend pods.

### IPVS

IP Virtual Server (IPVS) is constructed on top of Netfilter and implements transport-layer load balancing as part of the Linux kernel. IPVS can direct requests

for TCP- and UDP-based services to the real servers, and make services of the real servers appear as virtual services on a single IP address.

In the IPVS mode, kube-proxy uses IPVS load balancing instead of iptables. IPVS is designed to balance loads for a large number of Services. It has a set of optimized APIs and uses optimized search algorithms instead of simply searching for rules from a list.

The complexity of the connection process of IPVS-based kube-proxy is  $O(1)$ . In other words, in most cases, the connection processing efficiency is irrelevant to the cluster scale.

IPVS involves multiple load balancing algorithms, such as round-robin, shortest expected delay, least connections, and various hashing methods. However, iptables has only one algorithm for random selection.

Compared with iptables, IPVS has the following advantages:

1. Provides better scalability and performance for large clusters.
2. Supports better load balancing algorithms than iptables.
3. Supports functions including server health check and connection retries.

### 17.3.10.3 CPU Policy

By default, kubelet uses **CFS quotas** to enforce pod CPU limits. When the node runs many CPU-bound pods, the workload can move to different CPU cores depending on whether the pod is throttled and which CPU cores are available at scheduling time. Many workloads are not sensitive to this migration and thus work fine without any intervention.

When creating a cluster, you can configure the CPU management policy, as shown in the following figure.

CPU Policy



Off: Turn off the function that the Pod has exclusive CPU cores. The advantage is that the CPU shared pool can allocate more cores.

The CPU policy is specified by the kubelet parameter **--cpu-manager-policy**. The following policies are supported:

- Disabled (**none**): the default policy. The **none** policy explicitly enables the existing default CPU affinity scheme, providing no affinity beyond what the OS scheduler does automatically.
- Enabled (**static**): The **static** policy allows containers in **Guaranteed** pods with integer CPU requests access to exclusive CPUs on the node.

For details about CPU policies, see [Control CPU Management Policies on the Node](#).

## 17.4 Nodes

## 17.4.1 Overview

### Introduction

A container cluster consists of a set of worker machines, called nodes, that run containerized applications. A node can be a virtual machine (VM) or a physical machine (PM), depending on your service requirements. The components on a node include kubelet, container runtime, and kube-proxy.

#### NOTE

A Kubernetes cluster consists of master nodes and node nodes. The nodes described in this section refer to **worker nodes**, the computing nodes of a cluster that run containerized applications.

CCE uses high-performance Elastic Cloud Servers (ECSs) as nodes to build highly available Kubernetes clusters.

### Notes

- To ensure node stability, a certain amount of CCE node resources will be reserved for Kubernetes components (such as kubelet, kube-proxy, and docker) based on the node specifications. Therefore, the total number of node resources and the amount of allocatable node resources for your cluster are different. The larger the node specifications, the more the containers deployed on the node. Therefore, more node resources need to be reserved to run Kubernetes components.
- The node networking (such as the VM networking and container networking) is taken over by CCE. You are not allowed to add NICs or change routes. If you modify the networking configuration, the availability of CCE may be affected.

### Node Lifecycle

A lifecycle indicates the node statuses recorded from the time when the node is created through the time when the node is deleted or released.

**Table 17-20** Node statuses

Status	Status Attribute	Description
Available	Stable state	The node is running properly and is connected to the cluster. Nodes in this state can provide services.
Unavailable	Stable state	The node is not running properly. Instances in this state no longer provide services. In this case, perform the operations in <a href="#">Resetting a Node</a> .
Creating	Intermediate state	The node has been created but is not running.

Status	Status Attribute	Description
Installing	Intermediate state	The Kubernetes software is being installed on the node.
Deleting	Intermediate state	The node is being deleted. If this state stays for a long time, an exception occurs.
Stopped	Stable state	The node is stopped properly. A node in this state cannot provide services. You can start the node on the ECS console.
Error	Stable state	The node is abnormal. Instances in this state no longer provide services. In this case, perform the operations in <a href="#">Resetting a Node</a> .

## Mapping between Node OSs and Container Engines

Table 17-21 Node OSs and container engines in CCE clusters

OS	Kernel Version	Container Engine	Container Storage Rootfs	Container Runtime
EulerOS 2.9	4.x	Docker	OverlayFS	runC

### 17.4.2 Buying a Node

#### Scenario

A node is a virtual or physical machine that provides computing resources. Sufficient nodes must be available in your project to ensure that operations, such as creating workloads, can be performed.

#### Prerequisites

- At least one cluster is available. For details on how to create a cluster, see [Buying a CCE Cluster](#).
- A key pair has been created. The key pair will be used for identity authentication upon remote node login.  
If you use a password to log in to a node, skip this step. For details, see [Creating a Key Pair](#).

## Notes and Constraints

- During the node creation, software packages are downloaded from OBS using the domain name. You need to use a private DNS server to resolve the OBS domain name, and configure the subnet where the node resides with a [private DNS server address](#). When you create a subnet, the private DNS server is used by default. If you change the subnet DNS, ensure that the DNS server in use can resolve the OBS domain name.
- Only KVM nodes can be created. Non-KVM nodes cannot be used after being created.
- Once a node is created, its AZ cannot be changed.
- Nodes purchased in the pay-per-use billing mode will be deleted after you delete them on the **Resource Management > Nodes** page on the CCE console. **Yearly/monthly-billed nodes** in a cluster cannot be deleted on the CCE console. You can choose **Billing Center > My Orders** in the upper right corner of the page to unsubscribe from the nodes.
- CCE supports GPUs through an add-on named [gpu-beta](#). You need to install this add-on to use GPU-enabled nodes in your cluster.
- If the cluster network model is **container tunnel network**, the cluster cannot contain VM nodes, and the cluster version must be v1.13.10 or later. (Clusters using this network model can manage only one type of nodes.)
- If the network model is **VPC network**, the cluster version must be v1.11.7 or later. (Clusters using this network model can manage VM nodes and BMS nodes at the same time).

## Procedure

**Step 1** Log in to the CCE console. Use either of the following methods to add a node:

- In the navigation pane, choose **Resource Management > Nodes**. Select the cluster to which the node will belong and click **Buy Node** on the upper part of the node list page.
- In the navigation pane, choose **Resource Management > Clusters**. In the card view of the cluster to which you will add nodes, click **Buy Node**.

**Step 2** Set **Billing mode** to **Pay-per-use** or **Yearly/Monthly**. This section uses the **pay-per-use** billing mode as an example.

**Step 3** Select a region and an AZ.

- **Current Region**: geographic location of the nodes to be created.
- **AZ**: Set this parameter based on the site requirements. An AZ is a physical region where resources use independent power supply and networks. AZs are physically isolated but interconnected through an internal network.

You are advised to deploy worker nodes in different AZs after the cluster is created to make your workloads more reliable. When creating a cluster, you can deploy nodes only in one AZ.

**Figure 17-31** Worker nodes in different AZs

Node Name	Status	Node Pool	Specifications	Allocatable	IP	AZ	Billing Mode	Node Type	Operation
	Available	DefaultPool	2 cores   4 GB   513.large.2	CPU 1.07 Core Memory 0.46 GB	(Private)	AZ2	Pay-per-use Aug 06, 2020 11:40...	Node created	Monitoring More
	Available	DefaultPool	2 cores   4 GB   513.large.2	CPU 0.92 Core Memory 0.21 GB	(Private)	AZ3	Pay-per-use Jun 21, 2020 02:14...	Node created	Monitoring More
	Available	DefaultPool	2 cores   4 GB   513.large.2	CPU 0.92 Core Memory 0.39 GB	(Private) (EIP)	AZ3	Pay-per-use Jun 20, 2020 17:01...	Node created	Monitoring More

**Step 4** Configure node parameters.

- **Node Type**
  - **VM node:** A VM node will be created in the cluster.
- **Node Name:** Enter a node name. A node name contains 1 to 56 characters starting with a lowercase letter and not ending with a hyphen (-). Only lowercase letters, digits, and hyphens (-) are allowed.
- **Specifications:** Select the node specifications based on service requirements. The available node specifications vary depending on AZs.

To ensure node stability, CCE automatically reserves some resources to run necessary system components. For details, see [Formula for Calculating the Reserved Resources of a Node](#).

- **OS:** Select an OS for the node to be created.
  - **Public image:** Select an OS for the node.
 

A public image is a standard, widely used image. It contains an OS and preinstalled public applications and is available to all users.
  - **Private image (OBT):** A private image contains an OS or service data, preinstalled public applications, and the owner's private applications. It is available only to the user who created it. **Private images are supported only for clusters of v1.15 or later.**

If no private image is available, create one by following the instructions provided in .

Reinstalling the OS or modifying OS configurations could make the node unavailable. Exercise caution when performing these operations.

- **System Disk:** Set the system disk space of the worker node. The value ranges from 40GB to 1024 GB. The default value is 40GB.
 

By default, system disks support High I/O (SAS) and Ultra-high I/O (SSD) EVS disks.
- **Data Disk:** Set the data disk space of the worker node. The value ranges from 100 GB to 32,768 GB. The default value is 100 GB. The EVS disk types provided for the data disk are the same as those for the system disk.

**CAUTION**

If the data disk is uninstalled or damaged, the Docker service becomes abnormal and the node becomes unavailable. You are advised not to delete the data disk.

- **LVM:** If this option is selected, CCE data disks are managed by the Logical Volume Manager (LVM). On this condition, you can adjust the disk space

allocation for different resources. This option is selected for the first disk by default and cannot be unselected. You can choose to enable or disable LVM for new data disks.

- This option is selected by default, indicating that LVM management is enabled.
- You can deselect the check box to disable LVM management.

---

**⚠ CAUTION**

- Disk space of the data disks managed by LVM will be allocated according to the ratio you set.
  - When creating a node in a cluster of v1.13.10 or later, if LVM is not selected for a data disk, follow instructions in [Adding a Second Data Disk to a Node in a CCE Cluster](#) to fill in the pre-installation script and format the data disk. Otherwise, the data disk will still be managed by LVM.
  - When creating a node in a cluster earlier than v1.13.10, you must format the data disks that are not managed by LVM. Otherwise, either these data disks or the first data disk will be managed by LVM.
- 
- **Add Data Disk:** Currently, a maximum of two data disks can be attached to a node. After the node is created, you can go to the ECS console to attach more data disks. This function is available only to clusters of certain versions.
  - **Data disk space allocation:** Click [Change Configuration](#) to specify the resource ratio for **Kubernetes Space** and **User Space**. Disk space of the data disks managed by LVM will be allocated according to the ratio you set. This function is available only to clusters of certain versions.
    - **Kubernetes Space:** You can specify the ratio of the data disk space for storing Docker and kubelet resources. Docker resources include the Docker working directory, Docker images, and image metadata. kubelet resources include pod configuration files, secrets, and emptyDirs.

The Docker space cannot be less than 10%, and the space size cannot be less than 60 GB. The kubelet space cannot be less than 10%.


The Docker space size is determined by your service requirements. For details, see [Data Disk Space Allocation](#).
    - **User Space:** You can set the ratio of the disk space that is not allocated to Kubernetes resources and the path to which the user space is mounted.



 NOTE

Note that the mount path cannot be `/`, `/home/paas`, `/var/paas`, `/var/lib`, `/var/script`, `/var/log`, `/mnt/paas`, or `/opt/cloud`, and cannot conflict with the system directories (such as `bin`, `lib`, `home`, `root`, `boot`, `dev`, `etc`, `lost+found`, `mnt`, `proc`, `sbin`, `srv`, `tmp`, `var`, `media`, `opt`, `selinux`, `sys`, and `usr`). Otherwise, the system or node installation will fail.

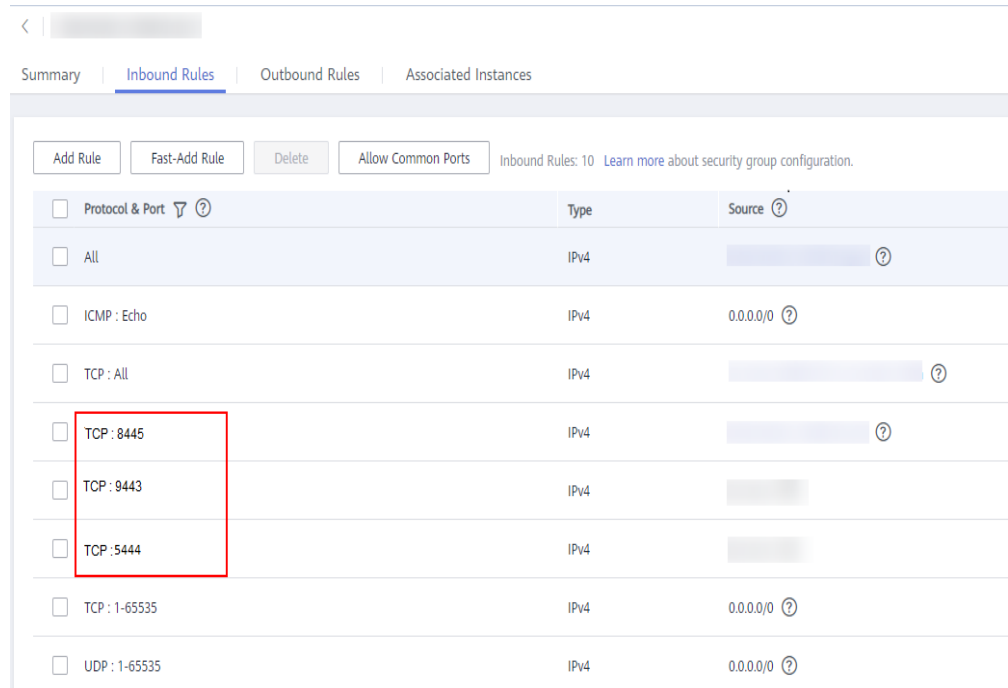
## NOTICE

- The ratio of disk space allocated to the Kubernetes space and user space must be equal to 100% in total. You can click  to refresh the data after you have modified the ratio.
- By default, disks run in the direct-lvm mode. If data disks are removed, the loop-lvm mode will be used and this will impair system stability.

- **VPC:** A VPC where the current cluster is located. This parameter cannot be changed and is displayed only for clusters of v1.13.10-r0 or later.
- **Subnet:** A subnet improves network security by providing exclusive network resources that are isolated from other networks. You can select any subnet in the cluster VPC. Cluster nodes can belong to different subnets.

During the node creation, software packages are downloaded from OBS using the domain name. You need to use a private DNS server to resolve the OBS domain name, and configure the subnet where the node resides with a [private DNS server address](#). When you create a subnet, the private DNS server is used by default. If you change the subnet DNS, ensure that the DNS server in use can resolve the OBS domain name.

When a node is added to an existing cluster, if an extended CIDR block is added to the VPC corresponding to the subnet and the subnet is an extended CIDR block, you need to add the following three security group rules to the master node security group (the group name is in the format of **Cluster name-cce-control-Random number**). These rules ensure that the nodes added to the cluster are available. (This step is not required if an extended CIDR block has been added to the VPC during cluster creation.)



**Step 5 EIP:** an independent public IP address. If the nodes to be created require public network access, select **Automatically assign** or **Use existing**.

An EIP bound to the node allows public network access. EIP bandwidth can be modified at any time. An ECS without a bound EIP cannot access the Internet or be accessed by public networks.

- **Do not use:** A node without an EIP cannot be accessed from public networks. It can be used only as a cloud server for deploying services or clusters on a private network.
- **Automatically assign:** An EIP with specified configurations is automatically assigned to each node. If the number of EIPs is smaller than the number of nodes, the EIPs are randomly bound to the nodes.

Configure the EIP specifications, billing factor, bandwidth type, and bandwidth size as required. When creating an ECS, ensure that the elastic IP address quota is sufficient.

- **Use existing:** Existing EIPs are assigned to the nodes to be created.

**NOTE**

By default, VPC's SNAT feature is disabled for CCE. If SNAT is enabled, you do not need to use EIPs to access public networks. For details about SNAT, see [Custom Policies](#).

**Step 6 Login Mode:** You can use a password or key pair.

- **Password:** The default username is **root**. Enter the password for logging in to the node and confirm the password.

Be sure to remember the password as you will need it when you log in to the node.

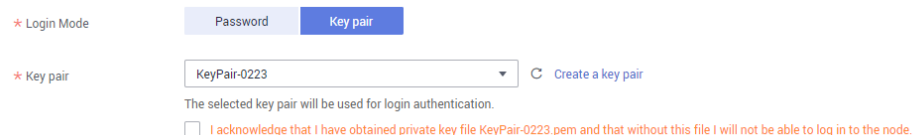
- **Key pair:** Select the key pair used to log in to the node. You can select a shared key.

A key pair is used for identity authentication when you remotely log in to a node. If no key pair is available, click **Create a key pair**.

**NOTICE**

When creating a node using a key pair, IAM users can select only the key pairs created by their own, regardless of whether these users are in the same group. For example, user B cannot use the key pair created by user A to create a node, and the key pair is not displayed in the drop-down list on the CCE console.

**Figure 17-32** Key pair



**Step 7 Advanced ECS Settings** (optional): Click to show advanced ECS settings.


- **ECS Group:** An ECS group logically groups ECSs. The ECSs in the same ECS group comply with the same policy associated with the ECS group.
  - **Anti-affinity:** ECSs in an ECS group are deployed on different physical hosts to improve service reliability.

Select an existing ECS group, or click **Create ECS Group** to create one. After the ECS group is created, click the refresh button.

- **Resource Tags:** By adding tags to resources, you can classify resources. You can create predefined tags in Tag Management Service (TMS). Predefined tags are visible to all service resources that support the tagging function. You can use predefined tags to improve tag creation and migration efficiency. CCE will automatically create the "CCE-Dynamic-Provisioning-Node=node id" tag. A maximum of 5 tags can be added.
- **Agency:** An agency is created by a tenant administrator on the IAM console. By creating an agency, you can share your cloud server resources with another account, or entrust a more professional person or team to manage your resources. To authorize an ECS or BMS to call cloud services, select **Cloud service** as the agency type, click **Select**, and then select **ECS BMS**.
- **Pre-installation Script:** Enter a maximum of 1,000 characters. The script will be executed before Kubernetes software is installed. Note that if the script is incorrect, Kubernetes software may fail to be installed. The script is usually used to format data disks.
- **Post-installation Script:** Enter a maximum of 1,000 characters. The script will be executed after Kubernetes software is installed and will not affect the installation. The script is usually used to modify Docker parameters.
- **Subnet IP Address:** Select **Automatically assign IP address** (recommended) or **Manually assigning IP addresses**.

**NOTE**


When you **manually assign IPs**, the master IP is randomly specified. Therefore, it may conflict with the worker node IP. If you prefer the manual operation, you are advised to select a subnet CIDR block different from that of the master node when setting worker node **subnet**.

**Step 8 Advanced Kubernetes Settings:** (Optional) Click  to show advanced cluster settings.

- **Max Pods:** maximum number of pods that can be created on a node, including the system's default pods. If the cluster uses the **VPC network model**, the maximum value is determined by the number of IP addresses that can be allocated to containers on each node.

This limit prevents the node from being overloaded by managing too many pods. For details, see [Maximum Number of Pods That Can Be Created on a Node](#).

- **Maximum Data Space per Container:** maximum data space that can be used by a container. The value ranges from 10 GB to 500 GB. If the value of this field is larger than the data disk space allocated to Docker resources, the latter will override the value specified here. Typically, 90% of the data disk space is allocated to Docker resources. This parameter is displayed only for clusters of v1.13.10-r0 and later.

**Step 9 Nodes:** The value cannot exceed the management scale you select when configuring cluster parameters. Set this parameter based on service requirements and the remaining quota displayed on the page. Click  to view the factors that affect the number of nodes to be added (depending on the factor with the minimum value).

**Step 10 Validity Duration:** Set this parameter if you select the **yearly/monthly** billing mode.

**Step 11 Click Next: Confirm.** After confirming that the configuration is correct, click **Submit**.

If the billing mode is **Yearly/Monthly**, click **Pay Now** after confirming the configuration, and pay as prompted.

The node list page is displayed. If the node status is **Available**, the node is added successfully. It takes about 6 to 10 minutes to create a node.

 **NOTE**

- A cloud server is automatically created during node creation. If the cloud server fails to be created, a rollback process starts and is charged according to the pricing principles of cloud servers. If there is a rollback fee, go to [Billing Center](#) to unsubscribe from the cloud server.
- Do not delete the security groups and related rules automatically configured during cluster creation. Otherwise, the cluster will exhibit unexpected behavior.

**Step 12 Click Back to Node List.** The node has been created successfully if it changes to the **Available** state.

 **NOTE**

The allocatable resources are calculated based on the resource request value (**Request**), which indicates the upper limit of resources that can be requested by pods on this node, but does not indicate the actual available resources of the node.

The calculation formula is as follows:

- Allocatable CPUs = Total CPUs – Requested CPUs of all pods – Reserved CPUs for other resources
- Allocatable memory = Total memory – Requested memory of all pods – Reserved memory for other resources

----End

## 17.4.3 Accepting ECSs as Nodes into a Cluster

### Scenario

In CCE, you can [add a new node](#) or add existing nodes (ECSs) into your cluster. These nodes can be **billed in yearly/monthly or pay-per-use mode**.

This section describes how to accept existing ECSs as nodes into a cluster on the CCE console.

---

**NOTICE**

- While an ECS is being accepted into a cluster, the operating system of the ECS will be reset to the standard OS image provided by CCE to ensure node stability. The CCE console prompts you to select the operating system and the login mode during the reset.
  - The system disk and data disk of an ECS will be formatted while the ECS is being accepted into a cluster. Ensure that information in the disks has been backed up.
  - While an ECS is being accepted into a cluster, do not perform any operation on the ECS through the ECS console.
- 

### Notes and Constraints

- The cluster version must be v1.13 or later.
- If the password or key has been set when a VM node is created, the VM node can be accepted into a cluster 10 minutes after it is available.

### Prerequisites

An ECS that meets the following conditions can be accepted:

- The node has been purchased and is in the running state, and is not used by other clusters.
- The node to be accepted and the CCE cluster must be in the same VPC. (If the cluster version is earlier than v1.13.10, the node to be accepted and the CCE cluster must be in the same subnet.)

- Only one data disk is attached to the node. The data disk capacity is greater than or equal to 100 GB. If the node has more than one data disk, detach other data disks first.
- The node has 2-core or higher CPU, 4 GB or larger memory, and only one NIC.
- If an enterprise project is used, the node to be managed and the CCE cluster must be in the same enterprise project. Otherwise, resources cannot be identified during management. As a result, the node cannot be managed.

## Procedure


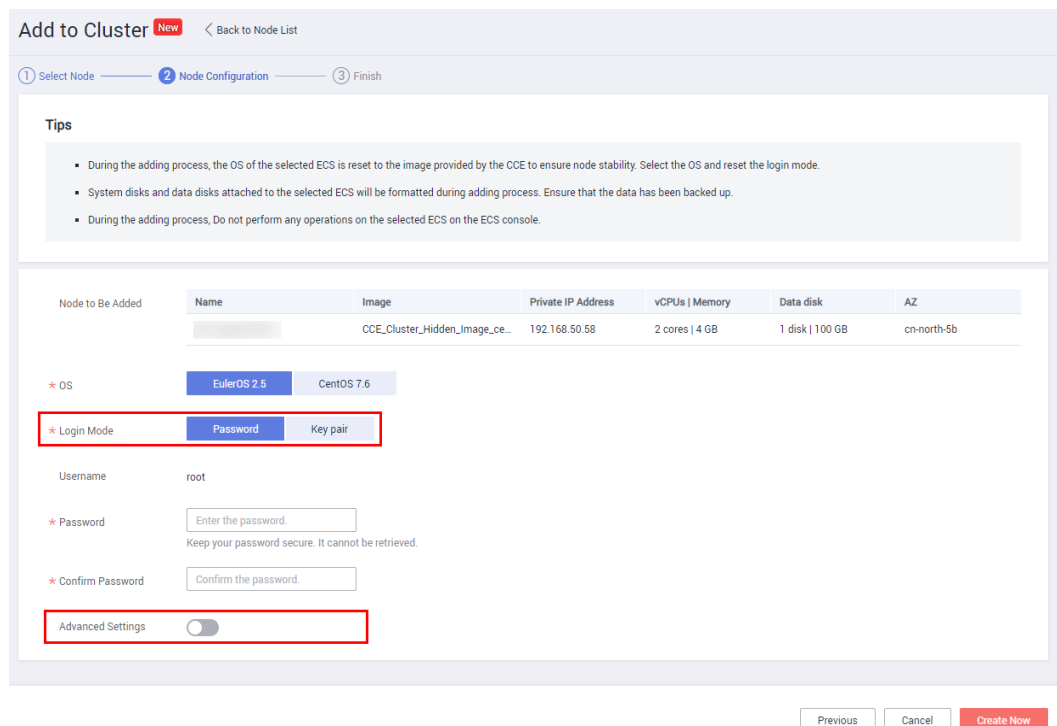
- Step 1** Log in to the CCE console. In the navigation pane, choose **Resource Management > Clusters**. Click **More** for the target cluster, and select **Add to Cluster**.
- Step 2** (Alternative) Log in to the CCE console. In the navigation pane, choose **Resource Management > Nodes**. On the page displayed, select the cluster to which the node belongs, and click **Add to Cluster** in the upper right corner.
- Step 3** If the nodes cannot be added, a red icon is displayed and the number of unqualified nodes is provided. You can move the cursor to  to view the cause.
- Step 4** Select the ECS to be accepted into the cluster, and click **Next**.
- Step 5** On the **Node Configuration** page, select a node login mode, set **Advanced Settings** to **On**, set **Max Pods** and **Disk Allocation**, and click **Create Now**.

Figure 17-33 Node settings



Add to Cluster New < Back to Node List

① Select Node — ② Node Configuration — ③ Finish

**Tips**

- During the adding process, the OS of the selected ECS is reset to the image provided by the CCE to ensure node stability. Select the OS and reset the login mode.
- System disks and data disks attached to the selected ECS will be formatted during adding process. Ensure that the data has been backed up.
- During the adding process, Do not perform any operations on the selected ECS on the ECS console.

Node to Be Added	Name	Image	Private IP Address	vCPUs   Memory	Data disk	AZ
		CCE_Cluster_Hidden_Image_ce...	192.168.50.58	2 cores   4 GB	1 disk   100 GB	cn-north-5b

\* OS: EulerOS 2.5 | CentOS 7.6

\* Login Mode: Password | Key pair

Username: root

\* Password: Enter the password. Keep your password secure. It cannot be retrieved.

\* Confirm Password: Confirm the password.

Advanced Settings:

Previous Cancel Create Now

- Step 6** In the dialog box displayed, confirm the information and click **OK**.

A message is displayed, indicating that the request for accepting the existing node into the selected cluster is successfully submitted.

----End

## 17.4.4 Removing a Node

### Scenario

Removing a node from a cluster in CCE will re-install the node OS and clear CCE components on the node.

Removing a node will not delete the server (ECS) corresponding to the node. You are advised to remove nodes at off-peak hours to avoid impacts on your services.

After a node is removed from the cluster, the node is still running and incurs fees.

### Notes and Constraints

- Nodes can be removed only when the cluster is in the Available or Unavailable state.
- A CCE node can be removed only when it is in the Active, Abnormal, or Error state.
- A CCE node in the Active state can have its OS re-installed and CCE components cleared after it is removed.
- If the OS fails to be re-installed after the node is removed, manually re-install the OS. After the re-installation, log in to the node and run the clearance script to clear CCE components. For details, see [Handling Failed OS Reinstallation](#).

### Precautions

- Removing a node will lead to pod migration, which may affect services. Perform this operation during off-peak hours.
- Unexpected risks may occur during the operation. Back up data in advance.
- While the node is being deleted, the backend will set the node to the unschedulable state.
- After you remove the node and re-install the OS, the original LVM partitions will be cleared and the data managed by LVM will be cleared. Therefore, back up data in advance.

### Procedure

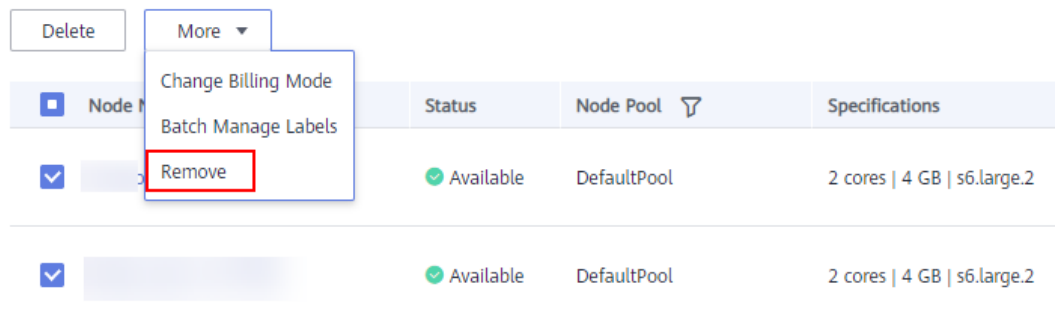
- Step 1** Log in to the CCE console. In the navigation pane, choose **Resource Management > Nodes**. In the same row as the target node, choose **More > Remove**.

Figure 17-34 Removing a node

<input type="checkbox"/>	Node Name	Status	Node Pool	Specifications	Allocatable	IP	AZ	Billing Mode	Node Type	Operation
<input type="checkbox"/>		Available	DefaultPool	2 cores   4 GB   x6.large.2	CPU: 0.88 Core Memory: 0.28 GiB	192.168.9.63 (Private)	AZ3	Pay-per-use Jul. 22, 2021 08:49:0...	Node created	Monitoring More
<input type="checkbox"/>		Available	DefaultPool	2 cores   4 GB   x6.large.2	CPU: 1.48 Core Memory: 1.12 GiB	192.168.9.71 (Private)	AZ3	Pay-per-use Jul. 22, 2021 17:07:1...	Node created	Manage Labels Sync Node Data Reset
<input type="checkbox"/>		Available	DefaultPool	2 cores   4 GB   x6.large.2	CPU: 1.48 Core Memory: 1.12 GiB	192.168.9.56 (Private) 100.85.127.120 (EIP)	AZ3	Pay-per-use Jul. 23, 2021 14:54:1...	Node created	Delete Remove

You can also select multiple nodes and remove them at a time.

**Figure 17-35** Removing multiple nodes at a time



**Step 2** In the dialog box displayed, enter **REMOVE**, configure the login information required for re-installing the OS, and click **Yes**. Wait until the node is removed.

After the node is removed, workload pods on the node are automatically migrated to other available nodes.

----End

## Handling Failed OS Reinstallation

You can perform the following steps to re-install the OS and clear the CCE components on the node if previous attempts fail:

**Step 1** Log in to the management console of the server and re-install the OS.

**Step 2** Log in to the server and run the following commands to clear the CCE components and LVM data:

Write the following scripts to the **clean.sh** file:

```
lsblk
vgs --noheadings | awk '{print $1}' | xargs vgremove -f
pvs --noheadings | awk '{print $1}' | xargs pvremove -f
lvs --noheadings | awk '{print $1}' | xargs -i lvremove -f --select {}
function init_data_disk() {
    all_devices=$(lsblk -o KNAME,TYPE | grep disk | grep -v nvme | awk '{print $1}' | awk '{ print "/dev/"$1}')
    for device in ${all_devices[@]}; do
        isRootDisk=$(lsblk -o KNAME,MOUNTPOINT $device 2>/dev/null | grep -E '[:,space:]'/$ | wc -l )
        if [[ ${isRootDisk} != 0 ]]; then
            continue
        fi
        dd if=/dev/urandom of=${device} bs=512 count=64
    done
    return
}
init_data_disk
lsblk
```

Run the following command:

**bash clean.sh**

----End



## 17.4.5 Logging In to a Node

### Notes and Constraints

- If you use SSH to log in to a node (an ECS), ensure that the ECS already has an EIP (a public IP address).
- Only login to a running ECS is allowed.
- Only the user root can log in to a Linux server.

### Login Modes

You can log in to an ECS in either of the following modes:

- Management console (VNC)  
If an ECS has no EIP, log in to the ECS console and click **Remote Login** in the same row as the ECS.
- SSH  
This mode applies only to ECSs running Linux. Usually, you can use a remote login tool, such as PuTTY, Xshell, and SecureCRT, to log in to your ECS. If none of the remote login tools can be used, log in to the ECS console and click **Remote Login** in the same row as the ECS to view the connection status and running status of the ECS.

#### NOTE

- When you use the Windows OS to log in to a Linux node, set **Auto-login username** to root.
- The CCE console does not support node OS upgrade. Do not upgrade the node OS using the **yum update** command. Otherwise, the container networking components will be unavailable.

**Table 17-22** Linux ECS login modes

EIP Binding	On-Premises OS	Connection Method
Yes	Windows	Use a remote login tool, such as PuTTY or Xshell.
Yes	Linux	Run commands.
Yes/No	Windows/ Linux	Use the remote login function available on the console.

## 17.4.6 Managing Node Labels

You can add different labels to nodes and define different attributes for labels. By using these node labels, you can quickly understand the characteristics of each node.

## Node Label Usage Scenario

Node labels are mainly used in the following scenarios:

- Node management: Node labels are used to classify nodes.
- Affinity and anti-affinity between a workload and node:
  - Some workloads require a large CPU, some require a large memory, some require a large I/O, and other workloads may be affected. In this case, you are advised to add different labels to nodes. When deploying a workload, you can select nodes with specified labels for affinity deployment to ensure the normal operation of the system. Otherwise, node anti-affinity deployment can be used.
  - A system can be divided into multiple modules. Each module consists of multiple microservices. To ensure the efficiency of subsequent O&M, you can add a module label to each node so that each module can be deployed on the corresponding node, does not interfere with other modules, and can be easily developed and maintained on its node.

## Inherent Label of a Node

After a node is created, some fixed labels exist and cannot be deleted. For details about these labels, see [Table 17-23](#).

### NOTE

Do not manually change the inherent labels that are automatically added to a node. If the manually changed value conflicts with the system value, the system value is used.

**Table 17-23** Inherent labels of a node

Key	Description
New: topology.kubernetes.io/ region Old: failure- domain.beta.kubernetes.io/region	Region where the node is located
New: topology.kubernetes.io/zone Old: failure- domain.beta.kubernetes.io/zone	AZ where the node is located
New: node.kubernetes.io/ baremetal Old: failure- domain.beta.kubernetes.io/is- baremetal	Whether the node is a bare metal node <b>false</b> indicates that the node is not a bare metal node.
node.kubernetes.io/container- engine	Container engine Example: <b>docker</b> or <b>containerd</b>
node.kubernetes.io/instance-type	Node specifications
kubernetes.io/arch	Node processor architecture

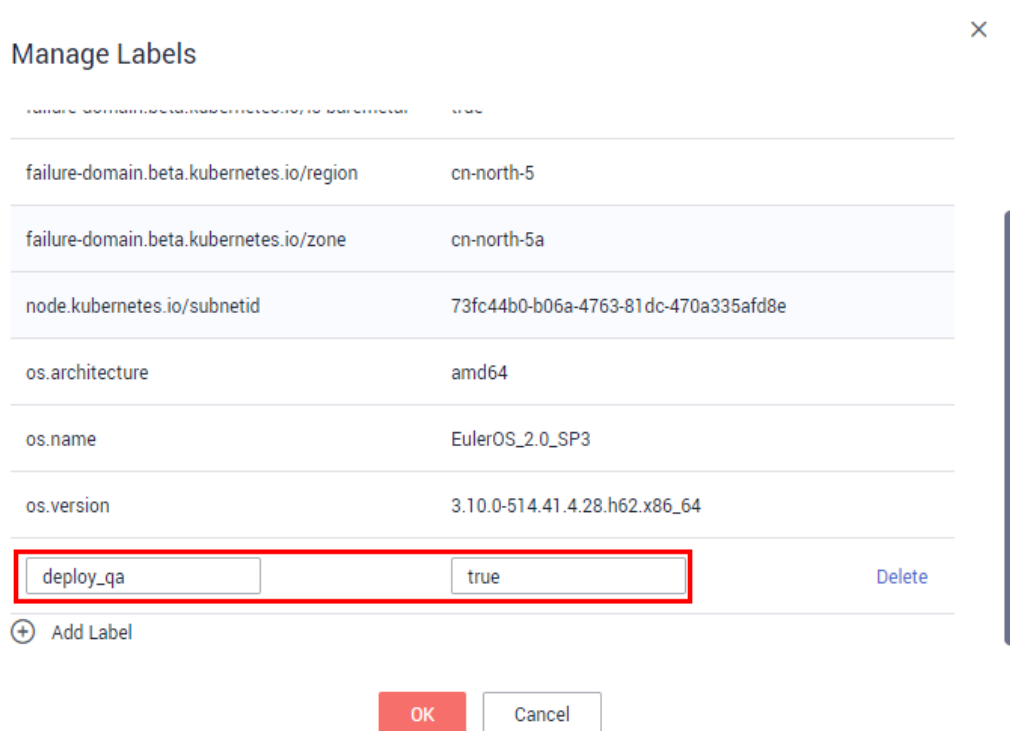
Key	Description
kubernetes.io/hostname	Node name
kubernetes.io/os	Node OS type
node.kubernetes.io/subnetid	ID of the subnet where the node is located
os.architecture	Node processor architecture For example, <b>amd64</b> indicates a AMD64-bit processor.
os.name	Node OS name
os.version	Node OS kernel version
accelerator/huawei-npu	NPU node labels
accelerator	GPU node labels
cce.cloud.com/cce-nodepool	The dedicated label of a node in a node pool

## Adding a Node Label

- Step 1** Log in to the CCE console. In the navigation pane, choose **Resource Management > Nodes**.
- Step 2** In the same row as the node for which you will add labels, choose **Operation > More > Manage Labels**.
- Step 3** In the dialog box displayed, click **Add Label** below the label list, enter the key and value of the label to be added, and click **OK**.

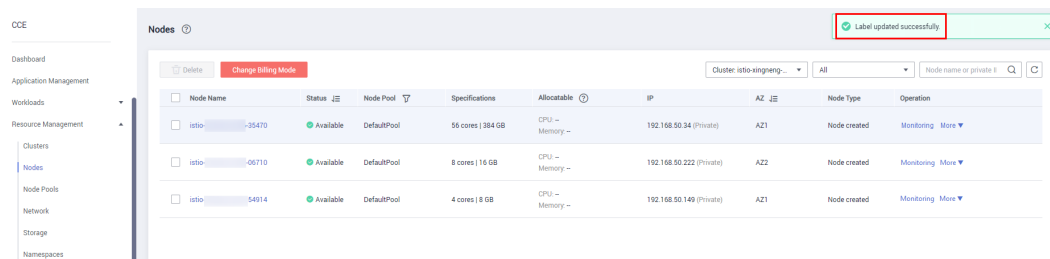
As shown in the figure, the key is **deploy\_qa** and the value is **true**, indicating that the node is used to deploy the QA (test) environment.

**Figure 17-36** Adding a label



**Step 4** After the label is added, click **Manage Labels**. Then, you will see the label that you have added.

**Figure 17-37** Label added successfully



----End

## Deleting a Node Label

Only labels added by users can be deleted. Labels that are fixed on the node cannot be deleted.

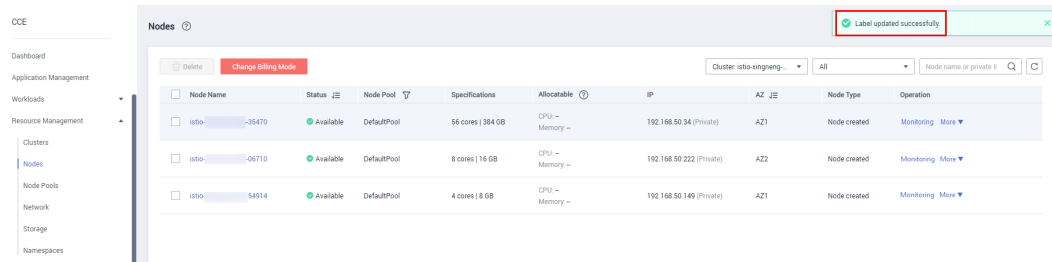
**Step 1** Log in to the CCE console. In the navigation pane, choose **Resource Management > Nodes**.

**Step 2** In the same row as the node for which you will delete labels, choose **Operation > More > Manage Labels**.

**Step 3** Click **Delete**, and then click **OK** to delete the label.

**Label updated successfully** is displayed.

Figure 17-38 Label updated successfully



----End

## Searching for a Node by Label

- Step 1** Log in to the CCE console. In the navigation pane, choose **Resource Management > Nodes**.
- Step 2** In the upper right corner of the node list, click **Search by Label**.
- Step 3** Enter a Kubernetes label to find the target node.

----End

## 17.4.7 Synchronizing Node Data

### Scenario

Each node in a cluster is a cloud server or physical machine. After a cluster node is created, you can change the cloud server name or specifications as required.

Some information about CCE nodes is maintained independently from the ECS console. After you change the name, EIP, billing mode, or specifications of an ECS on the ECS console, you need to **synchronize the ECS information** to the corresponding node on the CCE console. After the synchronization, information on both consoles is consistent.

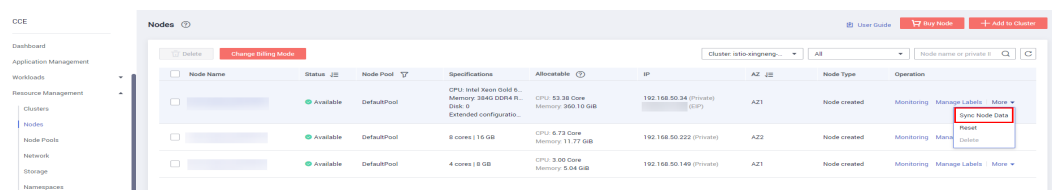
### Procedure

- Step 1** Log in to the CCE console. In the navigation pane, choose **Resource Management > Nodes**.
- Step 2** In the same row as the node whose data will be synchronized, choose **More > Sync Node data**.

#### NOTE

Alternatively, click the node name, and click **Sync Node Data** in the upper right corner of the node details page.

Figure 17-39 Synchronizing node data



After the synchronization is complete, the "Sync success" message is displayed in the upper right corner.

----End

## 17.4.8 Configuring Node Scheduling (Tainting)

Taints enable a node to repel specific pods to prevent these pods from being scheduled to the node.

### Taints

A taint is a key-value pair associated with an effect. The following effects are available:

- **NoSchedule**: No pod will be able to schedule onto the node unless it has a matching toleration. Existing pods will not be evicted from the node.
- **PreferNoSchedule**: Kubernetes prevents pods that cannot tolerate this taint from being scheduled onto the node.
- **NoExecute**: If the pod has been running on a node, the pod will be evicted from the node. If the pod has not been running on a node, the pod will not be scheduled onto the node.

To add a taint to a node, run the **kubectrl taint node *nodename*** command as follows:

```
$ kubectrl get node
NAME          STATUS  ROLES  AGE  VERSION
192.168.10.170 Ready  <none>  73d  v1.19.8-r1-CCE21.4.1.B003
192.168.10.240 Ready  <none>  4h8m v1.19.8-r1-CCE21.6.1.2.B001
$ kubectrl taint node 192.168.10.240 key1=value1:NoSchedule
node/192.168.10.240 tainted
```

To view the taint configuration, run the **describe** and **get** commands as follows:

```
$ kubectrl describe node 192.168.10.240
Name:          192.168.10.240
...
Taints:        key1=value1:NoSchedule
...
$ kubectrl get node 192.168.10.240 -oyaml
apiVersion: v1
...
spec:
  providerID: 06a5ea3a-0482-11ec-8e1a-0255ac101dc2
  taints:
  - effect: NoSchedule
    key: key1
    value: value1
...
```

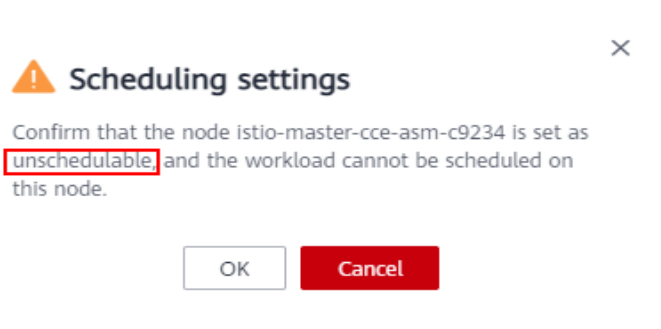
To remove a taint, run the following command with a hyphen (-) added after **NoSchedule**:

```
$ kubectrl taint node 192.168.10.240 key1=value1:NoSchedule-
node/192.168.10.240 untainted
$ kubectrl describe node 192.168.10.240
Name:          192.168.10.240
...
Taints:        <none>
...
```

To configure scheduling settings, log in to the CCE console, choose **Resource Management > Nodes** in the navigation pane, and choose **More > Scheduling settings** in the **Operation** column of a node in the node list.

Node Name	Status	Node Pool	Specifications	Allocatable	IP	AZ	Billing Mode	Node Type	Operation
istio-master-cce-asm-c9234	Available Schedulable	DefaultPool	8 cores   16 GB   s3.2xlarge...	CPU: 7.56 Core Memory: 12.42 GiB	10.10.0.138 (Private)	AZ2	Pay-per-use Sep 17, 2021 17:44...	Node created	Monitoring More ▾ Manage Labels Sync Node Data Reset Delete Remove Scheduling settings
cce-asm-75523	Available Schedulable	DefaultPool	4 cores   8 GB   c3.xlarge.2	CPU: 2.37 Core Memory: 3.55 GiB	10.10.0.71 (Private)	AZ1	Pay-per-use Sep 17, 2021 14:39...	Node created	
istio-master-cce-asm-84964	Available Schedulable	DefaultPool	8 cores   16 GB   s3.2xlarge...	CPU: 7.61 Core Memory: 12.55 GiB	10.10.0.128 (Private)	AZ1	Pay-per-use Sep 17, 2021 17:42...	Node created	

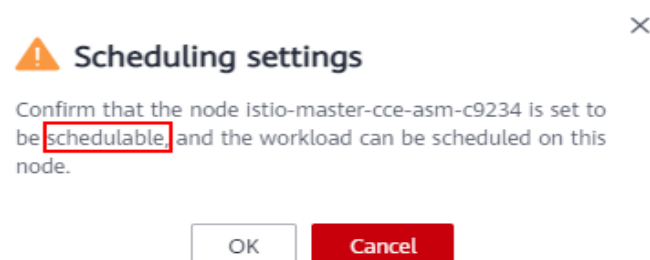
In the dialog box that is displayed, click **OK** to set the node to be unschedulable.



This operation will add a taint to the node. You can use kubectl to view the content of the taint.

```
$ kubectl describe node 192.168.10.240
...
Taints:          node.kubernetes.io/unschedulable:NoSchedule
...
```

On the CCE console, perform the same operations again to remove the taint and set the node to be schedulable.



## Tolerations

Tolerations are applied to pods, and allow (but do not require) the pods to schedule onto nodes with matching taints.

Taints and tolerations work together to ensure that pods are not scheduled onto inappropriate nodes. One or more taints are applied to a node. This marks that the node should not accept any pods that do not tolerate the taints.

Here's an example of a pod that uses tolerations:

```
apiVersion: v1
kind: Pod
metadata:
  name: nginx
  labels:
    env: test
spec:
  containers:
  - name: nginx
    image: nginx
    imagePullPolicy: IfNotPresent
  tolerations:
  - key: "key1"
    operator: "Equal"
    value: "value1"
    effect: "NoSchedule"
```

In the preceding example, the toleration label of the pod is `key1=value1` and the taint effect is `NoSchedule`. Therefore, the pod can be scheduled onto the corresponding node.

You can also configure tolerations similar to the following information, which indicates that the pod can be scheduled onto a node when the node has the taint `key1`:

```
tolerations:
- key: "key1"
  operator: "Exists"
  effect: "NoSchedule"
```

## 17.4.9 Resetting a Node

### Scenario

You can reset a node to modify the node configuration, such as the node OS and login mode.

Resetting a node will reinstall the node OS and the Kubernetes software on the node. If a node is unavailable because you modify the node configuration, you can reset the node to rectify the fault.

### Notes and Constraints

- The cluster version must be v1.13 or later.

### Notes

- Only worker nodes can be reset. If the node is still unavailable after the resetting, delete the node and create a new one.
- **Resetting a node will reinstall the node OS and interrupt workload services running on the node. Therefore, perform this operation during off-peak hours.**
- **Data in the system disk and Docker data disks will be cleared. Back up important data before resetting the node.**
- **When an extra data disk is mounted to a node, data in this disk will be cleared if the disk has not been unmounted before the node reset. To prevent data loss, back up data in advance and mount the data disk again after the node reset is complete.**

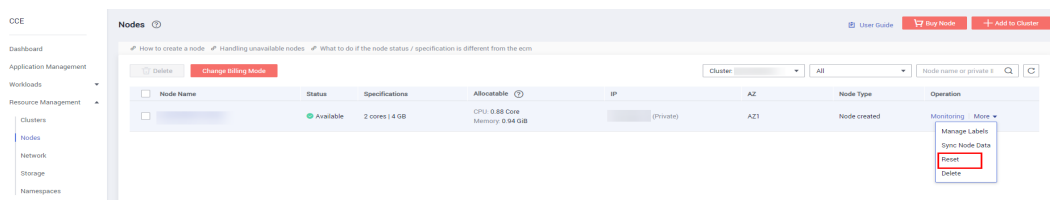


- The IP addresses of the workload pods on the node will change, but the container network access is not affected.
- There is remaining EVS disk quota.
- While the node is being deleted, the backend will set the node to the unschedulable state.

## Procedure

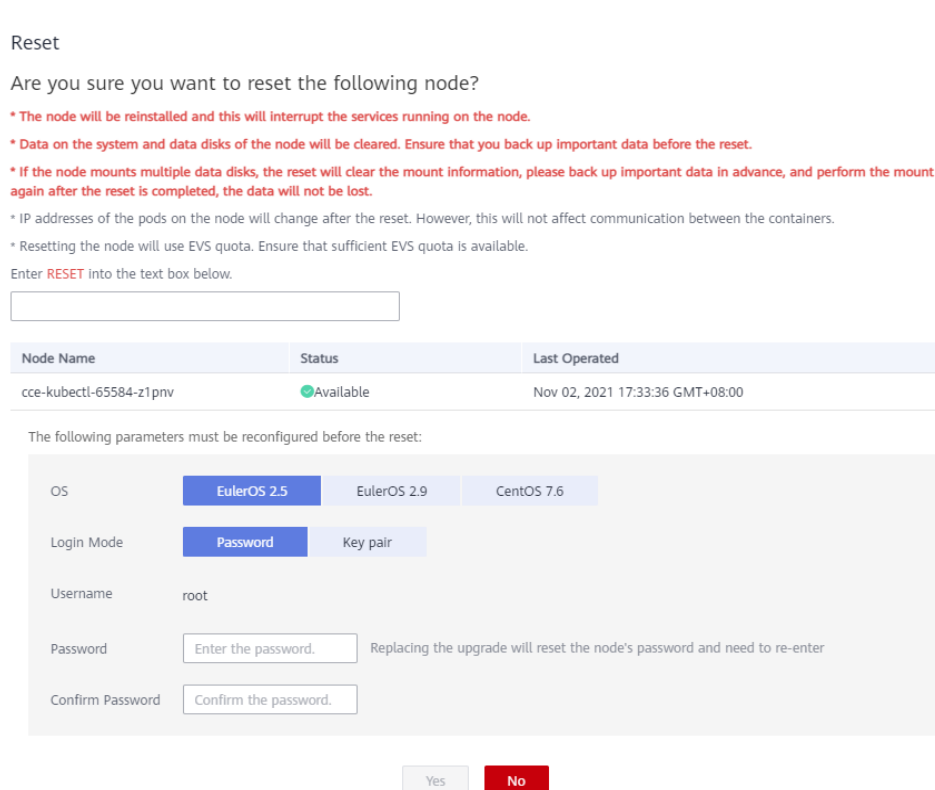
**Step 1** Log in to the CCE console. In the navigation pane, choose **Resource Management > Nodes**. In the same row as the node you will reset, choose **More > Reset**.

**Figure 17-40** Resetting a node



**Step 2** In the dialog box displayed, enter **RESET** and reconfigure the password or key pair for login.

**Figure 17-41** Node resetting dialog box



**Step 3** Click **Yes** and wait until the node is reset.

After the node is reset, pods on it are automatically migrated to other available nodes.

----End

## 17.4.10 Deleting a Node

### Scenario

When a node in a CCE cluster is deleted, services running on the node will also be deleted. Exercise caution when performing this operation.

### Notes and Constraints

- After a CCE cluster is deleted, the ECS nodes in the cluster are also deleted.
- If ECS nodes are billed on a yearly/monthly basis, they cannot be directly deleted. You can choose **Billing Center > My Orders** to unsubscribe from the nodes.

#### NOTICE

For clusters of v1.17.11 or later, after a VM is unsubscribed from or deleted on the ECS console, the corresponding node in the CCE cluster is automatically deleted.

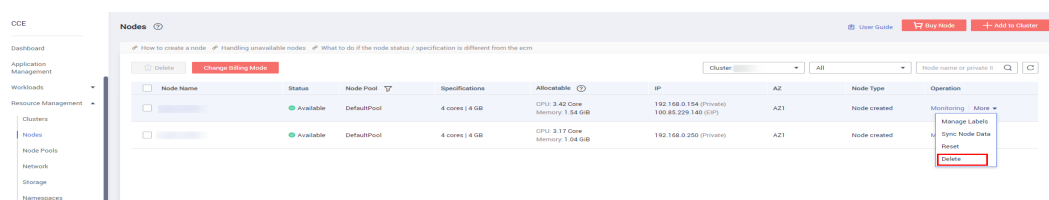
### Notes

- Deleting a node will lead to pod migration, which may affect services. Perform this operation during off-peak hours.
- Unexpected risks may occur during the operation. Back up related data in advance.
- During the operation, the backend will set the node to the unschedulable state.
- Only worker nodes can be deleted.

### Procedure

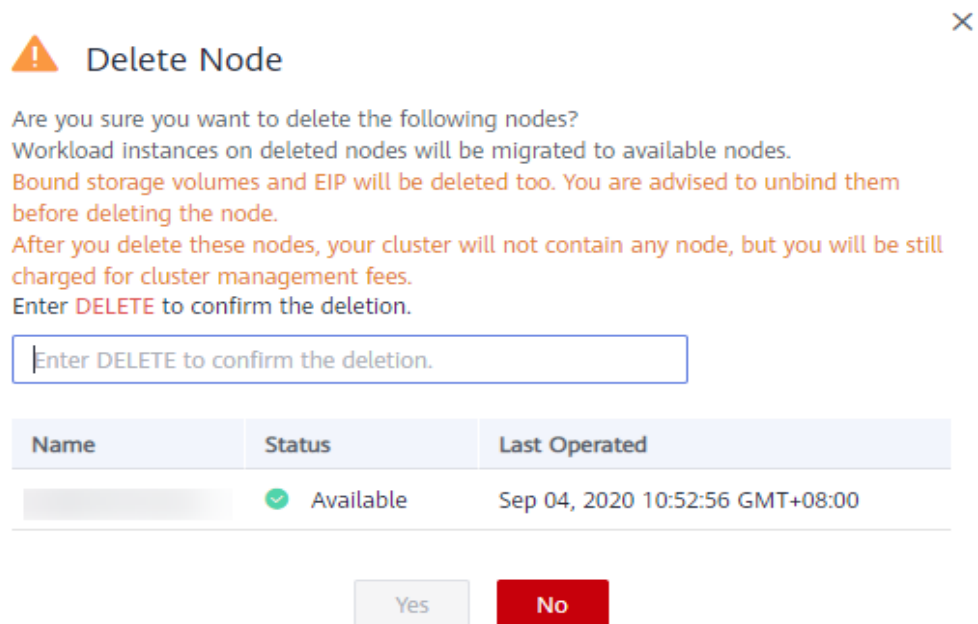
- Step 1** Log in to the CCE console. In the navigation pane, choose **Resource Management > Nodes**. In the same row as the node you will delete, choose **More > Delete**.

Figure 17-42 Deleting a node



- Step 2** In the **Delete Node** dialog box, enter **DELETE** and click **Yes**.

**Figure 17-43** Confirming the deletion



**NOTE**

- After the node is deleted, pods on it are automatically migrated to other available nodes.
- If the disks and EIPs bound to the node are important resources, unbind them first. Otherwise, they will be deleted with the node.
- ECSs are released during node deletion only if they are billed in **pay-per-use** mode.

----End

## 17.4.11 Stopping a Node

### Scenario

After a node in the cluster is stopped, services on the node are also stopped. Before stopping a node, ensure that discontinuity of the services on the node will not result in adverse impacts.

After a node is stopped, it is no longer billed.

### Notes and Constraints

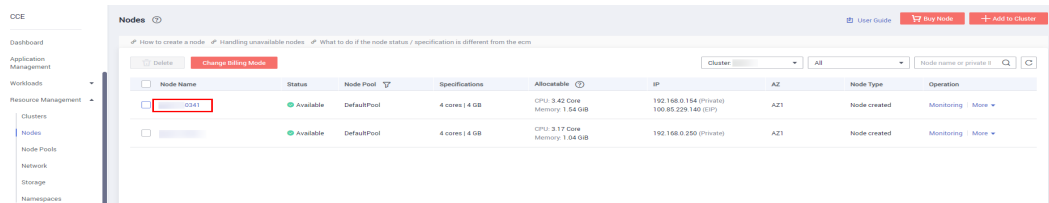
- Deleting a node will lead to pod migration, which may affect services. Therefore, delete nodes during off-peak hours.
- Unexpected risks may occur during node deletion. Back up related data in advance.
- While the node is being deleted, the backend will set the node to the unschedulable state.
- Only worker nodes can be stopped.

## Procedure

**Step 1** Log in to the CCE console. In the navigation pane, choose **Resource Management > Nodes**.

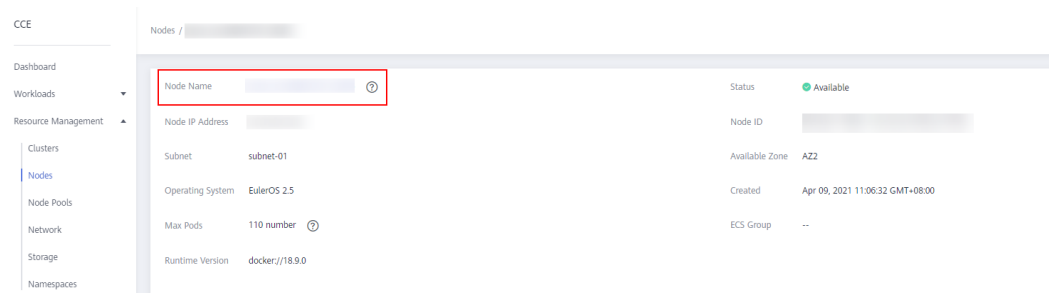
**Step 2** In the node list, click the name of the node to be stopped.

**Figure 17-44** Node list



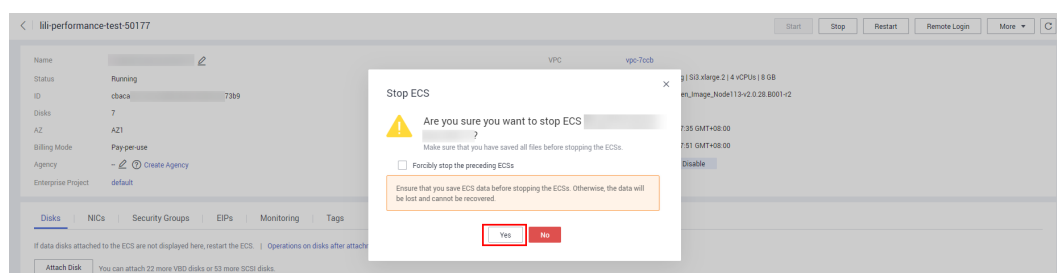
**Step 3** On the node details page displayed, click the node name.

**Figure 17-45** Node details page



**Step 4** In the upper right corner of the ECS details page, click **Stop**. In the **Stop ECS** dialog box, click **Yes**.

**Figure 17-46** ECS details page



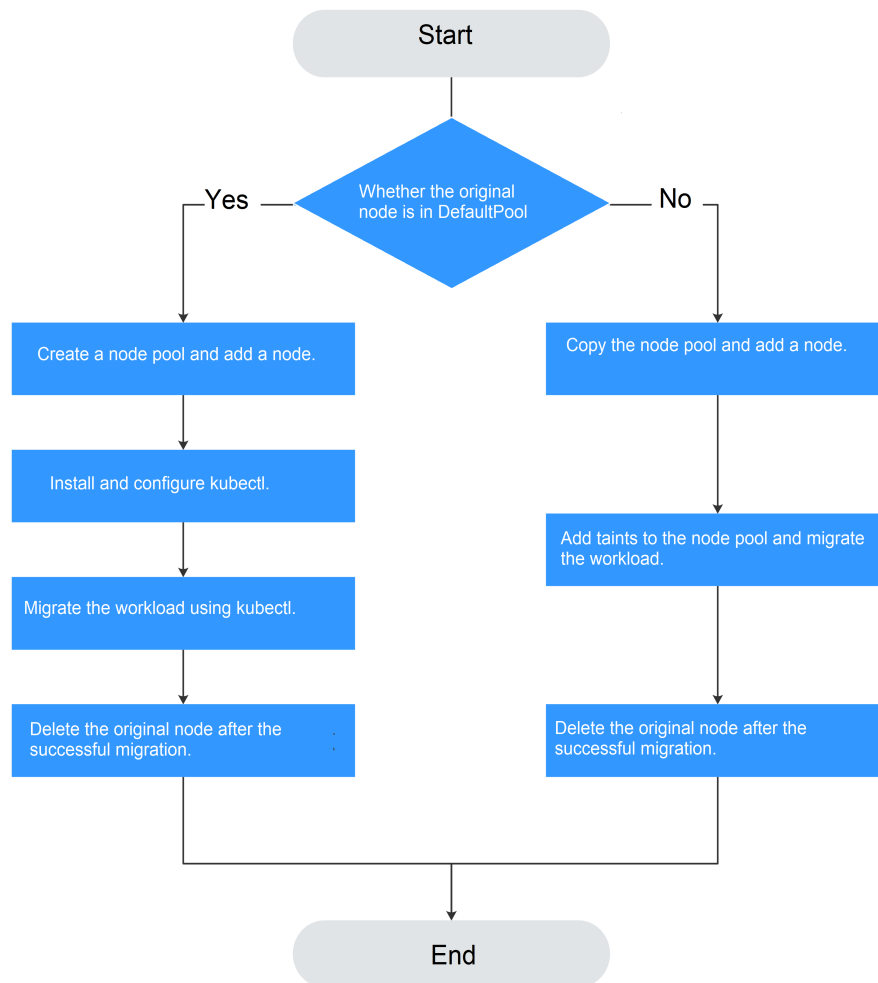
----End

## 17.4.12 Performing Rolling Upgrade for Nodes

### Scenario

In a rolling upgrade, a new node is created, existing workloads are migrated to the new node, and then the old node is deleted. [Figure 17-47](#) shows the migration process.

Figure 17-47 Workload migration



## Notes and Constraints

- The original node and the target node to which the workload is to be migrated must be in the same cluster.
- The cluster must be of v1.13.10 or later.
- The default node pool DefaultPool does not support this configuration.

## Scenario 1: The Original Node Is in DefaultPool

### Step 1 Create a node.

1. Log in to the CCE console. In the navigation pane, choose **Resource Management > Node Pools**.
2. Select the cluster to which the original node belongs.
3. Click **Create Node Pool**, set the following parameters, and modify other parameters as required. For details about the parameters, see [Creating a Node Pool](#).
  - a. **Name:** Enter the name of the new node pool, for example, **nodepool-demo**.
  - b. **Nodes:** In this example, add one node.

- c. **Specifications:** Select node specifications that best suit your needs.
- d. **OS:** Select the operating system (OS) of the nodes to be created.
- e. **Login Mode:** You can use a password or key pair.
  - If the login mode is **Password:** The default username is **root**. Enter the password for logging in to the node and confirm the password. Remember the password as you will need it when you log in to the node.
  - If the login mode is **Key pair**, select a key pair for logging in to the node and select the check box to acknowledge that you have obtained the key file and that without this file you will not be able to log in to the node.

A key pair is used for identity authentication when you remotely log in to a node. If no key pair is available, click **Create a key pair**.
4. Click **Next: Confirm**. Confirm the node pool configuration and click **Submit**.

Go back to the node pool list. In the node list, you can view that the new node pool has been created and is in the Normal state.

**Step 2** Click the name of the node pool. The IP address of the new node is displayed in the node list.

**Step 3** Install and configure kubectl.

1. In the navigation pane of the CCE console, choose **Resource Management > Clusters**, and click **Command Line Tool > Kubectl** under the cluster where the original node is located.
2. On the **Kubectl** tab page of the cluster details page, connect to the cluster as prompted.

**Step 4** Migrate the workload.

1. Add a taint to the node where the workload needs to be migrated out.

```
kubectl taint node [node] key=value:[effect]
```

In the preceding command, *[node]* indicates the IP address of the node where the workload to be migrated is located. The value of *[effect]* can be **NoSchedule**, **PreferNoSchedule**, or **NoExecute**. In this example, set this parameter to **NoSchedule**.

- **NoSchedule:** Pods that do not tolerate this taint are not scheduled on the node; existing pods are not evicted from the node.
- **PreferNoSchedule:** Kubernetes tries to avoid scheduling pods that do not tolerate this taint onto the node.
- **NoExecute:** A pod is evicted from the node if it is already running on the node, and is not scheduled onto the node if it is not yet running on the node.

#### NOTE

To reset a taint, run the **kubectl taint node *[node]* key:[effect]-** command to remove the taint.

2. Safely evicts the workload on the node.

```
kubectl drain [node]
```

In the preceding command, *[node]* indicates the IP address of the node where the workload to be migrated is located.

3. In the navigation pane of the CCE console, choose **Workloads > Deployments**. In the workload list, the status of the workload to be migrated changes from **Running** to **Unready**. If the workload status changes to **Running** again, the migration is successful.

 **NOTE**

During workload migration, if node affinity is configured for the workload, the workload keeps displaying a message indicating that the workload is not ready. In this case, click the workload name to go to the workload details page. On the **Scheduling Policies** tab page, delete the affinity configuration of the original node and click **Add Simple Scheduling Policy** to configure the affinity and anti-affinity policies of the new node. For details, see [Simple Scheduling Policies](#).

After the workload is successfully migrated, you can view that the workload is migrated to the node created in [Step 1](#) on the **Pods** tab page of the workload details page.

**Step 5** Delete the original node.

After the workload is successfully migrated and is running properly, choose **Resource Management > Nodes** to delete the original node.

----End

## Scenario 2: The Original Node Is Not in DefaultPool

**Step 1** Copy the node pool and add nodes to it.

1. Log in to the CCE console. In the navigation pane, choose **Resource Management > Node Pools**.
2. Select the cluster to which the original node belongs.  
In the node pool list, locate the node pool to which the original node belongs.
3. Click **More > Copy** next to the node pool name. On the **Create Node Pool** page, set the following parameters and modify other parameters as required. For details about the parameters, see [Creating a Node Pool](#).
  - **Name:** Enter the name of the new node pool, for example, **nodepool-demo**.
  - **Nodes:** In this example, add one node.
  - **Specifications:** Select node specifications that best suit your needs.
  - **OS:** Select the operating system (OS) of the nodes to be created.
  - **Login Mode:** You can use a password or key pair.
    - If the login mode is **Password:** The default username is **root**. Enter the password for logging in to the node and confirm the password. Remember the password as you will need it when you log in to the node.
    - If the login mode is **Key pair**, select a key pair for logging in to the node and select the check box to acknowledge that you have obtained the key file and that without this file you will not be able to log in to the node.

A key pair is used for identity authentication when you remotely log in to a node. If no key pair is available, click **Create a key pair**.

4. Click **Next: Confirm**. Confirm the node pool configuration and click **Submit**.  
Go back to the node pool list. In the node list, you can view that the new node pool has been created and is in the Normal state.

**Step 2** Click the name of the node pool. The IP address of the new node is displayed in the node list.

**Step 3** Migrate the workload.

1. Click **Edit** on the right of nodepool-demo and set **Taints**.
2. Click **Add Taint**, set **Key** and **Value**, and set **Effect** to **NoExecute**. The value options of **Effect** include **NoSchedule**, **PreferNoSchedule**, or **NoExecute**.
  - **NoSchedule**: Pods that do not tolerate this taint are not scheduled on the node; existing pods are not evicted from the node.
  - **PreferNoSchedule**: Kubernetes tries to avoid scheduling pods that do not tolerate this taint onto the node.
  - **NoExecute**: A pod is evicted from the node if it is already running on the node, and is not scheduled onto the node if it is not yet running on the node.

 **NOTE**

If you need to reset the taint, enter the new values or click **Delete**.

3. Click **Save**.
4. In the navigation pane of the CCE console, choose **Workloads > Deployments**. In the workload list, the status of the workload to be migrated changes from **Running** to **Unready**. If the workload status changes to **Running** again, the migration is successful.

 **NOTE**

During workload migration, if node affinity is configured for the workload, the workload keeps displaying a message indicating that the workload is not ready. In this case, click the workload name to go to the workload details page. On the **Scheduling Policies** tab page, delete the affinity configuration of the original node and click **Add Simple Scheduling Policy** to configure the affinity and anti-affinity policies of the new node. For details, see [Simple Scheduling Policies](#).

After the workload is successfully migrated, you can view that the workload is migrated to the node created in [Step 1](#) on the **Pods** tab page of the workload details page.

**Step 4** Delete the original node.

After the workload is successfully migrated and is running properly, choose **Resource Management > Node Pools** to delete the original node.

----End

## 17.4.13 Formula for Calculating the Reserved Resources of a Node

Some of the resources on the node need to run some necessary Kubernetes system components and resources to make the node as part of your cluster.



Therefore, the total number of node resources and the number of assignable node resources in Kubernetes are different. The larger the node specifications, the more the containers deployed on the node. Therefore, Kubernetes needs to reserve more resources.

To ensure node stability, a certain amount of CCE node resources will be reserved for Kubernetes components (such as kubelet, kube-proxy, and docker) based on the node specifications.

CCE calculates the resources that can be allocated to user nodes as follows:

**Allocatable resources = Total amount - Reserved amount - Eviction threshold**

 **NOTE**

For details about the eviction mechanism and thresholds in Kubernetes, see [What Should I Do If a Pod Fails to Be Evicted?](#)

## Rules for Reserving Node Memory

You can use the following formula calculate how much memory you should reserve for running containers on a node:

Total reserved amount = Reserved memory for system components + Reserved memory for kubelet to manage pods

**Table 17-24** Reservation rules for system components

Total Memory (TM)	Reserved Memory for System Components
TM ≤ 8 GB	0 MB
8 GB < TM ≤ 16 GB	[(TM - 8 GB) × 1024 × 10%] MB
16 GB < TM ≤ 128 GB	[8 GB × 1024 × 10% + (TM - 16 GB) × 1024 × 6%] MB
TM > 128 GB	(8 GB × 1024 × 10% + 112 GB × 1024 × 6% + (TM - 128 GB) × 1024 × 2%) MB

**Table 17-25** Reservation rules for kubelet

Total Memory (TM)	Number of Pods	Reserved Memory for kubelet
TM ≤ 2 GB	-	TM × 25%
TM > 2 GB	0 < Max. pods on a node ≤ 16	700 MB
	16 < Max. pods on a node ≤ 32	[700 + (Max. pods on a node - 16) × 18.75] MB

Total Memory (TM)	Number of Pods	Reserved Memory for kubelet
	32 < Max. pods on a node ≤ 64	[1024 + (Max. pods on a node - 32) x 6.25] MB
	64 < Max. pods on a node ≤ 128	[1230 + (Max. pods on a node - 64) x 7.80] MB
	Max. pods on a node > 128	[1740 + (Max. pods on a node - 128) x 11.20] MB

**NOTICE**

For a small-capacity node, adjust the maximum number of instances based on the site requirements. Alternatively, when creating a node on the CCE console, you can adjust the maximum number of instances for the node based on the node specifications.

## Rules for Reserving Node CPU

**Table 17-26** Node CPU reservation rules

Total CPU Cores (Total)	Reserved CPU Cores
Total ≤ 1 core	Total x 6%
1 core < Total ≤ 2 cores	1 core x 6% + (Total - 1 core) x 1%
2 cores < Total ≤ 4 cores	1 core x 6% + 1 core x 1% + (Total - 2 cores) x 0.5%
Total > 4 cores	1 core x 6% + 1 core x 1% + 2 cores x 0.5% + (Total - 4 cores) x 0.25%

**NOTICE**

CCE reserves an extra 100 MiB for kubelet eviction.

## 17.4.14 Creating a Linux LVM Disk Partition for Docker

### Scenario

This section describes how to check whether there are **available raw disks** and **Linux LVM disk partitions** and how to create Linux LVM disk partitions.

## Prerequisites

To improve the system stability, attach a data disk to Docker and use the direct-lvm mode.

## Procedure

**Step 1** Check whether available raw disks exist on the current node.

1. Log in to the target node as the **root** user.
2. Check the raw disk device.

### **lsblk -l | grep disk**

If the following information is displayed, the raw disks named **xvda** and **xvdb** exist on the node.

```
xvda 202:0 0 40G 0 disk
xvdb 202:16 0 100G 0 disk
```

3. Check whether the raw disk is in use.

### **lsblk /dev/<devicename>**

*devicename* indicates the raw disk name, for example, **xvda** and **xvdb** in the previous step.

Run the **lsblk /dev/xvda** and **lsblk /dev/xvdb** commands. If the following information is displayed, **xvda** has been partitioned and used while **xvdb** is available. If no raw disk is available, bind an EVS disk to the node. It is advised that the disk space be no less than 80 GB.

```
NAME MAJ:MIN RM SIZE RO TYPE MOUNTPOINT
xvda 202:0 0 40G 0 disk
├─xvda1 202:1 0 100M 0 part /boot
└─xvda2 202:2 0 39.9G 0 part /
NAME MAJ:MIN RM SIZE RO TYPE MOUNTPOINT
xvdb 202:16 0 100G 0 disk
```

**Step 2** Check whether there are partitions available. Currently, only Linux LVM partitions are supported.

1. Log in to the target node as the **root** user.
2. Check the partition whose system type is Linux LVM.

### **fdisk -l 2>>/dev/null | grep "Linux LVM"**

If the following information is displayed, two Linux LVM partitions, **/dev/nvme0n1p1** and **/dev/nvme0n1p2**, exist in the system.

```
/dev/nvme0n1p1 1 204800 204800 209715200 8e Linux LVM
/dev/nvme0n1p2 204801 409600 204800 209715200 8e Linux LVM
```

3. Check whether the partition is in use.

### **lsblk <partdevice>**

*<partdevice>* is the Linux LVM partition found in the previous step.

In this example, run the **lsblk/dev/nvme0n1p1** and **lsblk/dev/nvme0n1p2** commands. If the following information is displayed, partition **nvme0n1p1** is in use while **nvme0n1p2** is available.

```
NAME MAJ:MIN RM SIZE RO TYPE MOUNTPOINT
nvme0n1p1 259:3 0 200G 0 part
├─vgpaas-thinpool_tdata 251:8 0 360G 0 lvm
└─vgpaas-thinpool 251:10 0 360G 0 lvm
NAME MAJ:MIN RM SIZE RO TYPE MOUNTPOINT
nvme0n1p2 259:1 0 100G 0 part
```

If no AZ is available, perform [Step 3](#) to create a partition for Docker.

**Step 3** Create a Linux LVM disk partition for Docker.

1. Run the following command to create a disk partition. **devicename** indicates the available raw disk name, for example, **xvdb** in [Step 1](#).

```
fdisk /dev/devicename
```

2. Enter **n** to create a new partition. Enter **p** to display the primary partition number. Enter **4** to indicate the fourth primary partition.

**Figure 17-48** Creating a partition

```
Command (m for help): n
Partition type
  p   primary (0 primary, 0 extended, 4 free)
  e   extended (container for logical partitions)
Select (default p): p
Partition number (1-4, default 1): 4
```

3. Configure the start and last sectors as follows for example:  
Start sector (1048578048-4294967295, 1048578048 by default):  
1048578048  
Last sector, +sector or size {K, M, or G} (1048578048-4294967294, 4294967294 by default): +100G

This configuration indicates that partition 4 has been set to the Linux type and the size is 100 GiB.

4. Enter **t** to change the partition system type. Enter the hex code **8e** when prompted to change the system type to Linux LVM.

```
Command (enter m to obtain help): t
Partition ID (ranging from 1 to 4, 4 by default): 4
Hex code (enter L to list all codes): 8e
This configuration changes the type of the partition Linux to Linux LVM.
```

5. Enter **w** to save the modification.

```
Command (enter m to obtain help): w
The partition table has been altered!
```

6. Run the **partprobe** command to refresh the disk partition.

----End

## 17.4.15 Data Disk Space Allocation

When creating a node, you need to configure data disks for the node.

The data disk is divided into Kubernetes space and user space. The user space defines the space that is not allocated to Kubernetes in the local disk. The Kubernetes space consists of the following two parts:

- Docker space (90% by default): stores Docker working directories, Docker image data, and image metadata.
- kubelet space (10% by default): stores pod configuration files, secrets, and mounted storage such as emptyDir volumes.

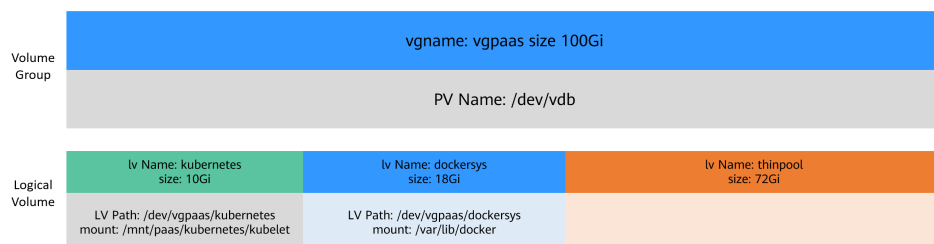
The Docker space size affects image download and container startup and running. This section describes how the Docker space is used so that you can configure the Docker space accordingly.

## Docker Space Description

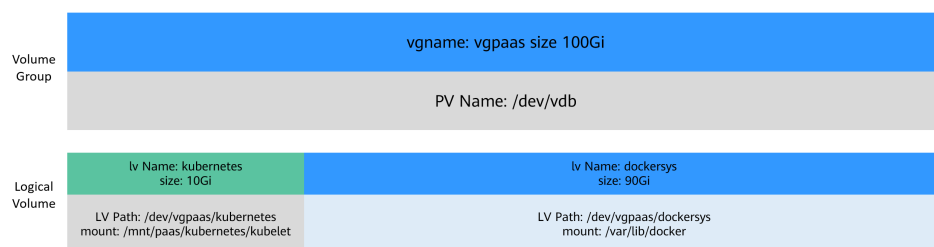
By default, a data disk, 100 GB for example, is divided as follows (depending on the container storage Rootfs):

- Rootfs (Device Mapper)
  - The `/var/lib/docker` directory is used as the Docker working directory and occupies 20% of the Docker space by default. (Space size of the `/var/lib/docker` directory = Data disk space x 90% x 20%)
  - The thin pool is used to store Docker image data, image metadata, and container data, and occupies 80% of the Docker space by default. (Thin pool space = Data disk space x 90% x 80%)

The thin pool is dynamically mounted. You can view it by running the `lsblk` command on a node, but not the `df -h` command.



- Rootfs (OverlayFS): No separate thinpool. The entire Docker space is in the `/var/lib/docker` directory.



Using rootfs for container storage in CCE

- CCE cluster: and EulerOS 2.9 nodes use OverlayFS.

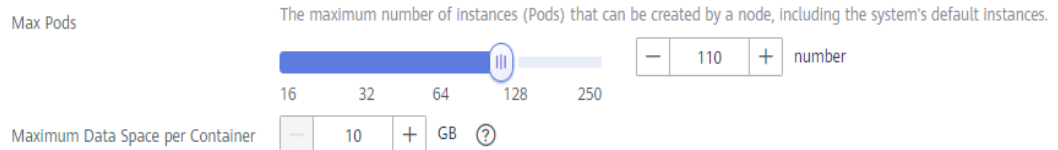
You can log in to the node and run the `docker info` command to view the storage engine type.

```
# docker info
Containers: 20
Running: 17
Paused: 0
Stopped: 3
Images: 16
Server Version: 18.09.0
Storage Driver: devicemapper
```

## Docker Space and Containers

The number of pods and the space configured for each container determine whether the Docker space of a node is sufficient.

## Advanced Kubernetes Settings ^



The Docker space should be greater than the total disk space used by containers. Formula: **Docker space > Number of containers x Available data space for a single container (basesize)**

**When Device Mapper is used**, although you can limit the size of the `/home` directory of a single container (to 10 GB by default), all containers on the node still share the thin pool of the node for storage. They are not completely isolated. When the sum of the thin pool space used by certain containers reaches the upper limit, other containers cannot run properly.

In addition, after a file is deleted in the `/home` directory of the container, the thin pool space occupied by the file is not released immediately. Therefore, even if **basesize** is set to 10 GB, the thin pool space occupied by files keeps increasing until 10 GB when files are created in the container. The space released after file deletion will be reused but after a while. If **the number of containers on the node multiplied by basesize** is greater than the thin pool space size of the node, there is a possibility that the thin pool space has been used up.

## Garbage Collection Policies for Container Images

When the Docker space is insufficient, image garbage collection is triggered.

The policy for garbage collecting images takes two factors into consideration: **HighThresholdPercent** and **LowThresholdPercent**. Disk usage above the high threshold (default: 85%) will trigger garbage collection. The garbage collection will delete least recently used images until the low threshold (default: 80%) has been met.

## Docker Space Configuration Suggestions

- The Docker space should be greater than the total disk space used by containers. Formula: **Docker space > Number of containers x Available data space for a single container (basesize)**
- You are advised to create and delete files of containerized services in local storage volumes (such as `emptyDir` and `hostPath` volumes) or cloud storage directories mounted to the containers. In this way, the thin pool space is not occupied. `emptyDir` volumes occupy the kubelet space. Therefore, properly plan the size of the kubelet space.
- Docker uses the OverlayFS storage mode. This mode is used in Ubuntu 18.04 nodes in CCE clusters by default. You can deploy services on these nodes to prevent that the disk space occupied by files created or deleted in containers is not released immediately.

## 17.4.16 Adding a Second Data Disk to a Node in a CCE Cluster

You can use the pre-installation script feature to configure CCE cluster nodes (ECSs).

 NOTE

- When creating a node in a cluster of v1.13.10 or later, if a data disk is not managed by LVM, follow instructions in this section to format the data disk before adding the disk. Otherwise, the data disk will still be managed by LVM.
- When creating a node in a cluster earlier than v1.13.10, you must format the data disks that are not managed by LVM. Otherwise, either these data disks or the first data disk will be managed by LVM.

Before using this feature, write a script that can format data disks and save it to your OBS bucket. This script must be executed by user **root**.

**Input Parameters**

1. Set the script name to **formatdisk.sh**, save the script to your OBS bucket, and obtain the address of the script in OBS.
2. You need to specify the size of the Docker data disk (the data disk managed by LVM is called the Docker data disk). The size of the Docker disk must be different from that of the second disk. For example, the Docker data disk is 100 GB and the new disk is 110 GB.
3. Set the mount path of the second data disk, for example, **/data/code**.

Run the following command in the pre-installation script to format the disk:

```
cd /tmp;curl -k -X GET OBS bucket address /formatdisk.sh -1 -O;fdisk -l;sleep 30;bash -x formatdisk.sh  
100 /data/code;fdisk -l
```

**Example script (formatdisk.sh):**

```
dockerdisksize=$1  
mountdir=$2  
systemdisksize=40  
i=0  
while [ 20 -gt $i ]; do  
    echo $i;  
    if [ $(lsblk -o KNAME,TYPE | grep disk | grep -v nvme | awk '{print $1}' | awk '{ print "/dev/"$1}' | wc -l) -  
ge 3 ]; then  
        break  
    else  
        sleep 5  
    fi;  
    i=$((i+1))  
done  
all_devices=$(lsblk -o KNAME,TYPE | grep disk | grep -v nvme | awk '{print $1}' | awk '{ print "/dev/"$1}')  
for device in ${all_devices[@]}; do  
    isRawDisk=$(lsblk -n $device 2>/dev/null | grep disk | wc -l)  
    if [[ ${isRawDisk} > 0 ]]; then  
        # is it partitioned ?  
        match=$(lsblk -n $device 2>/dev/null | grep -v disk | wc -l)  
        if [[ ${match} > 0 ]]; then  
            # already partited  
            [[ -n "${DOCKER_BLOCK_DEVICES}" ]] && echo "Raw disk ${device} has been partition, will skip  
this device"  
            continue  
        fi  
    else  
        isPart=$(lsblk -n $device 2>/dev/null | grep part | wc -l)  
        if [[ ${isPart} -ne 1 ]]; then  
            # not partied  
            [[ -n "${DOCKER_BLOCK_DEVICES}" ]] && echo "Disk ${device} has not been partition, will skip this  
device"  
            continue  
        fi  
        # is used ?  
        match=$(lsblk -n $device 2>/dev/null | grep -v part | wc -l)
```

```
if [[ ${match} > 0 ]]; then
    # already used
    [[ -n "${DOCKER_BLOCK_DEVICES}" ]] && echo "Disk ${device} has been used, will skip this device"
    continue
fi
isMount=$(lsblk -n -o MOUNTPOINT $device 2>/dev/null)
if [[ -n $isMount ]]; then
    # already used
    [[ -n "${DOCKER_BLOCK_DEVICES}" ]] && echo "Disk ${device} has been used, will skip this device"
    continue
fi
isLvm=$(sfdisk -lqL 2>>/dev/null | grep $device | grep "8e.*Linux LVM")
if [[ ! -n $isLvm ]]; then
    # part system type is not Linux LVM
    [[ -n "${DOCKER_BLOCK_DEVICES}" ]] && echo "Disk ${device} system type is not Linux LVM, will
skip this device"
    continue
fi
fi
block_devices_size=$(lsblk -n -o SIZE $device 2>/dev/null | awk '{ print $1}')
if [[ ${block_devices_size}"x" != "${dockerdisksize}Gx" ]] && [[ ${block_devices_size}"x" != "${
systemdisksize}Gx" ]]; then
echo "n
p
1

w
" | fdisk $device
    mkfs -t ext4 ${device}1
    mkdir -p $mountdir
    uuid=$(blkid ${device}1 | awk '{print $2}')
    echo "${uuid} $mountdir ext4 noatime 0 0" | tee -a /etc/fstab >/dev/null
    mount $mountdir
fi
done
```

#### NOTE

If the preceding example cannot be executed, use the dos2unix tool to convert the format.

## 17.5 Node Pools

### 17.5.1 Node Pool Overview

#### Introduction

CCE introduces node pools to help you better manage nodes in Kubernetes clusters. A node pool contains one node or a group of nodes with identical configuration in a cluster.

You can create custom node pools on the CCE console. With node pools, you can quickly create, manage, and destroy nodes without affecting the cluster. All nodes in a custom node pool have identical parameters and node type. You cannot configure a single node in a node pool; any configuration changes affect all nodes in the node pool.

You can also use node pools for auto scaling.

- When a pod in a cluster cannot be scheduled due to insufficient resources, scale-out can be automatically triggered.

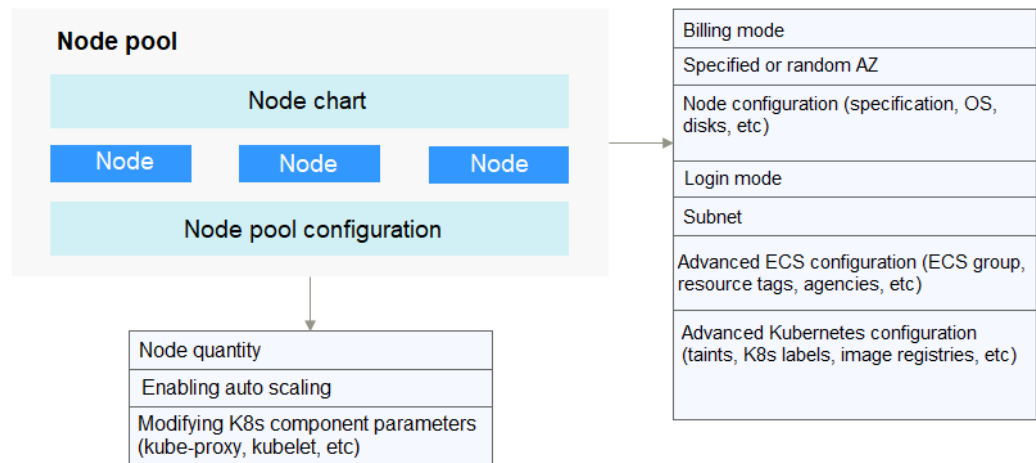


- When there is an idle node or a monitoring metric threshold is met, scale-in can be automatically triggered.

This section describes how node pools work in CCE and how to create and manage node pools.

## Node Pool Architecture

Figure 17-49 Overall architecture of a node pool



Generally, all nodes in a node pool have the following same attributes:

- Node OS
- Startup parameters of Kubernetes components on a node
- User-defined startup script of a node
- **K8S Labels and Taints**

CCE provides the following extended attributes for node pools:

- Node pool OS
- Maximum number of pods on each node in a node pool

## Description of DefaultPool

DefaultPool is not a real node pool. It only **classifies** nodes that are not in any node pool. These nodes are directly created on the console or by calling APIs. DefaultPool does not support any node pool functions, including scaling and parameter configuration. DefaultPool cannot be edited, deleted, expanded, or auto scaled, and nodes in it cannot be migrated.

## Applicable Scenarios

When a large-scale cluster is required, you are advised to use node pools to manage nodes.

The following table describes multiple scenarios of large-scale cluster management and the functions of node pools in each scenario.

**Table 17-27** Using node pools for different management scenarios

Scenario	Function
Multiple heterogeneous nodes (with different models and configurations) in the cluster	Nodes can be grouped into different pools for management.
Frequent node scaling required in a cluster	Node pools support auto scaling to dynamically add or reduce nodes.
Complex application scheduling rules in a cluster	Node pool tags can be used to quickly specify service scheduling rules.

## Functions and Precautions

Function	Description	Notes
Creating a node pool	Add a node pool.	It is recommended that a cluster contain no more than 100 node pools.
Deleting a node pool	Deleting a node pool will delete nodes in the pool. Pods on these nodes will be automatically migrated to available nodes in other node pools.	If pods in the node pool have a specific node selector and none of the other nodes in the cluster satisfies the node selector, the pods will become unschedulable.
Enabling auto scaling for a node pool	After auto scaling is enabled, nodes will be automatically created or deleted in the node pool based on the cluster loads.	You are advised not to store important data on nodes in a node pool because after auto scaling, data cannot be restored as nodes may be deleted.
Enabling auto scaling for a node pool	After auto scaling is disabled, the number of nodes in a node pool will not automatically change with the cluster loads.	/
Adjusting the size of a node pool	The number of nodes in a node pool can be directly adjusted. If the number of nodes is reduced, nodes are randomly removed from the current node pool.	After auto scaling is enabled, you are not advised to manually adjust the node pool size.

Function	Description	Notes
Changing node pool configurations	You can modify the node pool name, node quantity, Kubernetes labels, taints, and resource tags.	The modified Kubernetes labels and taints will apply to all nodes in the node pool, which may cause pod re-scheduling. Therefore, exercise caution when performing this operation.
Adding an existing node to a node pool	Nodes that do not belong to the cluster can be added to a node pool. The following requirements must be met: <ul style="list-style-type: none"> <li>The node to be added and the CCE cluster are in the same VPC and subnet.</li> <li>The node is not used by other clusters and has the same configurations (such as specifications and billing mode) as the node pool.</li> </ul>	Unless required, you are not advised to add existing nodes. You are advised to create a node pool.
Removing a node from a node pool	Nodes in a node pool can be migrated to the default node pool of the same cluster.	Nodes in the default node pool cannot be migrated to other node pools, and nodes in a user-created node pool cannot be migrated to other user-created node pools.
Cloning a node pool	You can copy the configuration of an existing node pool to create a new node pool.	/
Setting Kubernetes parameters	You can configure core components with fine granularity.	<ul style="list-style-type: none"> <li>This function is supported only for clusters of v1.15 and later. It is not displayed for versions earlier than v1.15</li> <li>The default node pool DefaultPool does not support this type of configuration.</li> </ul>

## Deploying a Workload in a Specified Node Pool

When creating a workload, you can constrain pods to run in a specified node pool.

For example, on the CCE console, you can set the affinity between the workload and the node on the **Scheduling Policies** tab page on the workload details page to forcibly deploy the workload to a specific node pool. In this way, the workload

runs only on nodes in the node pool. If you need to better control where the workload is to be scheduled, you can use affinity or anti-affinity policies between workloads and nodes described in [Scheduling Policy Overview](#).

For example, you can use container's resource request as a nodeSelector so that workloads will run only on the nodes that meet the resource request.

If the workload definition file defines a container that requires four CPUs, the scheduler will not choose the nodes with two CPUs to run workloads.

## Related Operations

You can log in to the CCE console and refer to the following sections to perform operations on node pools:

- [Creating a Node Pool](#)
- [Managing a Node Pool](#)
- [Creating a Deployment](#)
- [Workload-Node Affinity](#)

## 17.5.2 Creating a Node Pool

### Scenario

This section describes how to create a node pool and perform operations on the node pool. For details about how a node pool works, see [Node Pool Overview](#).

### Notes and Constraints

- The autoscaler add-on needs to be installed for node auto scaling. For details about the add-on installation and parameter configuration, see [autoscaler](#).
- Auto scaling is available only for pay-per-use node pools, not for those billed by year or month. Changing nodes in a pay-per-use node pool to yearly/monthly may cause auto scaling to fail. You need to migrate these nodes to DefaultPool by following the instructions in [Migrating a Node](#).

### Procedure

To create a node pool in a cluster, perform the following steps:

**Step 1** Log in to the CCE console. In the navigation pane, choose **Resource Management > Node Pools**.

**Step 2** In the upper right corner of the page, click **Create Node Pool**.


**Step 3** Set node pool parameters.

- **Billing Mode**

Only pay-per-use is supported. In this billing mode, resources are billed by hour.

After a node pool is created, the billing mode of resources in the created node pool cannot be changed to the yearly/monthly billing, while this change is allowed for resources in the default node pool. You can migrate resources

from a self-built node pool to the default node pool and then change the billing mode to yearly/monthly billing. For details, see [Migrating a Node](#).

- **Current Region:** geographic location of the node pool to be created.  
To minimize network latency and resource access time, select the region nearest to your node pool. Cloud resources are region-specific and cannot be used across regions over internal networks.
- **Name:** name of the new node pool. By default, the name is in the format of *Cluster name-nodepool-Random number*. You can also use a custom name.
- **Node Type:** Currently, only VM nodes are supported.
- **Nodes:** number of nodes to be purchased for this node pool. The value cannot exceed the maximum number of nodes that can be managed by the cluster.
- **Auto Scaling:**
  - By default, this parameter is disabled.
  - After you enable autoscaler by clicking , nodes in the node pool will be automatically created or deleted based on cluster loads.
    - **Maximum Nodes and Minimum Nodes:** You can set the maximum and minimum number of nodes to ensure that the number of nodes to be scaled is within a proper range.
    - **Priority:** Set this parameter based on service requirements. A larger value indicates a higher priority. For example, if this parameter is set to **1** and **4** respectively for node pools A and B, B has a higher priority than A. If the priorities of multiple node pools are set to the same value, for example, **2**, the node pools are not prioritized and the system performs scaling based on the minimum resource waste principle.

#### NOTE

CCE selects a node pool for auto scaling based on the following policies:

1. CCE uses algorithms to determine whether a node pool meets the conditions to allow scheduling of a pod in pending state, including whether the node resources are greater than requested by the pod, and whether the nodeSelect, nodeAffinity, and taints meet the conditions. In addition, the node pools that fail to be scaled (due to insufficient resources or other reasons) and are still in the 15-minute cool-down interval are filtered.
  2. If multiple node pools meet the scaling requirements, the system checks the priority of each node pool and selects the node pool with the highest priority for scaling. The value ranges from 0 to 100 and the default priority is 0. The value 100 indicates the highest priority, and the value 0 indicates the lowest priority.
  3. If multiple node pools have the same priority or no priority is configured for them, the system selects the node pool that will consume the least resources based on the configured VM specification.
  4. If the VM specifications of multiple node pools are the same but the node pools are deployed in different AZs, the system randomly selects a node pool to trigger scaling.
- **Scale-In Cooling Interval:** Set this parameter in the unit of minute or hour. This parameter indicates the interval between the previous scale-out action and the next scale-in action.

Scale-in cooling intervals can be configured in the node pool settings and the [autoscaler add-on](#) settings.

#### Scale-in cooling interval configured in a node pool

This interval indicates the period during which nodes added to the current node pool after a scale-out operation cannot be deleted. This interval takes effect at the node pool level.

#### Scale-in cooling interval configured in the autoscaler add-on

The interval after a scale-out indicates the period during which the entire cluster cannot be scaled in after the autoscaler add-on triggers scale-out (due to the unschedulable pods, metrics, and scaling policies). This interval takes effect at the cluster level.

The interval after a node is deleted indicates the period during which the cluster cannot be scaled in after the autoscaler add-on triggers scale-in. This interval takes effect at the cluster level.

The interval after a failed scale-in indicates the period during which the cluster cannot be scaled in after the autoscaler add-on triggers scale-in. This interval takes effect at the cluster level.

#### NOTE

You are advised not to store important data on nodes in a node pool because after auto scaling, data cannot be restored as nodes may be deleted.

If **Autoscaler** is enabled, install the [autoscaler add-on](#) to use the auto scaling feature.

- **AZ:** An AZ is a physical region where resources use independent power supply and networks. AZs are physically isolated but interconnected through an internal network.

Set an AZ based on your requirements. After a node pool is created, **AZ** cannot be modified. Exercise caution when selecting an AZ for the node pool.

To enhance workload reliability, you are advised to select **Random AZ**, allowing nodes to be randomly and evenly distributed among different AZs.

- **Specifications:** Select the node specifications based on service requirements. The available node specifications vary depending on AZs.

To ensure node stability, CCE automatically reserves some resources to run necessary system components. For details, see [Formula for Calculating the Reserved Resources of a Node](#).

- **OS:** Select an OS for the node to be created.
  - **Public image:** Select an OS for the node.
  - **Private image (OBT):** If no private image is available, click **Creating a Private Image** to create one. **This function is available only for clusters of v1.15 or later.**

---

#### NOTICE

Reinstalling the OS or modifying OS configurations could make the node unavailable. Exercise caution when performing these operations.

- **VPC:** The value is the same as that of the cluster and cannot be changed.

**This parameter is displayed only for clusters of v1.13.10-r0 and later.**

- **Subnet:** A subnet improves network security by providing exclusive network resources that are isolated from other networks.

You can select any subnet in the cluster VPC. Cluster nodes can belong to different subnets.

Ensure that the DNS server in the subnet can resolve the OBS domain name. Otherwise, nodes cannot be created.

**This parameter is displayed only for clusters of v1.13.10-r0 and later.**

- **System Disk:** Set the system disk space of the worker node. The value ranges from 40GB to 1024 GB. The default value is 40GB.

By default, system disks support High I/O (SAS) and Ultra-high I/O (SSD) EVS disks.

- **Data Disk:** Set the data disk space of the worker node. The value ranges from 100 GB to 32,768 GB. The default value is 100 GB. The EVS disk types provided for the data disk are the same as those for the system disk.

---

 **CAUTION**

If the data disk is uninstalled or damaged, the Docker service becomes abnormal and the node becomes unavailable. You are advised not to delete the data disk.

- 
- **LVM:** If this option is selected, CCE data disks are managed by the Logical Volume Manager (LVM). On this condition, you can adjust the disk space allocation for different resources. This option is selected for the first disk by default and cannot be unselected. You can choose to enable or disable LVM for new data disks.
    - This option is selected by default, indicating that LVM management is enabled.
    - You can deselect the check box to disable LVM management.

---

 **CAUTION**

- Disk space of the data disks managed by LVM will be allocated according to the ratio you set.
  - When creating a node in a cluster of v1.13.10 or later, if LVM is not selected for a data disk, follow instructions in [Adding a Second Data Disk to a Node in a CCE Cluster](#) to fill in the pre-installation script and format the data disk. Otherwise, the data disk will still be managed by LVM.
  - When creating a node in a cluster earlier than v1.13.10, you must format the data disks that are not managed by LVM. Otherwise, either these data disks or the first data disk will be managed by LVM.
- 
- **Add Data Disk:** Currently, a maximum of two data disks can be attached to a node. After the node is created, you can go to the ECS console to

attach more data disks. This function is available only to clusters of certain versions.

- **Data disk space allocation:** Click [Change Configuration](#) to specify the resource ratio for **Kubernetes Space** and **User Space**. Disk space of the data disks managed by LVM will be allocated according to the ratio you set. This function is available only to clusters of certain versions.
  - **Kubernetes Space:** You can specify the ratio of the data disk space for storing Docker and kubelet resources. Docker resources include the Docker working directory, Docker images, and image metadata. kubelet resources include pod configuration files, secrets, and emptyDirs.

The Docker space cannot be less than 10%, and the space size cannot be less than 60 GB. The kubelet space cannot be less than 10%.


The Docker space size is determined by your service requirements. For details, see [Data Disk Space Allocation](#).
  - **User Space:** You can set the ratio of the disk space that is not allocated to Kubernetes resources and the path to which the user space is mounted.

#### NOTE

Note that the mount path cannot be `/`, `/home/paas`, `/var/paas`, `/var/lib`, `/var/script`, `/var/log`, `/mnt/paas`, or `/opt/cloud`, and cannot conflict with the system directories (such as `bin`, `lib`, `home`, `root`, `boot`, `dev`, `etc`, `lost+found`, `mnt`, `proc`, `sbin`, `srv`, `tmp`, `var`, `media`, `opt`, `selinux`, `sys`, and `usr`). Otherwise, the system or node installation will fail.

---

#### NOTICE

- The ratio of disk space allocated to the Kubernetes space and user space must be equal to 100% in total. You can click  to refresh the data after you have modified the ratio.
- By default, disks run in the direct-lvm mode. If data disks are removed, the loop-lvm mode will be used and this will impair system stability.

- 
- **Login Mode:** You can use a password or key pair.
    - **Password:** The default username is `root`. Enter the password for logging in to the node and confirm the password.

Be sure to remember the password as you will need it when you log in to the node.
    - **Key pair:** Select the key pair used to log in to the node. You can select a shared key.

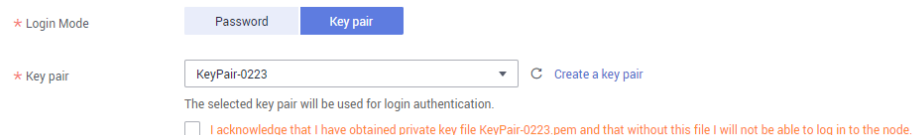
A key pair is used for identity authentication when you remotely log in to a node. If no key pair is available, click **Create a key pair**.




**NOTICE**

When creating a node using a key pair, IAM users can select only the key pairs created by their own, regardless of whether these users are in the same group. For example, user B cannot use the key pair created by user A to create a node, and the key pair is not displayed in the drop-down list on the CCE console.

**Figure 17-50** Key pair



**Step 4 Advanced ECS Settings** (optional): Click  to show advanced ECS settings.


- **ECS Group:** An ECS group logically groups ECSs. The ECSs in the same ECS group comply with the same policy associated with the ECS group.
  - **Anti-affinity:** ECSs in an ECS group are deployed on different physical hosts to improve service reliability.

Select an existing ECS group, or click **Create ECS Group** to create one. After the ECS group is created, click the refresh button.

- **Resource Tags:** By adding tags to resources, you can classify resources. You can create predefined tags in Tag Management Service (TMS). Predefined tags are visible to all service resources that support the tagging function. You can use predefined tags to improve tag creation and migration efficiency. CCE will automatically create the "CCE-Dynamic-Provisioning-Node=node id" tag. A maximum of 5 tags can be added.
- **Agency:** An agency is created by a tenant administrator on the IAM console. By creating an agency, you can share your cloud server resources with another account, or entrust a more professional person or team to manage your resources. To authorize an ECS or BMS to call cloud services, select **Cloud service** as the agency type, click **Select**, and then select **ECS BMS**.
- **Pre-installation Script:** Enter a maximum of 1,000 characters. The script will be executed before Kubernetes software is installed. Note that if the script is incorrect, Kubernetes software may fail to be installed. The script is usually used to format data disks.
- **Post-installation Script:** Enter a maximum of 1,000 characters. The script will be executed after Kubernetes software is installed and will not affect the installation. The script is usually used to modify Docker parameters.
- **Subnet IP Address:** Select **Automatically assign IP address** (recommended) or **Manually assigning IP addresses**.

 **NOTE**

When you **manually assign IPs**, the master IP is randomly specified. Therefore, it may conflict with the worker node IP. If you prefer the manual operation, you are advised to select a subnet CIDR block different from that of the master node when setting worker node **subnet**.

**Step 5 Advanced Kubernetes Settings** (optional): Click  to show advanced Kubernetes settings.

- **Max Pods:** maximum number of pods that can be created on a node, including the system's default pods. If the cluster uses the **VPC network model**, the maximum value is determined by the number of IP addresses that can be allocated to containers on each node.

This limit prevents the node from being overloaded by managing too many pods. For details, see [Maximum Number of Pods That Can Be Created on a Node](#).


- **Taints:** This field is left blank by default. Taints allow nodes to repel a set of pods. You can add a maximum of 10 taints for each node. Each taint contains the following parameters:
  - **Key:** A key must contain 1 to 63 characters starting with a letter or digit. Only letters, digits, hyphens (-), underscores (\_), and periods (.) are allowed. A DNS subdomain name can be used as the prefix of a key.
  - **Value:** A value must start with a letter or digit and can contain a maximum of 63 characters, including letters, digits, hyphens (-), underscores (\_), and periods (.).
  - **Effect:** Available options are **NoSchedule**, **PreferNoSchedule**, and **NoExecute**.


---

**NOTICE**

- If taints are used, you must configure tolerations in the YAML files of pods. Otherwise, scale-up may fail or pods cannot be scheduled onto the added nodes.
- After a node pool is created, you can click **Edit** to modify its configuration. The modification will be synchronized to all nodes in the node pool.

- **K8S Labels:** Labels are key/value pairs that are attached to objects, such as pods. Labels are used to specify identifying attributes of objects that are meaningful and relevant to users, but do not directly imply semantics to the core system. For more information, see [Labels and Selectors](#).
- **Maximum Data Space per Container:** maximum data space that can be used by a container. The value ranges from 10 GB to 500 GB. If the value of this field is larger than the data disk space allocated to Docker resources, the latter will override the value specified here. Typically, 90% of the data disk space is allocated to Docker resources. This parameter is displayed only for clusters of v1.13.10-r0 and later.

 [Add Node Pool](#)

**Step 6** (Optional) Click  [Add Node Pool](#) on the left to add multiple node pools. You can view the available node pool quotas under the button.

**Step 7** Click **Next: Confirm** to confirm the configured service parameters, fees, and specifications.

**Step 8** Click **Submit**.

It takes about 6 to 10 minutes to create a node pool. You can click **Back to Node Pool List** to perform other operations on the node pool or click **Go to Node Pool**

**Events** to view the node pool details. If the status of the node pool is Normal, the node pool is successfully created.

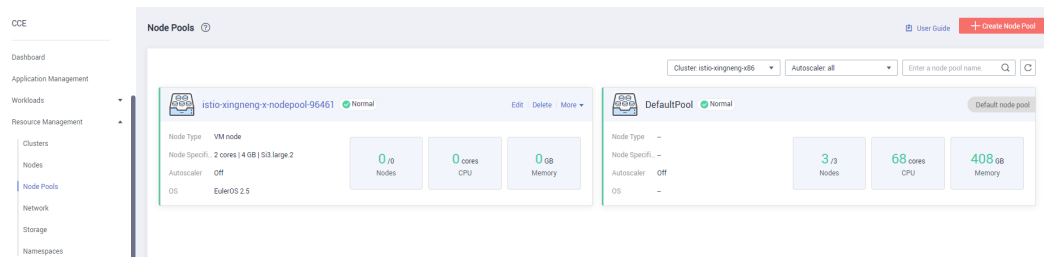
----End

## Viewing Node Pools in a Cluster

**Step 1** Log in to the CCE console. In the navigation pane, choose **Resource Management** > **Node Pools**.

**Step 2** In the upper right corner of the node pool list, select a cluster. All node pools in the cluster will be displayed. You can view the node type, node specifications, autoscaler status, and OS of each node pool.

**Figure 17-51** Viewing node pools in a cluster



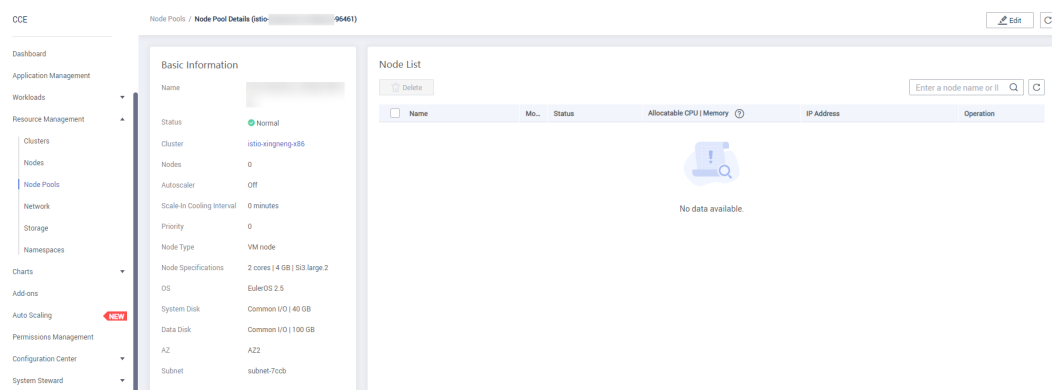
### NOTE

- A default node pool **DefaultPool** is automatically created in each cluster. The default node pool cannot be edited, deleted, or migrated. All nodes created during and after cluster creation are displayed in the default node pool.
- To display a list of nodes in **DefaultPool**, click the **Nodes** subcard in the **DefaultPool** card.

**Step 3** To filter node pools by autoscaler status, select the autoscaler status in the upper right corner of the node pool list.

**Step 4** In the node pool list, click a node pool name. On the node pool details page, view the basic information, advanced ECS settings, advanced Kubernetes settings, and node list of the node pool.

**Figure 17-52** Node pool details



----End

## 17.5.3 Managing a Node Pool

### Notes and Constraints

The default node pool DefaultPool does not support the following management operations.

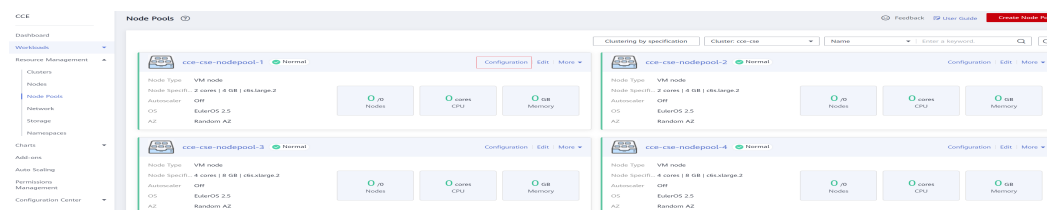
### Configuring Kubernetes Parameters

CCE allows you to highly customize Kubernetes parameter settings on core components in a cluster. For more information, see [kubelet](#).

This function is supported only in clusters of **v1.15 and later**. It is not displayed for clusters earlier than v1.15.

- Step 1** Log in to the CCE console. In the navigation pane, choose **Resource Management > Node Pools**.
- Step 2** In the upper right corner of the displayed page, select a cluster to filter node pools by cluster.
- Step 3** Click **Configuration** next to the node pool name.

Figure 17-53 Node pool configuration



- Step 4** On the **Configuration** page on the right, change the values of the following Kubernetes parameters:

Table 17-28 Kubernetes parameters

Compon ent	Parameter	Description	Default Value	Remarks
docker	native-umask	`--exec-opt native.umask	normal	Cannot be changed.
	docker-base-size	`--storage-opts dm.basesize	10G	Cannot be changed.
	insecure-registry	Address of an insecure image registry	false	Cannot be changed.
	limitcore	Limit on the number of cores	5368709120	-

Component	Parameter	Description	Default Value	Remarks
	default-ulimit-nofile	Limit on the number of handles in a container	{soft}: {hard}	-
kube-proxy	contrack-min	sysctl -w net.nf_contrack_max	131072	The values can be modified during the node pool lifecycle.
	contrack-tcp-timeout-close-wait	sysctl -w net.netfilter.nf_contrack_tcp_timeout_close_wait	1h0m0s	
kubelet	cpu-manager-policy	--cpu-manager-policy	none	The values can be modified during the node pool lifecycle.
	kube-api-qps	Query per second (QPS) to use while talking with kube-apiserver.	100	
	kube-api-burst	Burst to use while talking with kube-apiserver.	100	
	max-pods	Maximum number of pods managed by kubelet.	110	
	pod-pids-limit	PID limit in Kubernetes	-1	
	with-local-dns	Whether to use the local IP address as the ClusterDNS of the node.	false	

Component	Parameter	Description	Default Value	Remarks
	allowed-unsafe-sysctls	Insecure system configuration allowed. Starting from <b>v1.17.17</b> , CCE enables pod security policies for kube-apiserver. You need to add corresponding configurations to <b>allowedUnsafeSysctls</b> of a pod security policy to make the policy take effect. (This configuration is not required for clusters earlier than v1.17.17.) For details, see <a href="#">Example of Enabling Unsafe Sysctls in Pod Security Policy</a> .	[]	

**Step 5** Click **OK**.

----End

## Editing a Node Pool


**Step 1** Log in to the CCE console. In the navigation pane, choose **Resource Management > Node Pools**.

**Step 2** In the upper right corner of the displayed page, select a cluster to filter node pools by cluster.

**Step 3** Click **Edit** next to the name of the node pool you will edit. In the **Edit Node Pool** dialog box, edit the following parameters:

**Table 17-29** Node pool parameters

Parameter	Description
Name	Name of the node pool.
Nodes	Modify the number of nodes based on service requirements.

Parameter	Description
Autoscaler	<p>By default, autoscaler is disabled.</p> <p>After you enable autoscaler by clicking , nodes in the node pool are automatically created or deleted based on service requirements.</p> <ul style="list-style-type: none"> <li>• <b>Maximum Nodes</b> and <b>Minimum Nodes</b>: You can set the maximum and minimum number of nodes to ensure that the number of nodes to be scaled is within a proper range.</li> <li>• <b>Priority</b>: A larger value indicates a higher priority. For example, if this parameter is set to <b>1</b> and <b>4</b> respectively for node pools A and B, B has a higher priority than A, and auto scaling is first triggered for B. If the priorities of multiple node pools are set to the same value, for example, <b>2</b>, the node pools are not prioritized and the system performs scaling based on the minimum resource waste principle.</li> <li>• <b>Scaling-In Cooling Interval</b>: Set this parameter in the unit of minute. This field indicates the period during which the nodes added in the current node pool cannot be scaled in.</li> </ul> <p>If the <b>Autoscaler</b> field is set to on, install the <a href="#">autoscaler add-on</a> to use the autoscaler feature.</p>
Taints	<ul style="list-style-type: none"> <li>• This field is left blank by default. Taints allow nodes to repel a set of pods. You can add a maximum of 10 taints for each node pool. Each taint contains the following parameters: <ul style="list-style-type: none"> <li>– <b>Key</b>: A key must contain 1 to 63 characters starting with a letter or digit. Only letters, digits, hyphens (-), underscores (_), and periods (.) are allowed. A DNS subdomain name can be used as the prefix of a key.</li> <li>– <b>Value</b>: A value must start with a letter or digit and can contain a maximum of 63 characters, including letters, digits, hyphens (-), underscores (_), and periods (.)</li> <li>– <b>Effect</b>: Available options are <b>NoSchedule</b>, <b>PreferNoSchedule</b>, and <b>NoExecute</b>.</li> </ul> </li> </ul> <p><b>NOTICE</b> If taints are used, you must configure tolerations in the YAML files of pods. Otherwise, scale-up may fail or pods cannot be scheduled onto the added nodes.</p>

Parameter	Description
K8S Labels	K8S labels are key/value pairs that are attached to objects, such as pods. Labels are used to specify identifying attributes of objects that are meaningful and relevant to users, but do not directly imply semantics to the core system. For more information, see <a href="#">Labels and Selectors</a> .
Resource Tags	It is recommended that you use TMS's <b>predefined tag</b> function to add the same tag to different cloud resources.  Predefined tags are visible to all service resources that support the tagging function. You can use predefined tags to improve tag creation and migration efficiency.  Tag changes do not affect the node.

**Step 4** After the configuration is complete, click **Save**.

In the node pool list, the node pool status becomes **Scaling**. After the status changes to **Completed**, the node pool parameters are modified successfully. The modified configuration will be synchronized to all nodes in the node pool.

----End

## Deleting a Node Pool

Deleting a node pool will delete nodes in the pool. Pods on these nodes will be automatically migrated to available nodes in other node pools. If pods in the node pool have a specific node selector and none of the other nodes in the cluster satisfies the node selector, the pods will become unschedulable.

**Step 1** Log in to the CCE console. In the navigation pane, choose **Resource Management > Node Pools**.

**Step 2** In the upper right corner of the displayed page, select a cluster to filter node pools by cluster.

**Step 3** Choose **More > Delete** next to a node pool name to delete the node pool.

**Step 4** Read the precautions in the **Delete Node Pool** dialog box.

**Step 5** Enter **DELETE** in the text box and click **Yes** to confirm that you want to continue the deletion.

----End

## Copying a Node Pool

You can copy the configuration of an existing node pool to create a new node pool on the CCE console.

**Step 1** Log in to the CCE console. In the navigation pane, choose **Resource Management > Node Pools**.



- Step 2** In the upper right corner of the displayed page, select a cluster to filter node pools by cluster.
  - Step 3** Choose **More > Copy** next to a node pool name to copy the node pool.
  - Step 4** The configuration of the selected node pool is replicated to the **Create Node Pool** page. You can edit the configuration as required and click **Next: Confirm**.
  - Step 5** On the **Confirm** page, confirm the node pool configuration and click **Create Now**. Then, a new node pool is created based on the edited configuration.
- End

## Migrating a Node

Nodes in a node pool can be migrated. Currently, nodes in a node pool can be migrated only to the default node pool (defaultpool) in the same cluster.

- Step 1** Log in to the CCE console. In the navigation pane, choose **Resource Management > Node Pools**.
- Step 2** In the upper right corner of the displayed page, select a cluster to filter node pools by cluster.
- Step 3** Click **More > Migrate** next to the name of the node pool.
- Step 4** In the dialog box displayed, select the destination node pool and the node to be migrated.

### NOTE

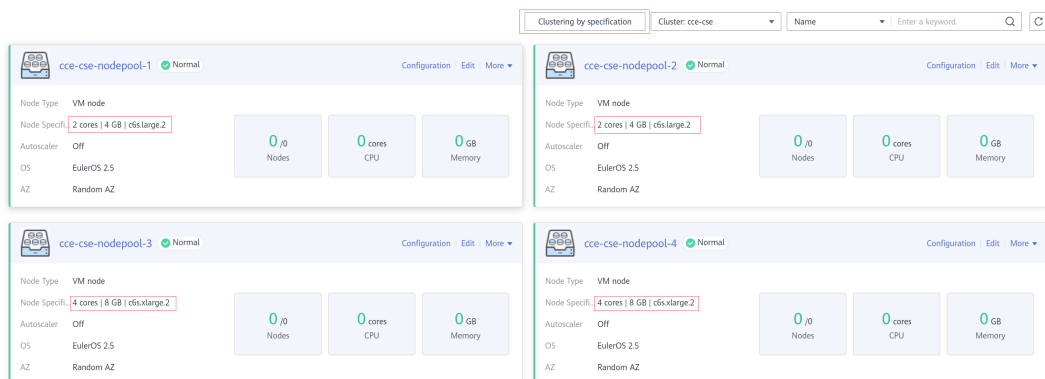
After node migration, original resource tags, Kubernetes labels, and taints will be retained, and new Kubernetes labels and taints from the destination node pool will be added.

- Step 5** Click **OK**.
- End

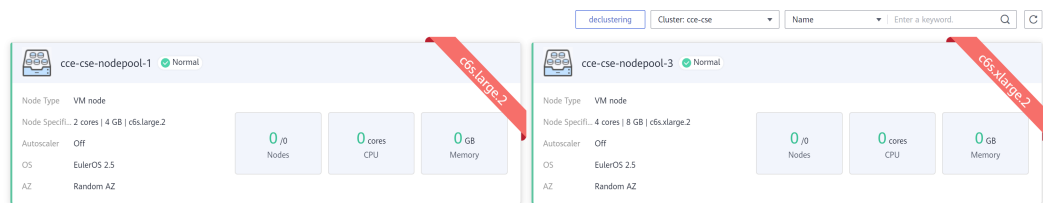
## Clustering by Specifications

When a large number of node pools are created, searching for node pools becomes difficult. CCE provides the clustering function to aggregate node pools of the same specifications to facilitate searching.

As shown in the following figure, there are four node pools. The specifications of the upper two node pools are the same, and the specifications of the lower two node pools are the same.



Click **Cluster by specifications** to aggregate node pools of the same specifications, as shown in the following figure. After you click a node pool, the node pools with the same specifications are displayed on the right.



## 17.6 Workloads

### 17.6.1 Overview

A workload is an application running on Kubernetes. No matter how many components are there in your workload, you can run it in a group of Kubernetes pods. A workload is an abstract model of a group of pods in Kubernetes. Workloads classified in Kubernetes include Deployments, StatefulSets, DaemonSets, jobs, and cron jobs.

CCE provides Kubernetes-native container deployment and management and supports lifecycle management of container workloads, including creation, configuration, monitoring, auto scaling, upgrade, uninstall, service discovery, and load balancing.

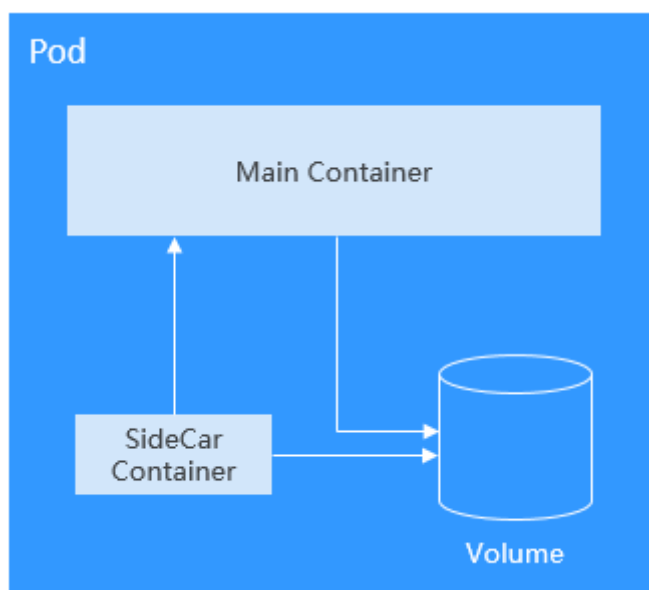
### Overview of Pod

A pod is the smallest and simplest unit in the Kubernetes object model that you create or deploy. A pod is a group of one or more containers, with shared storage and network resources, and a specification for how to run the containers. Each pod has a separate IP address.

Pods can be used in either of the following ways:

- A pod runs only one container. This is the most common usage of pods in Kubernetes. You can consider a pod as a container, but Kubernetes directly manages pods instead of containers.
- A pod runs multiple containers that need to be tightly coupled. In this scenario, a pod contains a main container and several sidecar containers, as shown in [Figure 17-54](#). For example, the main container is a web server that provides file services from a fixed directory, and sidecar containers periodically download files to this fixed directory.

**Figure 17-54** Pod running multiple containers

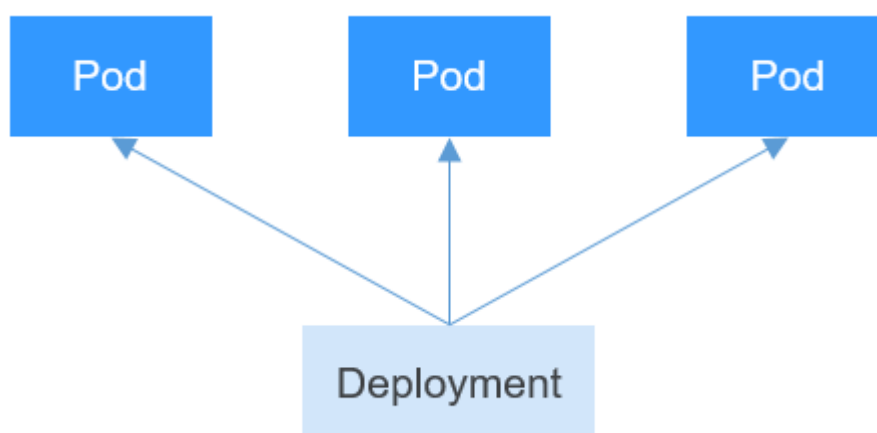


In Kubernetes, pods are rarely created directly. Instead, Kubernetes controller manages pods through pod instances such as Deployments and jobs. A controller typically uses a pod template to create pods. The controller can also manage multiple pods and provide functions such as replica management, rolling upgrade, and self-healing.

## Overview of Deployment

A pod is the smallest and simplest unit that you create or deploy in Kubernetes. It is designed to be an ephemeral, one-off entity. A pod can be evicted when node resources are insufficient and disappears along with a cluster node failure. Kubernetes provides controllers to manage pods. Controllers can create and manage pods, and provide replica management, rolling upgrade, and self-healing capabilities. The most commonly used controller is Deployment.

**Figure 17-55** Relationship between a Deployment and pods



A Deployment can contain one or more pods. These pods have the same role. Therefore, the system automatically distributes requests to multiple pods of a Deployment.

A Deployment integrates a lot of functions, including online deployment, rolling upgrade, replica creation, and restoration of online jobs. To some extent, Deployments can be used to realize unattended rollout, which greatly reduces difficulties and operation risks in the rollout process.

## Overview of StatefulSet

All pods under a Deployment have the same characteristics except for the name and IP address. If required, a Deployment can use a pod template to create new pods. If not required, the Deployment can delete any one of the pods.

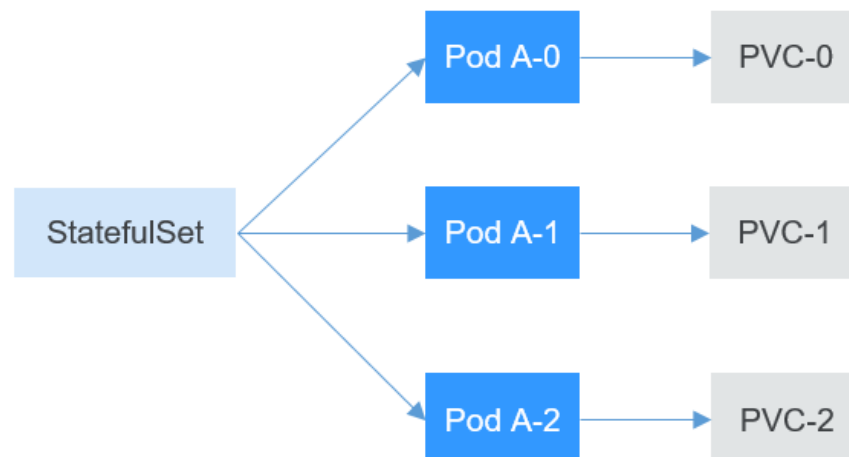
However, Deployments cannot meet the requirements in some distributed scenarios when each pod requires its own status or in a distributed database where each pod requires independent storage.

Distributed stateful applications involve different roles for different responsibilities. For example, databases work in active/standby mode, and pods depend on each other. To deploy stateful applications in Kubernetes, ensure pods meet the following requirements:

- Each pod must have a fixed identifier so that it can be recognized by other pods.
- Separate storage resources must be configured for each pod. In this way, the original data can be retrieved after a pod is deleted and restored. Otherwise, the pod status will be changed after the pod is rebuilt.

To address the preceding requirements, Kubernetes provides StatefulSets.

1. StatefulSets provide a fixed name for each pod following a fixed number ranging from 0 to N. After a pod is rescheduled, the pod name and the hostname remain unchanged.
2. StatefulSets use a headless Service to allocate a fixed domain name for each pod.
3. StatefulSets create PersistentVolumeClaims (PVCs) with fixed identifiers to ensure that pods can access the same persistent data after being rescheduled.

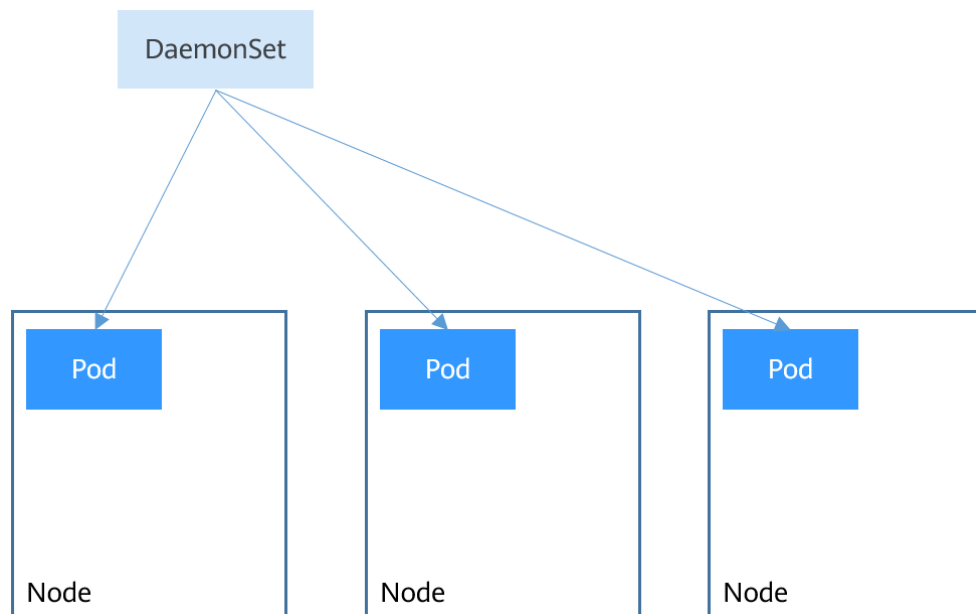


## Overview of DaemonSet

A DaemonSet runs a pod on each node in a cluster and ensures that there is only one pod. This works well for certain system-level applications such as log collection and resource monitoring since they must run on each node and need only a few pods. A good example is kube-proxy.

DaemonSets are closely related to nodes. If a node becomes faulty, the DaemonSet will not create the same pods on other nodes.

**Figure 17-56** DaemonSet



## Overview of Job and CronJob

Jobs and CronJobs allow you to run short lived, one-off tasks in batch. They ensure the task pods run to completion.

- A job is a resource object used by Kubernetes to control batch tasks. Jobs are different from long-term servo tasks (such as Deployments and StatefulSets). The former is started and terminated at specific times, while the latter runs unceasingly unless being terminated. The pods managed by a job will be automatically removed after successfully completing tasks based on user configurations.
- A CronJob runs a job periodically on a specified schedule. A CronJob object is similar to a line of a crontab file in Linux.

This run-to-completion feature of jobs is especially suitable for one-off tasks, such as continuous integration (CI).

## Workload Lifecycle

**Table 17-30** Status description

Status	Description
Running	All pods are running.
Unready	A container is abnormal, the number of pods is 0, or the workload is in pending state.
Upgrading/Rolling back	The workload is being upgraded or rolled back.
Available	For a multi-pod Deployment, some pods are abnormal but at least one pod is available.
Completed	The task is successfully executed. This status is available only for common tasks.
Stopped	The workload is stopped and the number of pods changes to 0. This status is available for workloads earlier than v1.13.
Deleting	The workload is being deleted.
Pausing	The workload is being paused.

## 17.6.2 Creating a Deployment

### Scenario

Deployments are workloads (for example, Nginx) that do not store any data or status. You can create Deployments on the CCE console or by running `kubectl` commands.

### Prerequisites

- Before creating a containerized workload, you must have an available cluster. For details on how to create a cluster, see [Buying a CCE Cluster](#).
- To enable public access to a workload, ensure that an EIP or load balancer has been bound to at least one node in the cluster.

#### NOTE

If a pod has multiple containers, ensure that the ports used by the containers do not conflict with each other. Otherwise, creating the Deployment will fail.

### Using the CCE Console

CCE provides multiple methods for creating a workload. You can use any of the following methods:

- Use an image in Open Source Images. You do not need to upload any image before using it.

- Use an image that you have uploaded to SWR.
- Use a **shared image** to create a workload. Specifically, other tenants share an image with you by using the **SWR service**.
- Use a YAML file to create a workload. You can click **Create YAML** on the right of the **Configure Advanced Settings** page when creating a Deployment. For details about YAML, see [Table 17-33](#). After the YAML file is written, click **Create** to create a workload.

 **NOTE**

Settings in the YAML file are synchronized with those on the console. You can edit the YAML file on the console to create a workload. For example:

- If you enter a workload name on the console, the name will automatically appear in the YAML file.
- If you add an image on the console, the image will be automatically added to the YAML file.

When you click **Create YAML** on the right of the console, do not create multiple YAML files in the YAML definition pane displayed. You need to create them one by one. Otherwise, an error will be reported during the creation.

**Step 1** Log in to the CCE console. In the navigation pane, choose **Workloads > Deployments**. On the page displayed, click **Create Deployment**. Set basic workload parameters as described in [Table 17-31](#). The parameters marked with an asterisk (\*) are mandatory.

**Table 17-31** Basic workload parameters

Parameter	Description
* Workload Name	Name of the workload to be created. The name must be unique. Enter 4 to 63 characters starting with a letter and ending with a letter or digit. Only lowercase letters, digits, and hyphens (-) are allowed.
* Cluster Name	Cluster to which the workload belongs.
* Namespace	In a single cluster, data in different namespaces is isolated from each other. This enables applications to share the services of the same cluster without interfering each other. If no namespace is set, the default namespace is used.
* Instances	Number of pods in the workload. A workload can have one or more pods. You can set the number of pods. The default value is <b>2</b> and can be set to <b>1</b> . Each workload pod consists of the same containers. Configuring multiple pods for a workload ensures that the workload can still run properly even if a pod is faulty. If only one pod is used, a node or pod exception may cause service exceptions.

Parameter	Description
Time Zone Synchronization	<p>If this parameter is enabled, the container and the node use the same time zone.</p> <p><b>NOTICE</b> After time zone synchronization is enabled, disks of the hostPath type will be automatically added and listed in the <b>Data Storage &gt; Local Volume</b> area. Do not modify or delete the disks.</p>
Description	Description of the workload.

**Step 2** Click **Next: Add Container**.

- Click **Add Container** and select the image to be deployed.
  - My Images:** Create a workload using an image in the image repository you created.
  - Third-Party Images:** Create a workload using an image from any third-party image repository. When you create a workload using a third-party image, ensure that the node where the workload is running can access public networks. For details on how to create a workload using a third-party image, see [Using a Third-Party Image](#).
    - If your image repository does not require authentication, set **Secret Authentication** to **No**, enter an image pull address, and then click **OK**.
    - If your image repository must be authenticated (account and password), you need to create a secret and then use a third-party image. For details, see [Using a Third-Party Image](#).
  - Shared Images:** Create a workload using an image shared by another tenant through the SWR service.
- Configure basic image information.

A workload is an abstract model of a group of pods. One pod can encapsulate one or more containers. You can click **Add Container** in the upper right corner to add multiple container images and set them separately.

**Table 17-32** Image parameters

Parameter	Description
Image Name	Name of the image. You can click <b>Change Image</b> to update it.
*Image Version	Select the image tag to be deployed.
*Container Name	Name of the container. You can modify it.



Parameter	Description
Privileged Container	<p>Programs in a privileged container have certain privileges. If <b>Privileged Container</b> is <b>On</b>, the container is granted superuser permissions. For example, privileged containers can manipulate network devices on the host machine and modify kernel parameters.</p>
Container Resources	<p><b>CPU</b></p> <ul style="list-style-type: none"> <li>– <b>Request</b>: minimum number of CPU cores required by a container. The default value is 0.25 cores.</li> <li>– <b>Limit</b>: maximum number of CPU cores available for a container. Do not leave <b>Limit</b> unspecified. Otherwise, intensive use of container resources will occur and your workload may exhibit unexpected behavior.</li> </ul> <p><b>Memory</b></p> <ul style="list-style-type: none"> <li>– <b>Request</b>: minimum amount of memory required by a container. The default value is 512 MiB.</li> <li>– <b>Limit</b>: maximum amount of memory available for a container. When memory usage exceeds the specified memory limit, the container will be terminated.</li> </ul> <p>For more information about <b>Request</b> and <b>Limit</b>, see <a href="#">Setting Container Specifications</a>.</p> <p><b>GPU</b>: configurable only when the cluster contains GPU nodes.</p> <p>It indicates the percentage of GPU resources reserved for a container. Select <b>Use</b> and set the percentage. For example, if this parameter is set to 10%, the container is allowed to use 10% of GPU resources. If you do not select <b>Use</b> or set this parameter to <b>0</b>, no GPU resources can be used.</p> <p><b>GPU/Graphics Card</b>: The workload's pods will be scheduled to the node with the specified GPU.</p> <p>If <b>Any GPU type</b> is selected, the container uses a random GPU in the node. If you select a specific GPU, the container uses this GPU accordingly.</p>

3. **Lifecycle**: Commands for starting and running containers can be set.
  - **Start Command**: executed when the workload is started. For details, see [Setting Container Startup Commands](#).
  - **Post-Start**: executed after the workload runs successfully. For more information, see [Setting Container Lifecycle Parameters](#).
  - **Pre-Stop**: executed to delete logs or temporary files before the workload ends. For more information, see [Setting Container Lifecycle Parameters](#).
4. **Health Check**: CCE provides two types of probes: liveness probe and readiness probe. They are used to determine whether containers and user services are running properly. For more information, see [Setting Health Check for a Container](#).

- **Liveness Probe:** used to restart the unhealthy container.
  - **Readiness Probe:** used to change the container to the unready state when detecting that the container is unhealthy. In this way, service traffic will not be directed to the container.
5. **Environment Variables:** Environment variables can be added to a container. In general, environment variables are used to set parameters.

On the **Environment Variables** tab page, click **Add Environment Variable**. Currently, three types of environment variables are supported:

- **Added manually:** Set **Variable Name** and **Variable Value/Reference**.
- **Added from Secret:** Set **Variable Name** and select the desired secret name and data. A secret must be created in advance. For details, see [Creating a Secret](#).
- **Added from ConfigMap:** Set **Variable Name** and select the desired ConfigMap name and data. A ConfigMap must be created in advance. For details, see [Creating a ConfigMap](#).

 **NOTE**

To edit an environment variable that has been set, click **Edit**. To delete an environment variable that has been set, click **Delete**.

6. **Data Storage:** Data storage can be mounted to containers for persistent storage and high disk I/O. Local volume and cloud storage are supported. For details, see [Storage \(CSI\)](#).
7. **Security Context:** Container permissions can be configured to protect CCE and other containers from being affected.
- Enter the user ID to set container permissions and prevent systems and other containers from being affected.
8. **Log Policies:** Log collection policies and log directory can be configured to collect container logs for unified management and analysis. For details, see [Container Logs](#).

**Step 3** Click **Next: Set Application Access**. Then, click **Add Service** and set the workload access type.

If your workload will be reachable to other workloads or public networks, add a Service to define the workload access type.

The workload access type determines the network attributes of the workload. Workloads with different access types can provide different network capabilities. For details, see [Overview](#).

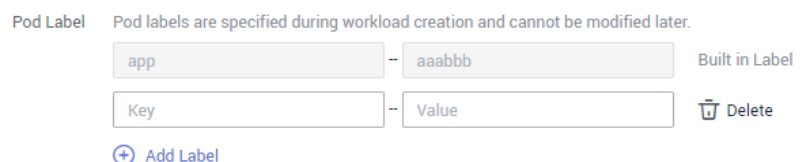
**Step 4** Click **Next: Configure Advanced Settings** to configure advanced policies.

- **Upgrade Mode:** You can specify the upgrade mode of a Deployment, including **Rolling upgrade** and **In-place upgrade**.
  - **Rolling upgrade:** Old pods are gradually replaced with new ones. During the upgrade, service traffic is evenly distributed to both pods to ensure service continuity.
    - **Maximum Number of Unavailable 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. Minimum number of alive pods = Total pods – Maximum number of unavailable pods

- **In-place upgrade:** Old pods are deleted before new pods are created. Services will be interrupted during an in-place upgrade.
- **Graceful Deletion:** A time window can be set for workload deletion and reserved for executing commands in the pre-stop phase in the lifecycle. If workload processes are not terminated after the time window elapses, the workload will be forcibly deleted.
  - **Graceful Time Window (s):** Set a time window (0–9999s) for pre-stop commands to finish execution before a workload is deleted. The default value is 30s.
  - **Scale Order:** Choose **Prioritize new pods** or **Prioritize old pods** based on service requirements. **Prioritize new pods** indicates that new pods will be first deleted when a scale-in is triggered.
- **Migration Policy:** When the node where a workload's pods are located is unavailable for the specified amount of time, the pods will be rescheduled to other available nodes.
  - **Migration Time Window (s):** Set a time window for migration. The default value is 300s.
- **Scheduling Policies:** You can combine static global scheduling policies or dynamic runtime scheduling policies as required. For details, see [Scheduling Policy Overview](#).
- **Advanced Pod Settings**
  - **Pod Label:** The built-in **app** label is specified when the workload is created. It is used to set affinity and anti-affinity scheduling and cannot be modified. You can click **Add Label** to add labels.

**Figure 17-57** Advanced pod settings

Advanced Pod Settings




- **Client DNS Configuration:** A CCE cluster has a built-in DNS add-on (CoreDNS) to provide domain name resolution for workloads in the cluster.
  - **DNS Policy**
    - **ClusterFirst:** The default DNS configuration overrides the **Nameserver** and **DNS Search Domain** configurations of the client.
    - **None:** Only the **Nameserver** and **DNS Search Domain** configurations are used for domain name resolution.
    - **Default:** The pod inherits the DNS configuration from the node on which the pod runs.
  - **Nameserver:** You can configure a domain name server for a user-defined domain name. The value is one or a group of DNS IP addresses, for example, 1.2.3.4.

- **DNS Search Domain:** a search list for host-name lookup. When a domain name cannot be resolved, DNS queries will be attempted combining the domain name with each domain in the search list in turn until a match is found or all domains in the search list are tried.
- **Timeout (s):** amount of time the resolver will wait for a response from a remote name server before retrying the query on a different name server. Set it based on the site requirements.
- **ndots:** threshold for the number of dots that must appear in a domain name before an initial absolute query will be made. If a domain name has **ndots** or more than **ndots** dots, the name is a fully qualified domain name (FQDN) and will be tried first as an absolute name. If a domain name has less than **ndots** dots, the operating system will look up the name in a list of search domain names.

**Step 5** After the preceding configurations are complete, click **Create**. On the page displayed, click **Return to Workload List** to view the workload status.

If the workload is in the **Running** state, it has been successfully created.

Workload status is not updated in real time. Click  in the upper right corner or press **F5** to refresh the page.

**Step 6** To access the workload in a browser, go to the workload list on the **Deployments** page. Copy the corresponding **External Access Address** and paste it into the address box in the browser.

 **NOTE**

- External access addresses are available only if the Deployment access type is set to **NodePort** and an EIP is assigned to any node in the cluster, or if the Deployment access type is set to **LoadBalancer (ELB)**.
- If the workload list contains more than 500 records, the Kubernetes pagination mechanism will be used. Specifically, you can only go to the first page or the next page, but cannot go to the previous page. In addition, if resources are divided into discrete pages, the total number of resources displayed is the maximum number of resources that can be queried at a time, not the actual total number of resources.

----End

## Using kubectl

The following procedure uses Nginx as an example to describe how to create a workload using kubectl.

**Step 1** Use kubectl to connect to the cluster. For details, see [Connecting to a Cluster Using kubectl](#).

**Step 2** Create and edit the **nginx-deployment.yaml** file. **nginx-deployment.yaml** is an example file name. You can rename it as required.

**vi nginx-deployment.yaml**

The following is an example YAML file. For more information about Deployments, see [Kubernetes documentation](#).

```
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 you use an image from an open-source image registry, 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

```

For details about these parameters, see [Table 17-33](#).

**Table 17-33** Deployment YAML parameters

Parameter	Description	Mandatory/Optional
apiVersion	API version. <b>NOTE</b> Set this parameter based on the cluster version. <ul style="list-style-type: none"> <li>For clusters of v1.17 or later, the apiVersion format of Deployments is <b>apps/v1</b>.</li> <li>For clusters of v1.15 or earlier, the apiVersion format of Deployments is <b>extensions/v1beta1</b>.</li> </ul>	Mandatory
kind	Type of a created object.	Mandatory
metadata	Metadata of a resource object.	Mandatory
name	Name of the Deployment.	Mandatory
Spec	Detailed description of the Deployment.	Mandatory
replicas	Number of pods.	Mandatory
selector	Determines container pods that can be managed by the Deployment.	Mandatory
strategy	Upgrade mode. Possible values: <ul style="list-style-type: none"> <li>RollingUpdate</li> <li>ReplaceUpdate</li> </ul> By default, rolling update is used.	Optional

Parameter	Description	Mandatory/Optional
template	Detailed description of a created container pod.	Mandatory
metadata	Metadata.	Mandatory
labels	<b>metadata.labels</b> : Container labels.	Optional
spec: containers	<ul style="list-style-type: none"> <li>• <b>image</b> (mandatory): Name of a container image.</li> <li>• <b>imagePullPolicy</b> (optional): Policy for obtaining an image. The options include <b>Always</b> (attempting to download images each time), <b>Never</b> (only using local images), and <b>IfNotPresent</b> (using local images if they are available; downloading images if local images are unavailable). The default value is <b>Always</b>.</li> <li>• <b>name</b> (mandatory): Container name.</li> </ul>	Mandatory
imagePullSecrets	Name of the secret used during image pulling. If a private image is used, this parameter is mandatory. <ul style="list-style-type: none"> <li>• To pull an image from the Software Repository for Container (SWR), set this parameter to <b>default-secret</b>.</li> <li>• To pull an image from a third-party image repository, set this parameter to the name of the created secret.</li> </ul>	Optional

**Step 3** Create a Deployment.

**kubectl create -f nginx-deployment.yaml**

If the following information is displayed, the Deployment is being created.

```
deployment "nginx" created
```

**Step 4** Query the Deployment status.

**kubectl get deployment**

If the following information is displayed, the Deployment is running.

```
NAME      READY   UP-TO-DATE   AVAILABLE   AGE
nginx     1/1     1             1           4m5s
```

**Parameter description**

- **NAME:** pod name
- **READY:** number of pod replicas that have been deployed
- **STATUS:** status of the Deployment
- **RESTARTS:** restart times
- **AGE:** period the Deployment keeps running

**Step 5** If the Deployment will be accessed through a ClusterIP or NodePort Service, add the corresponding Service. For details, see [Networking](#).

----End

## 17.6.3 Creating a StatefulSet

### Scenario

StatefulSets are a type of workloads whose data or status is stored while they are running. For example, MySQL is a StatefulSet because it needs to store new data.

A container can be migrated between different hosts, but data is not stored on the hosts. To store StatefulSet data persistently, attach HA storage volumes provided by CCE to the container.

### Prerequisites

- Before creating a workload, you must have an available cluster. For details on how to create a cluster, see [Buying a CCE Cluster](#).
- To enable public access to a workload, ensure that an EIP or load balancer has been bound to at least one node in the cluster.

#### NOTE

If a pod has multiple containers, ensure that the ports used by the containers do not conflict with each other. Otherwise, creating the StatefulSet will fail.

### Using the CCE Console

CCE provides multiple methods for creating a workload. You can use any of the following methods:

1. Use an image in Open Source Images. You do not need to upload any image before using it.
2. Use an image that you have uploaded to SWR.
3. Use a shared image to create a workload. Specifically, other tenants share an image with you by using the SWR service.
4. Use a YAML file to create a workload. You can click **Create YAML** on the right of the **Create StatefulSet** page. For details about YAML, see [Using kubectl](#). After the YAML file is written, click **Create** to create a workload.

 **NOTE**

Settings in the YAML file are synchronized with those on the console. You can edit the YAML file on the console to create a workload. For example:

- If you enter a workload name on the console, the name will automatically appear in the YAML file.
- If you add an image on the console, the image will be automatically added to the YAML file.

When you click **Create YAML** on the right of the console, do not create multiple YAML files in the YAML definition pane displayed. You need to create them one by one. Otherwise, an error will be reported during the creation.

**Step 1** Log in to the CCE console. In the navigation pane, choose **Workloads > StatefulSets**. On the displayed page, click **Create StatefulSet**. Set basic workload parameters as described in [Table 17-34](#). The parameters marked with an asterisk (\*) are mandatory.

**Table 17-34** Basic workload parameters

Parameter	Description
* Workload Name	Name of a workload, which must be unique. Enter 4 to 52 characters starting with a lowercase letter and ending with a letter or digit. Only lowercase letters, digits, and hyphens (-) are allowed.
* Cluster Name	Cluster to which the workload belongs.
* Namespace	In a single cluster, data in different namespaces is isolated from each other. This enables applications to share the services of the same cluster without interfering each other. If no namespace is set, the <b>default</b> namespace is used.
* Instances	Number of pods in a workload. A workload can have one or more pods. The default value is <b>2</b> . You can customize the value, for example, setting it to <b>1</b> .  Each workload pod consists of the same containers. You can configure multiple pods for a workload to ensure high reliability. For such a workload, if one pod is faulty, the workload can still run properly. If only one pod is used, a node or pod exception may cause service exceptions.
Time Zone Synchronization	If this parameter is enabled, the container and the node use the same time zone.  <b>NOTICE</b> After time zone synchronization is enabled, disks of the hostPath type will be automatically added and listed in the <b>Data Storage &gt; Local Volume</b> area. Do not modify or delete the disks.
Description	Description of the workload.

**Step 2** Click **Next: Add Container**.

1. Click **Add Container** and select the image to be deployed.



- **My Images:** Create a workload using an image in the image repository you created.
  - **Third-Party Images:** Create a workload using an image from any third-party image repository. When you create a workload using a third-party image, ensure that the node where the workload is running can access public networks. For details on how to create a workload using a third-party image, see [Using a Third-Party Image](#).
    - If your image repository does not require authentication, set **Secret Authentication** to **No**, enter an image pull address, and then click **OK**.
    - If your image repository must be authenticated (account and password), you need to create a secret and then use a third-party image. For details, see [Using a Third-Party Image](#).
  - **Shared Images:** Create a workload using an image shared by another tenant through the SWR service.
2. Configure basic image information.

A workload is an abstract model of a group of pods. One pod can encapsulate one or more containers. You can click **Add Container** in the upper right corner to add multiple container images and set them separately.

**Table 17-35** Image parameters

Parameter	Description
Image Name	Name of the image. You can click <b>Change Image</b> to update it.
*Image Version	Select the image tag to be deployed.
*Container Name	Name of the container. You can modify it.
Privileged Container	Programs in a privileged container have certain privileges. If <b>Privileged Container</b> is <b>On</b> , the container is granted superuser permissions. For example, privileged containers can manipulate network devices on the host machine and modify kernel parameters.

Parameter	Description
Container Resources	<p><b>CPU</b></p> <ul style="list-style-type: none"> <li>– <b>Request:</b> minimum number of CPU cores required by a container. The default value is 0.25 cores.</li> <li>– <b>Limit:</b> maximum number of CPU cores available for a container. Do not leave <b>Limit</b> unspecified. Otherwise, intensive use of container resources will occur and your workload may exhibit unexpected behavior.</li> </ul> <p><b>Memory</b></p> <ul style="list-style-type: none"> <li>– <b>Request:</b> minimum amount of memory required by a container. The default value is 512 MiB.</li> <li>– <b>Limit:</b> maximum amount of memory available for a container. When memory usage exceeds the specified memory limit, the container will be terminated.</li> </ul> <p>For more information about <b>Request</b> and <b>Limit</b>, see <a href="#">Setting Container Specifications</a>.</p> <p><b>GPU:</b> configurable only when the cluster contains GPU nodes.</p> <p>It indicates the percentage of GPU resources reserved for a container. Select <b>Use</b> and set the percentage. For example, if this parameter is set to 10%, the container is allowed to use 10% of GPU resources. If you do not select <b>Use</b> or set this parameter to <b>0</b>, no GPU resources can be used.</p> <p><b>GPU/Graphics Card:</b> The workload's pods will be scheduled to the node with the specified GPU.</p> <p>If <b>Any GPU type</b> is selected, the container uses a random GPU in the node. If you select a specific GPU, the container uses this GPU accordingly.</p>

3. **Lifecycle:** Commands for starting and running containers can be set.
  - **Start Command:** executed when the workload is started. For details, see [Setting Container Startup Commands](#).
  - **Post-Start:** executed after the workload runs successfully. For more information, see [Setting Container Lifecycle Parameters](#).
  - **Pre-Stop:** executed to delete logs or temporary files before the workload ends. For more information, see [Setting Container Lifecycle Parameters](#).
4. **Health Check:** CCE provides two types of probes: liveness probe and readiness probe. They are used to determine whether containers and user services are running properly. For more information, see [Setting Health Check for a Container](#).
  - **Liveness Probe:** used to restart the unhealthy container.
  - **Readiness Probe:** used to change the container to the unready state when detecting that the container is unhealthy. In this way, service traffic will not be directed to the container.

5. **Environment Variables:** Environment variables can be added to a container. In general, environment variables are used to set parameters. On the **Environment Variables** tab page, click **Add Environment Variable**. Currently, three types of environment variables are supported:
  - **Added manually:** Set **Variable Name** and **Variable Value/Reference**.
  - **Added from Secret:** Set **Variable Name** and select the desired secret name and data. A secret must be created in advance. For details, see [Creating a Secret](#).
  - **Added from ConfigMap:** Set **Variable Name** and select the desired ConfigMap name and data. A ConfigMap must be created in advance. For details, see [Creating a ConfigMap](#).

 **NOTE**

To edit an environment variable that has been set, click **Edit**. To delete an environment variable that has been set, click **Delete**.

6. **Data Storage:** Data storage can be mounted to containers for persistent storage and high disk I/O. Local volume and cloud storage are supported. For details, see [Storage \(CSI\)](#).

 **NOTE**

You can add data storage volumes only when creating a StatefulSet.

7. **Security Context:** Container permissions can be configured to protect CCE and other containers from being affected. Enter the user ID to set container permissions and prevent systems and other containers from being affected.
8. **Log Policies:** Log collection policies and log directory can be configured to collect container logs for unified management and analysis. For details, see [Container Logs](#).

**Step 3** Click **Next: Set Application Access** and set **Headless Service** and workload access type.

[Table 17-36](#) describes the parameters in the **Headless Service** area.

**Table 17-36** Parameter description

Parameter	Description
Service Name	Name of the Service corresponding to the workload for mutual access between pods. This Service is used for internal discovery of pods, and does not require an independent IP address or load balancing.
Port Name	Name of the container port. You are advised to enter a name that indicates the function of the port.
Container Port	Listening port inside the container.

Click **Add Service** and set the workload access type.

If your workload will be reachable to other workloads or public networks, add a Service to define the workload access type.

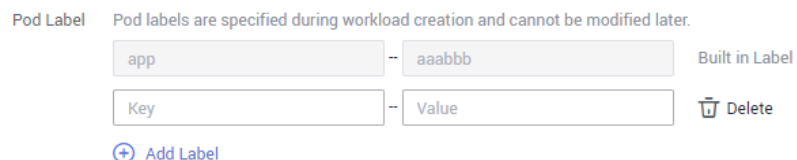
The workload access type determines the network attributes of the workload. Workloads with different access types can provide different network capabilities. For details, see [Overview](#).

**Step 4** Click **Next: Configure Advanced Settings**.

- **Upgrade Policy:** Only **Rolling upgrade** is supported.  
During a rolling upgrade, old pods are gradually replaced with new ones, and service traffic is evenly distributed to both pods to ensure service continuity.
- **Pod Management Policy:** There are two types of policies: ordered and parallel.  
**Ordered:** The StatefulSet will deploy, delete, or scale pods in order and one by one (the StatefulSet waits until each pod is ready before continuing). This is the default policy.  
**Parallel:** The StatefulSet will create pods in parallel to match the desired scale without waiting, and will delete all pods at once.
- **Graceful Deletion:** A time window can be set for workload deletion and reserved for executing commands in the pre-stop phase in the lifecycle. If workload processes are not terminated after the time window elapses, the workload will be forcibly deleted.
  - **Graceful Time Window (s):** Set a time window (0–9999s) for pre-stop commands to finish execution before a workload is deleted. The default value is 30s.
  - **Scale Order:** Choose **Prioritize new pods** or **Prioritize old pods** based on service requirements. **Prioritize new pods** indicates that new pods will be first deleted when a scale-in is triggered.
- **Scheduling Policies:** You can combine static global scheduling policies or dynamic runtime scheduling policies as required. For details, see [Scheduling Policy Overview](#).
- **Advanced Pod Settings**
  - **Pod Label:** The built-in **app** label is specified when the workload is created. It is used to set affinity and anti-affinity scheduling and cannot be modified. You can click **Add Label** to add labels.

**Figure 17-58** Advanced pod settings

Advanced Pod Settings



- **Client DNS Configuration:** A CCE cluster has a built-in DNS add-on (CoreDNS) to provide domain name resolution for workloads in the cluster.
  - **DNS Policy**

- **ClusterFirst:** The default DNS configuration overrides the **Nameserver** and **DNS Search Domain** configurations of the client.
- **None:** Only the **Nameserver** and **DNS Search Domain** configurations are used for domain name resolution.
- **Default:** The pod inherits the DNS configuration from the node on which the pod runs.
- **Nameserver:** You can configure a domain name server for a user-defined domain name. The value is one or a group of DNS IP addresses, for example, 1.2.3.4.
- **DNS Search Domain:** a search list for host-name lookup. When a domain name cannot be resolved, DNS queries will be attempted combining the domain name with each domain in the search list in turn until a match is found or all domains in the search list are tried.
- **Timeout (s):** amount of time the resolver will wait for a response from a remote name server before retrying the query on a different name server. Set it based on the site requirements.
- **ndots:** threshold for the number of dots that must appear in a domain name before an initial absolute query will be made. If a domain name has **ndots** or more than **ndots** dots, the name is a fully qualified domain name (FQDN) and will be tried first as an absolute name. If a domain name has less than **ndots** dots, the operating system will look up the name in a list of search domain names.

**Step 5** Click **Create** and then **Back to StatefulSet List**. If the workload is in the **Running** state, it has been successfully created. If the workload status is not updated, click



in the upper right corner or press **F5** to refresh the page.

**NOTE**

- When a node is unavailable, pods become **Unready**. In this case, you need to manually delete the pods of the StatefulSet so that the pods can be migrated to a normal node.
- If the workload list contains more than 500 records, the Kubernetes pagination mechanism will be used. Specifically, you can only go to the first page or the next page, but cannot go to the previous page. In addition, if resources are divided into discrete pages, the total number of resources displayed is the maximum number of resources that can be queried at a time, not the actual total number of resources.

----End

## Using kubectl

The following procedure uses an etcd workload as an example to describe how to create a workload using kubectl.

**Step 1** Use kubectl to connect to the cluster. For details, see [Connecting to a Cluster Using kubectl](#).

**Step 2** Create and edit the **etcd-statefulset.yaml** file.

**etcd-statefulset.yaml** is an example file name, and you can change it as required.

```
vi etcd-statefulset.yaml
```

The following provides an example of the file contents. For more information on StatefulSet, see the [Kubernetes documentation](#).

```
apiVersion: apps/v1
kind: StatefulSet
metadata:
  name: etcd
spec:
  replicas: 2
  selector:
    matchLabels:
      app: etcd
  serviceName: etcd-svc
  template:
    metadata:
      labels:
        app: etcd
    spec:
      containers:
      - env:
        - name: PAAS_APP_NAME
          value: tesyhhj
        - name: PAAS_NAMESPACE
          value: default
        - name: PAAS_PROJECT_ID
          value: 9632fae707ce4416a0ab1e3e121fe555
        image: etcd # If you use an image from an open-source image registry, enter the image name. If you
        use an image in My Images, obtain the image path from SWR.
        imagePullPolicy: IfNotPresent
        name: container-0
      updateStrategy:
        type: RollingUpdate
```

#### vi etcd-headless.yaml

```
apiVersion: v1
kind: Service
metadata:
  labels:
    app: etcd
  name: etcd-svc
spec:
  clusterIP: None
  ports:
  - name: etcd-svc
    port: 3120
    protocol: TCP
    targetPort: 3120
  selector:
    app: etcd
  sessionAffinity: None
  type: ClusterIP
```

**Step 3** Create a workload and the corresponding headless service.

#### **kubectl create -f etcd-statefulset.yaml**

If the following information is displayed, the StatefulSet has been successfully created.

```
statefulset.apps/etcd created
```

#### **kubectl create -f etcd-headless.yaml**

If the following information is displayed, the headless service has been successfully created.

```
service/etcd-svc created
```

**Step 4** If the workload will be accessed through a ClusterIP or NodePort Service, set the corresponding workload access type. For details, see [Networking](#).

----End

## 17.6.4 Creating a DaemonSet

### Scenario

CCE provides deployment and management capabilities for multiple types of containers and supports features of container workloads, including creation, configuration, monitoring, scaling, upgrade, uninstall, service discovery, and load balancing.

DaemonSet ensures that only one pod runs on all or some nodes. When a node is added to a cluster, a new pod is also added for the node. When a node is removed from a cluster, the pod is also reclaimed. If a DaemonSet is deleted, all pods created by it will be deleted.

The typical application scenarios of a DaemonSet are as follows:

- Run the cluster storage daemon, such as glusterd or Ceph, on each node.
- Run the log collection daemon, such as Fluentd or Logstash, on each node.
- Run the monitoring daemon, such as Prometheus Node Exporter, collectd, Datadog agent, New Relic agent, or Ganglia (gmond), on each node.

You can deploy a DaemonSet for each type of daemons on all nodes, or deploy multiple DaemonSets for the same type of daemons. In the second case, DaemonSets have different flags and different requirements on memory and CPU for different hardware types.

### Prerequisites

You must have one cluster available before creating a DaemonSet. For details on how to create a cluster, see [Buying a CCE Cluster](#).

### Procedure

**Step 1** Log in to the CCE console.

**Step 2** In the navigation pane on the left, choose **Workloads > DaemonSets**. Click **Create DaemonSet** in the upper right corner of the page. Set basic workload parameters as described in [Table 17-37](#). The parameters marked with an asterisk (\*) are mandatory.

**Table 17-37** Basic workload parameters

Parameter	Description
* Workload Name	Name of the containerized workload to be created. The name must be unique. Enter 4 to 63 characters starting with a letter and ending with a letter or digit. Only lowercase letters, digits, and hyphens (-) are allowed.
* Cluster Name	Cluster to which the workload belongs.
* Namespace	In a single cluster, data in different namespaces is isolated from each other. This enables applications to share the services of the same cluster without interfering each other. If no namespace is set, the default namespace is used.
Time Zone Synchronization	If this parameter is enabled, the container and the node use the same time zone. <b>NOTICE</b> After time zone synchronization is enabled, disks of the hostPath type will be automatically added and listed in the <b>Data Storage &gt; Local Volume</b> area. Do not modify or delete the disks.
Description	Description of the workload.

**Step 3** Click **Next: Add Container**.

1. Click **Add Container** and select the image to be deployed.
  - **My Images:** Create a workload using an image in the image repository you created.
  - **Third-Party Images:** Create a workload using an image from any third-party image repository. When you create a workload using a third-party image, ensure that the node where the workload is running can access public networks. For details on how to create a workload using a third-party image, see [Using a Third-Party Image](#).
    - If your image repository does not require authentication, set **Secret Authentication** to **No**, enter an image pull address, and then click **OK**.
    - If your image repository must be authenticated (account and password), you need to create a secret and then use a third-party image. For details, see [Using a Third-Party Image](#).
  - **Shared Images:** Create a workload using an image shared by another tenant through the SWR service.
2. Configure basic image information.  
A workload is an abstract model of a group of pods. One pod can encapsulate one or more containers. You can click **Add Container** in the upper right corner to add multiple container images and set them separately.



**Table 17-38** Image parameters

Parameter	Description
Image Name	Name of the image. You can click <b>Change Image</b> to update it.
*Image Version	Select the image tag to be deployed.
*Container Name	Name of the container. You can modify it.
Privileged Container	Programs in a privileged container have certain privileges. If <b>Privileged Container</b> is <b>On</b> , the container is granted superuser permissions. For example, privileged containers can manipulate network devices on the host machine and modify kernel parameters.
Container Resources	<p><b>CPU</b></p> <ul style="list-style-type: none"> <li>– <b>Request:</b> minimum number of CPU cores required by a container. The default value is 0.25 cores.</li> <li>– <b>Limit:</b> maximum number of CPU cores available for a container. Do not leave <b>Limit</b> unspecified. Otherwise, intensive use of container resources will occur and your workload may exhibit unexpected behavior.</li> </ul> <p><b>Memory</b></p> <ul style="list-style-type: none"> <li>– <b>Request:</b> minimum amount of memory required by a container. The default value is 512 MiB.</li> <li>– <b>Limit:</b> maximum amount of memory available for a container. When memory usage exceeds the specified memory limit, the container will be terminated.</li> </ul> <p>For more information about <b>Request</b> and <b>Limit</b>, see <a href="#">Setting Container Specifications</a>.</p> <p><b>GPU:</b> configurable only when the cluster contains GPU nodes.</p> <p>It indicates the percentage of GPU resources reserved for a container. Select <b>Use</b> and set the percentage. For example, if this parameter is set to 10%, the container is allowed to use 10% of GPU resources. If you do not select <b>Use</b> or set this parameter to <b>0</b>, no GPU resources can be used.</p> <p><b>GPU/Graphics Card:</b> The workload's pods will be scheduled to the node with the specified GPU.</p> <p>If <b>Any GPU type</b> is selected, the container uses a random GPU in the node. If you select a specific GPU, the container uses this GPU accordingly.</p>

3. **Lifecycle:** Commands for starting and running containers can be set.
  - **Start Command:** executed when the workload is started. For details, see [Setting Container Startup Commands](#).

- **Post-Start:** executed after the workload runs successfully. For more information, see [Setting Container Lifecycle Parameters](#).
  - **Pre-Stop:** executed to delete logs or temporary files before the workload ends. For more information, see [Setting Container Lifecycle Parameters](#).
4. **Health Check:** CCE provides two types of probes: liveness probe and readiness probe. They are used to determine whether containers and user services are running properly. For more information, see [Setting Health Check for a Container](#).
- **Liveness Probe:** used to restart the unhealthy container.
  - **Readiness Probe:** used to change the container to the unready state when detecting that the container is unhealthy. In this way, service traffic will not be directed to the container.
5. **Environment Variables:** Environment variables can be added to a container. In general, environment variables are used to set parameters. On the **Environment Variables** tab page, click **Add Environment Variable**. Currently, three types of environment variables are supported:
- **Added manually:** Set **Variable Name** and **Variable Value/Reference**.
  - **Added from Secret:** Set **Variable Name** and select the desired secret name and data. A secret must be created in advance. For details, see [Creating a Secret](#).
  - **Added from ConfigMap:** Set **Variable Name** and select the desired ConfigMap name and data. A ConfigMap must be created in advance. For details, see [Creating a ConfigMap](#).

 NOTE

To edit an environment variable that has been set, click **Edit**. To delete an environment variable that has been set, click **Delete**.

6. **Data Storage:** Data storage can be mounted to containers for persistent storage and high disk I/O. Local volume and cloud storage are supported. For details, see [Storage \(CSI\)](#).
7. **Security Context:** Container permissions can be configured to protect CCE and other containers from being affected. Enter the user ID to set container permissions and prevent systems and other containers from being affected.
8. **Log Policies:** Log collection policies and log directory can be configured to collect container logs for unified management and analysis. For details, see [Container Logs](#).

**Step 4** Click **Next: Set Application Access**. Then, click **Add Service** and set the workload access type.

If your workload will be reachable to other workloads or public networks, add a Service to define the workload access type.

The workload access type determines the network attributes of the workload. Workloads with different access types can provide different network capabilities. For details, see [Overview](#).

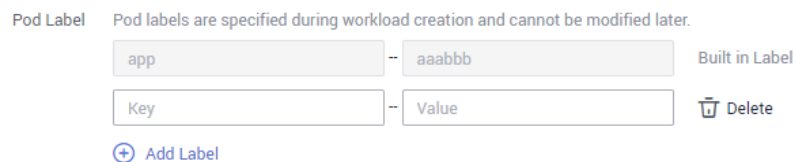
**Step 5** Click **Next: Configure Advanced Settings** to configure advanced policies.

- **Upgrade Policy:**

- **Upgrade Mode:** Only **Rolling upgrade** is supported. During a rolling upgrade, old pods are gradually replaced with new ones. During the upgrade, service traffic is evenly distributed to both pods to ensure service continuity.
- **Maximum Number of Unavailable 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. Minimum number of alive pods = Total pods - Maximum number of unavailable pods
- **Graceful Deletion:**
  - Graceful Time Window:** Enter the time. The graceful scale-in policy provides a time window for workload deletion and is reserved for executing commands in the PreStop phase in the lifecycle. If workload processes are not terminated after the time window elapses, the workload will be forcibly deleted.
- **Scheduling Policies:** You can combine static global scheduling policies or dynamic runtime scheduling policies as required. For details, see [Scheduling Policy Overview](#).
- **Advanced Pod Settings**
  - **Pod Label:** The built-in **app** label is specified when the workload is created. It is used to set affinity and anti-affinity scheduling and cannot be modified. You can click **Add Label** to add labels.

Figure 17-59 Advanced pod settings

Advanced Pod Settings




- **Client DNS Configuration:** A CCE cluster has a built-in DNS add-on (CoreDNS) to provide domain name resolution for workloads in the cluster.
  - **DNS Policy**
    - **ClusterFirst:** The default DNS configuration overrides the **Nameserver** and **DNS Search Domain** configurations of the client.
    - **None:** Only the **Nameserver** and **DNS Search Domain** configurations are used for domain name resolution.
    - **Default:** The pod inherits the DNS configuration from the node on which the pod runs.
  - **Nameserver:** You can configure a domain name server for a user-defined domain name. The value is one or a group of DNS IP addresses, for example, 1.2.3.4.
  - **DNS Search Domain:** a search list for host-name lookup. When a domain name cannot be resolved, DNS queries will be attempted combining the domain name with each domain in the search list in turn until a match is found or all domains in the search list are tried.

- **Timeout (s):** amount of time the resolver will wait for a response from a remote name server before retrying the query on a different name server. Set it based on the site requirements.
- **ndots:** threshold for the number of dots that must appear in a domain name before an initial absolute query will be made. If a domain name has **ndots** or more than **ndots** dots, the name is a fully qualified domain name (FQDN) and will be tried first as an absolute name. If a domain name has less than **ndots** dots, the operating system will look up the name in a list of search domain names.

**Step 6** After the preceding configurations are complete, click **Create**. On the page displayed, click **Return to Workload List** to view the workload status.

If the workload is in the **Running** state, it has been successfully created.

Workload status is not updated in real time. Click  in the upper right corner or press **F5** to refresh the page.

----End

## 17.6.5 Creating a Job

### Scenario

Jobs are short-lived and run for a certain time to completion. They can be executed immediately after being deployed. It is completed after it exits normally (exit 0).

A job is a resource object that is used to control batch tasks. It is different from a long-term servo workload (such as Deployment and StatefulSet).

A job is started and terminated at specific times, while a long-term servo workload runs unceasingly unless being terminated. The pods managed by a job automatically exit after successfully completing the job based on user configurations. The success flag varies according to the spec.completions policy.

- One-off jobs: A single pod runs once until successful termination.
- Jobs with a fixed success count: N pods run until successful termination.
- A queue job is considered completed based on the global success confirmed by the application.

### Prerequisites

Resources have been created. For details, see [Buying a Node](#). If clusters and nodes are available, you need not create them again.

### Procedure

**Step 1** (Optional) If you use a private container image to create your job, upload the container image to the image repository.

**Step 2** Log in to the CCE console. In the navigation pane, choose **Workloads > Jobs**. Click **Create Job**.

**Step 3** Configure the basic job information listed in [Table 17-39](#). The parameters marked with an asterisk (\*) are mandatory.

**Table 17-39** Basic job information

Parameter	Description
* Job Name	Name of a new job. The name must be unique. Enter 4 to 63 characters starting with a lowercase letter and ending with a letter or digit. Only lowercase letters, digits, and hyphens (-) are allowed.
* Cluster	Cluster to which a new job belongs.
* Namespace	Namespace to which the new job belongs. By default, this parameter is set to <b>default</b> .
*Instances	Number of pods in this job. A job can have one or more pods. You can specify the number of pods. The default value is <b>1</b> . Each job pod consists of the same containers. Configuring multiple job pods can ensure high availability. The job can continue to run even if one of the pods is faulty.
Description	Description of a job.

**Step 4** Click **Next: Add Container** to add a container and an image.

1. Click **Select Container Image** to select the image to be deployed.
  - **My Images:** displays all image repositories you created.
  - **Third-Party Images:** Create a job using an image from any third-party image repository. When you create a job using a third-party image, ensure that the node where the job is running can access public networks. For details about how to use a third-party image, see [Using a Third-Party Image](#).
    - If your image repository does not require authentication, set **Secret Authentication** to **No**, enter an image address in **Image Address**, and then click **OK**.
    - If your image repository must be authenticated (account and password), you need to create a secret and then use a third-party image. For details, see [Using a Third-Party Image](#).
  - **Shared Images:** The images shared by other tenants using the SWR service are displayed here. You can create workloads based on the shared images.
2. Set image parameters.

**Table 17-40** Image parameters

Parameter	Description
Image	Name of the image. You can click <b>Change Image</b> to update it.
*Image Version	Select the image tag to be deployed.
*Container Name	Name of the container. You can modify it.
Container Resources	<p><b>CPU</b></p> <ul style="list-style-type: none"> <li>– <b>Request:</b> minimum number of CPU cores required by a container. The default value is 0.25 cores.</li> <li>– <b>Limit:</b> maximum number of CPU cores available for a container. Do not leave <b>Limit</b> unspecified. Otherwise, intensive use of container resources will occur and your workload may exhibit unexpected behavior.</li> </ul> <p><b>Memory</b></p> <ul style="list-style-type: none"> <li>– <b>Request:</b> minimum amount of memory required by a container. The default value is 0.5 GiB.</li> <li>– <b>Limit:</b> maximum amount of memory available for a container. When memory usage exceeds the specified memory limit, the container will be terminated.</li> </ul> <p>For more information about <b>Request</b> and <b>Limit</b>, see <a href="#">Setting Container Specifications</a>.</p> <p><b>GPU:</b> configurable only when the cluster contains GPU nodes.</p> <p>It indicates the percentage of GPU resources reserved for a container. Select <b>Use</b> and set the percentage. For example, if this parameter is set to 10%, the container is allowed to use 10% of GPU resources. If you do not select <b>Use</b> or set this parameter to <b>0</b>, no GPU resources can be used.</p> <p><b>GPU/Graphics Card:</b> The workload's pods will be scheduled to the node with the specified GPU.</p> <p>If <b>Any GPU type</b> is selected, the container uses a random GPU in the node. If you select a specific GPU, the container uses that GPU accordingly.</p>

3. (Optional) Configure advanced settings.

**Table 17-41** Advanced settings

Parameter	Description
Lifecycle	<p>Lifecycle scripts define the actions taken for container-related jobs when a lifecycle event occurs.</p> <ul style="list-style-type: none"> <li>– <b>Start Command:</b> You can set the command to be executed immediately after the container is started. For details, see <a href="#">Configuring a Container</a>.</li> <li>– <b>Post-Start:</b> The command is triggered after a job starts. For details, see <a href="#">Setting Container Lifecycle Parameters</a>.</li> <li>– <b>Pre-Stop:</b> The command is triggered before a job is stopped. For details, see <a href="#">Setting Container Lifecycle Parameters</a>.</li> </ul>
Environment Variables	<p>Environment variables can be added to a container. In general, environment variables are used to set parameters. On the <b>Environment Variables</b> tab page, click <b>Add Environment Variable</b>. Currently, environment variables can be added using any of the following methods:</p> <ul style="list-style-type: none"> <li>– <b>Added manually:</b> Set <b>Variable Name</b> and <b>Variable Value/Reference</b>.</li> <li>– <b>Added from Secret:</b> Set <b>Variable Name</b> and select the desired secret name and data. A secret must be created in advance. For details, see <a href="#">Creating a Secret</a>.</li> <li>– <b>Added from ConfigMap:</b> Set <b>Variable Name</b> and select the desired ConfigMap name and data. A ConfigMap must be created in advance. For details, see <a href="#">Creating a ConfigMap</a>.</li> </ul>
Data Storage	<p>The local disk or cloud storage can be mounted to a container to implement persistent data file storage. For details, see <a href="#">Storage (CSI)</a>.</p>
Log Policies	<p>Set a log policy and log path for collecting workload logs and preventing logs from being over-sized. For details, see <a href="#">Container Logs</a>.</p>

4. (Optional) One job pod contains one or more related containers. If your job contains multiple containers, click **Add Container** to add containers.

**Step 5** Click **Create**.

If the status of the job is **Executing**, the job has been created successfully.

----End

## Using kubectl

A job has the following configuration parameters:

- **spec.template:** has the same schema as a pod.
- **RestartPolicy:** can only be set to **Never** or **OnFailure**.
- For a single-pod job, the job ends after the pod runs successfully by default.
- **.spec.completions:** indicates the number of pods that need to run successfully to end a job. The default value is **1**.
- **.spec.parallelism:** indicates the number of pods that run concurrently. The default value is **1**.
- **spec.backoffLimit:** indicates the maximum number of retries performed if a pod fails. When the limit is reached, the pod will not try again.
- **.spec.activeDeadlineSeconds:** indicates the running time of pods. Once the time is reached, all pods of the job are terminated. The priority of **.spec.activeDeadlineSeconds** is higher than that of **.spec.backoffLimit**. That is, if a job reaches the **.spec.activeDeadlineSeconds**, the **spec.backoffLimit** is ignored.

Based on the **.spec.completions** and **.spec.Parallelism** settings, jobs are classified into the following types.

**Table 17-42** Job types

Job Type	Description	Example
One-off jobs	A single pod runs once until successful termination.	Database migration
Jobs with a fixed completion count	One pod runs until reaching the specified <b>completions</b> count.	Work queue processing pod
Parallel jobs with a fixed completion count	Multiple pods run until reaching the specified <b>completions</b> count.	Multiple pods for processing work queues concurrently
Parallel jobs	One or more pods run until successful termination.	Multiple pods for processing work queues concurrently

The following is an example job, which calculates  $\pi$  till the 2000<sup>th</sup> digit and prints the output.

```
apiVersion: batch/v1
kind: Job
metadata:
  name: myjob
spec:
  completions: 50      # 50 pods need to be run to finish a job. In this example,  $\pi$  is printed for 50 times.
  parallelism: 5      # 5 pods are run in parallel.
  backoffLimit: 5     # The maximum number of retry times is 5.
  template:
    spec:
      containers:
        - name: pi
          image: perl
```



```
command: ["perl", "-Mbignum=bpi", "-wle", "print bpi(2000)"]
restartPolicy: Never
```

### Description

- **apiVersion: batch/v1** indicates the version of the current job.
- **kind: Job** indicates that the current resource is a job.
- **restartPolicy: Never** indicates the current restart policy. For jobs, this parameter can only be set to **Never** or **OnFailure**. For other controllers (for example, Deployments), you can set this parameter to **Always**.

### Run the job.

#### Step 1 Start the job.

```
[root@k8s-master k8s]# kubectl apply -f myjob.yaml
job.batch/myjob created
```

#### Step 2 View the job details.

##### kubectl get job

```
[root@k8s-master k8s]# kubectl get job
NAME      COMPLETIONS  DURATION  AGE
myjob    50/50         23s      3m45s
```

If the value of **COMPLETIONS** is **50/50**, the job is successfully executed.

#### Step 3 Query the pod status.

##### kubectl get pod

```
[root@k8s-master k8s]# kubectl get pod
NAME      READY  STATUS   RESTARTS  AGE
myjob-29qlw  0/1    Completed  0         4m5s
...
```

If the status is **Completed**, the job is complete.

#### Step 4 View the pod logs.

##### kubectl logs

```
# kubectl logs myjob-29qlw
3.141592653589793238462643383279502884197169399375105820974944592307816406286208998628034
8253421170679821480865132823066470938446095505822317253594081284811174502841027019385211
0555964462294895493038196442881097566593344612847564823378678316527120190914564856692346
0348610454326648213393607260249141273724587006606315588174881520920962829254091715364367
8925903600113305305488204665213841469519415116094330572703657595919530921861173819326117
9310511854807446237996274956735188575272489122793818301194912983367336244065664308602139
4946395224737190702179860943702770539217176293176752384674818467669405132000568127145263
5608277857713427577896091736371787214684409012249534301465495853710507922796892589235420
199561121290219608640344181598136297747713099605187072113499999837297804995105973173281
6096318595024459455346908302642522308253344685035261931188171010003137838752886587533208
3814206171776691473035982534904287554687311595628638823537875937519577818577805321712268
0661300192787661119590921642019893809525720106548586327886593615338182796823030195203530
1852968995773622599413891249721775283479131515574857242454150695950829533116861727855889
0750983817546374649393192550604009277016711390098488240128583616035637076601047101819429
5559619894676783744944825537977472684710404753464620804668425906949129331367702898915210
4752162056966024058038150193511253382430035587640247496473263914199272604269922796782354
7816360093417216412199245863150302861829745557067498385054945885869269956909272107975093
0295532116534498720275596023648066549911988183479775356636980742654252786255181841757467
289097772793800081647060016145249192173217214772350141441973568548161361157352552133475
7418494684385233239073941433345477624168625189835694855620992192221842725502542568876717
9049460165346680498862723279178608578438382796797668145410095388378636095068006422512520
```

```
5117392984896084128488626945604241965285022210661186306744278622039194945047123713786960
9563643719172874677646575739624138908658326459958133904780275901
```

----End

## Related Operations

After a one-off job is created, you can perform operations listed in [Table 17-43](#).

**Table 17-43** Other operations

Operation	Description
Viewing a YAML	Click <b>View YAML</b> next to the job name to view the YAML file corresponding to the current job.
Deleting a one-off job	<ol style="list-style-type: none"> <li>1. Select the job to be deleted and click <b>Delete</b> in the <b>Operation</b> column.</li> <li>2. Click <b>OK</b>. Deleted jobs cannot be restored. Exercise caution when deleting a job.</li> </ol>

## 17.6.6 Creating a Cron Job

### Scenario

A cron job runs on a repeating schedule. You can perform time synchronization for all active nodes at a fixed time point.

A cron job runs periodically at the specified time. It is similar with Linux crontab. A cron job has the following characteristics:

- Runs only once at the specified time.
- Runs periodically at the specified time.

The typical usage of a cron job is as follows:

- Schedules jobs at the specified time.
- Creates jobs to run periodically, for example, database backup and email sending.

### Prerequisites

Resources have been created. For details, see [Buying a Node](#). If clusters and nodes are available, you need not create them again.

### Procedure

- Step 1** (Optional) If you use a private container image to create your cron job, upload the container image to the image repository.
- Step 2** Log in to the CCE console. In the navigation pane, choose **Workloads > Cron Jobs**. Then, click **Create Cron Job**.

**Step 3** Configure the basic cron job information listed in [Table 17-44](#). The parameters marked with an asterisk (\*) are mandatory.

**Table 17-44** Basic cron job information

Parameter	Description
* Job Name	Name of a new cron job. The name must be unique. Enter 4 to 52 characters starting with a lowercase letter and ending with a letter or digit. Only lowercase letters, digits, and hyphens (-) are allowed.
* Cluster	Cluster to which a new cron job belongs.
* Namespace	Namespace to which a cron job belongs. If you do not specify this parameter, the value <b>default</b> is used by default.
Description	Description of a cron job.

**Step 4** Click **Next: Configure Timing Rule**.

**Step 5** Set the timing rule.

**Table 17-45** Timing rule parameters

Parameter	Description
* Concurrency Policy	The following policies are supported: <ul style="list-style-type: none"> <li>• <b>Forbid</b>: A new job cannot be created before the previous job is complete.</li> <li>• <b>Allow</b>: The cron job allows concurrently running jobs, which preempt cluster resources.</li> <li>• <b>Replace</b>: A new job replaces the previous job when it is time to create the job but the previous job is not complete.</li> </ul>
* Schedule	Time when a new cron job is executed.
Job Records	You can set the number of jobs that are successfully executed or fail to be executed. Setting a limit to <b>0</b> corresponds to keeping none of the jobs after they finish.

**Step 6** Click **Next: Add Container** to add a container.

1. Click **Select Container Image** to select the image to be deployed.
  - **My Images**: displays all image repositories you created.
  - **Third-Party Images**: Create a job using an image from any third-party image repository. When you create a job using a third-party image, ensure that the node where the job is running can access public networks. For details about how to use a third-party image, see [Using a Third-Party Image](#).

- If your image repository does not require authentication, set **Secret Authentication** to **No**, enter an image address in **Image Address**, and then click **OK**.
    - If your image repository must be authenticated (account and password), you need to create a secret and then use a third-party image. For details, see [Using a Third-Party Image](#).
  - **Shared Images:** The images shared by other tenants using the SWR service are displayed here. You can create workloads based on the shared images.
2. Set image parameters.

**Table 17-46** Image parameters

Parameter	Description
Image	Name of the image. You can click <b>Change Image</b> to update it.
*Image Version	Select the image tag to be deployed.
*Container Name	Name of the container. You can modify it.

Parameter	Description
Container Resources	<p><b>CPU</b></p> <ul style="list-style-type: none"> <li>– <b>Request:</b> minimum number of CPU cores required by a container. The default value is 0.25 cores.</li> <li>– <b>Limit:</b> maximum number of CPU cores available for a container. Do not leave <b>Limit</b> unspecified. Otherwise, intensive use of container resources will occur and your workload may exhibit unexpected behavior.</li> </ul> <p><b>Memory</b></p> <ul style="list-style-type: none"> <li>– <b>Request:</b> minimum amount of memory required by a container. The default value is 0.5 GiB.</li> <li>– <b>Limit:</b> maximum amount of memory available for a container. When memory usage exceeds the specified memory limit, the container will be terminated.</li> </ul> <p>For more information about <b>Request</b> and <b>Limit</b>, see <a href="#">Setting Container Specifications</a>.</p> <p><b>GPU:</b> configurable only when the cluster contains GPU nodes.</p> <p>It indicates the percentage of GPU resources reserved for a container. Select <b>Use</b> and set the percentage. For example, if this parameter is set to 10%, the container is allowed to use 10% of GPU resources. If you do not select <b>Use</b> or set this parameter to <b>0</b>, no GPU resources can be used.</p> <p><b>GPU/Graphics Card:</b> The workload's pods will be scheduled to the node with the specified GPU.</p> <p>If <b>Any GPU type</b> is selected, the container uses a random GPU in the node. If you select a specific GPU, the container uses that GPU accordingly.</p>

3. (Optional) Configure advanced settings.

**Table 17-47** Advanced settings

Parameter	Description
Lifecycle	<p>Actions defined in the lifecycle script definition are taken for the lifecycle events of container tasks.</p> <ul style="list-style-type: none"> <li>– <b>Start Command:</b> You can set the command to be executed immediately after the container is started. For details, see <a href="#">Configuring a Container</a>.</li> <li>– <b>Post-Start:</b> The command is triggered after a job starts. For details, see <a href="#">Setting Container Lifecycle Parameters</a>.</li> <li>– <b>Pre-Stop:</b> The command is triggered before a job is stopped. For details, see <a href="#">Setting Container Lifecycle Parameters</a>.</li> </ul>

Parameter	Description
Environment Variables	<p>Environment variables can be added to a container. In general, environment variables are used to set parameters. On the <b>Environment Variables</b> tab page, click <b>Add Environment Variable</b>. Currently, environment variables can be added using any of the following methods:</p> <ul style="list-style-type: none"> <li>– <b>Added manually:</b> Set <b>Variable Name</b> and <b>Variable Value/Reference</b>.</li> <li>– <b>Added from Secret:</b> Set <b>Variable Name</b> and select the desired secret name and data. A secret must be created in advance. For details, see <a href="#">Creating a Secret</a>.</li> <li>– <b>Added from ConfigMap:</b> Set <b>Variable Name</b> and select the desired ConfigMap name and data. A ConfigMap must be created in advance. For details, see <a href="#">Creating a ConfigMap</a>.</li> </ul>

4. (Optional) One job pod contains one or more related containers. If your cron job contains multiple containers, click **Add Container** to add containers.

**Step 7** Click **Create**.

If the status is **Started**, the cron job has been created successfully.

----End

## Using kubectl

A cron job has the following configuration parameters:

- **.spec.schedule:** takes a [Cron](#) format string, for example, `0 * * * *` or `@hourly`, as schedule time of jobs to be created and executed.
- **.spec.jobTemplate:** specifies jobs to be run, and has the same schema as when you are [Creating a Job Using kubectl](#).
- **.spec.startingDeadlineSeconds:** specifies the deadline for starting a job.
- **.spec.concurrencyPolicy:** specifies how to treat concurrent executions of a job created by the Cron job. The following options are supported:
  - **Allow** (default value): allows concurrently running jobs.
  - **Forbid:** forbids concurrent runs, skipping next run if previous has not finished yet.
  - **Replace:** cancels the currently running job and replaces it with a new one.

The following is an example cron job, which is saved in the `cronjob.yaml` file.

```
apiVersion: batch/v1beta1
kind: CronJob
metadata:
  name: hello
spec:
  schedule: "*/1 * * * *"
  jobTemplate:
```

```
spec:
  template:
    spec:
      containers:
      - name: hello
        image: busybox
        args:
        - /bin/sh
        - -c
        - date; echo Hello from the Kubernetes cluster
      restartPolicy: OnFailure
```

### Run the job.

**Step 1** Create a cron job.

**kubectl create -f cronjob.yaml**

Information similar to the following is displayed:

```
cronjob.batch/hello created
```

**Step 2** Query the running status of the cron job:

**kubectl get cronjob**

NAME	SCHEDULE	SUSPEND	ACTIVE	LAST SCHEDULE	AGE
hello	*/* * * * *	False	0	<none>	9s

**kubectl get jobs**

NAME	COMPLETIONS	DURATION	AGE
hello-1597387980	1/1	27s	45s

**kubectl get pod**

NAME	READY	STATUS	RESTARTS	AGE
hello-1597387980-tjv8f	0/1	Completed	0	114s
hello-1597388040-lckg9	0/1	Completed	0	39s

**kubectl logs hello-1597387980-tjv8f**

```
Fri Aug 14 06:56:31 UTC 2020
Hello from the Kubernetes cluster
```

**kubectl delete cronjob hello**

```
cronjob.batch "hello" deleted
```

### NOTICE

When a cron job is deleted, the related jobs and pods are deleted too.

----End

## Related Operations

After a cron job is created, you can perform operations listed in [Table 17-48](#).

**Table 17-48** Other operations

Operation	Description
Editing a YAML file	Click <b>More &gt; View YAML</b> next to the cron job name to view the YAML file of the current job.
Stopping a cron job	<ol style="list-style-type: none"> <li>1. Select the job to be stopped and click <b>Stop</b> in the <b>Operation</b> column.</li> <li>2. Click <b>OK</b>.</li> </ol>
Deleting a cron job	<ol style="list-style-type: none"> <li>1. Select the cron job to be deleted and click <b>More &gt; Delete</b> in the <b>Operation</b> column.</li> <li>2. Click <b>OK</b>. Deleted jobs cannot be restored. Therefore, exercise caution when deleting a job.</li> </ol>

## 17.6.7 Managing Pods

### Scenario

A pod is the smallest and simplest unit in the Kubernetes object model that you create or deploy. A pod encapsulates an application's container (or, in some cases, multiple containers), storage resources, a unique network identity (IP address), as well as options that govern how the container(s) should run. A pod represents a single instance of an application in Kubernetes, which might consist of either a single container or a small number of containers that are tightly coupled and that share resources.

Pods in a Kubernetes cluster can be used in either of the following ways:

- **Pods that run a single container.** The "one-container-per-pod" model is the most common Kubernetes use case. In this case, a pod functions as a wrapper around a single container, and Kubernetes manages the pods rather than the containers directly.
- **Pods that run multiple containers that need to work together.** A pod might encapsulate an application composed of multiple co-located containers that are tightly coupled and need to share resources. The possible scenarios are as follows:
  - Content management systems, file and data loaders, local cache managers, etc;
  - Log and checkpoint backup, compression, rotation, snapshotting, etc;
  - Data change watchers, log tailers, logging and monitoring adapters, event publishers, etc;
  - Proxies, bridges, adapters, etc;
  - Controllers, managers, configurators, and updaters

You can easily manage pods on CCE, such as editing YAML files and monitoring pods.



## Editing a YAML File

To edit and download the YAML file of a pod online, do as follows:

- Step 1** Log in to the CCE console. In the navigation pane, choose **Workloads > Pods**.
- Step 2** Click **Edit YAML** at the same row as the target pod. In the **Edit YAML** dialog box displayed, modify the YAML file of the pod.
- Step 3** Click **Edit** and then **OK** to save the changes.

### NOTE

If a pod is created by another workload, its YAML file cannot be modified individually on the **Pods** page.

- Step 4** (Optional) In the **Edit YAML** window, click **Download** to download the YAML file.

----End

## Monitoring Pods

On the CCE console, you can view the CPU and memory usage, upstream and downstream rates, and disk read/write rates of a workload pod to determine the required resource specifications.

- Step 1** Log in to the CCE console. In the navigation pane, choose **Workloads > Pods**.
- Step 2** Click **Monitoring** at the same row as the target pod to view the CPU and memory usage, upstream and downstream rates, and disk read/write rates of the pod.

### NOTE

You cannot view the monitoring data of a pod that is not running.

----End

## Deleting a Pod

If a pod is no longer needed, you can delete it. Deleted pods cannot be recovered. Exercise caution when performing this operation.

- Step 1** Log in to the CCE console. In the navigation pane, choose **Workloads > Pods**.
- Step 2** Click **Delete** at the same row as the pod to be deleted.

Read the system prompts carefully. A pod cannot be restored after it is deleted. Exercise caution when performing this operation.

- Step 3** Click **Yes** to delete the pod.

### NOTE

- If the node where the pod is located is unavailable or shut down and the workload cannot be deleted, you can forcibly delete the pod from the pod list on the workload details page.
- Ensure that the storage volumes to be deleted are not used by other workloads. If these volumes are imported or have snapshots, you can only unbind them.

----End

## Helpful Links

- [The Distributed System Toolkit: Patterns for Composite Containers](#)
- [Container Design Patterns](#)

## 17.6.8 GPU Scheduling

You can use GPUs in CCE containers.

### Prerequisites

- A GPU node has been ready for use. For details, see [Buying a Node](#).
- The gpu-beta add-on has been installed. During the installation, select the GPU driver on the node. For details, see [gpu-beta](#).

### Using GPUs

Create a workload and request GPUs. You can specify the number of GPUs as follows:

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: gpu-test
  namespace: default
spec:
  replicas: 1
  selector:
    matchLabels:
      app: gpu-test
  template:
    metadata:
      labels:
        app: gpu-test
    spec:
      containers:
      - image: nginx:perl
        name: container-0
        resources:
          requests:
            cpu: 250m
            memory: 512Mi
            nvidia.com/gpu: 1 # Number of requested GPUs
          limits:
            cpu: 250m
            memory: 512Mi
            nvidia.com/gpu: 1 # Maximum number of GPUs that can be used
        imagePullSecrets:
        - name: default-secret
```

**nvidia.com/gpu** specifies the number of GPUs to be requested. The value can be smaller than 1. For example, **nvidia.com/gpu: 0.5** indicates that multiple pods share a GPU.

After **nvidia.com/gpu** is specified, workloads will not be scheduled to nodes without GPUs. If GPUs are insufficient, a Kubernetes event similar to "0/2 nodes are available: 2 Insufficient nvidia.com/gpu." will be reported.

To use GPUs on the CCE console, select the GPU quota and specify the percentage of GPUs reserved for the container when creating a workload.

**Figure 17-60** Using GPUs

Container Resources

**CPU**  Request  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  cores

---

**Memory**  Request  MIB  
 Limit  MIB  
If the memory limit is exceeded, the container will be terminated.

**i** The cluster where the GPU-accelerated nodes will reside must have the [gpu-beta](#) add-on installed. Otherwise, workloads will not run properly when GPU resources are used. ✕

**GPU**  Use  %  
Percentage of GPU resources reserved for the container.

## GPU Node Labels

CCE will label GPU-enabled nodes that are ready to use. Different types of GPU-enabled nodes have different labels.

**Figure 17-61** GPU node labels

✕

### Manage Labels

node.kubernetes.io/subnetid	0c45e406-a929-49f0-8987-0f509701695e	
os.architecture	amd64	
os.name	EulerOS_2.0_SP9x86_64	
os.version	4.18.0-147.5.1.6.h541.eulerosv2r9.x86_64	
topology.kubernetes.io/region	cn-north-7	
topology.kubernetes.io/zone	cn-north-7c	
<input type="text" value="accelerator"/>	<input type="text" value="nvidia-t4"/>	<a href="#">Delete</a>

+ Add Label

OK
Cancel

When using GPUs, you can enable the affinity between pods and nodes based on labels so that the pods can be scheduled to the correct nodes.

```
apiVersion: apps/v1
kind: Deployment
```

```

metadata:
  name: gpu-test
  namespace: default
spec:
  replicas: 1
  selector:
    matchLabels:
      app: gpu-test
  template:
    metadata:
      labels:
        app: gpu-test
    spec:
      nodeSelector:
        accelerator: nvidia-t4
      containers:
      - image: nginx:perl
        name: container-0
        resources:
          requests:
            cpu: 250m
            memory: 512Mi
            nvidia.com/gpu: 1 # Number of requested GPUs
          limits:
            cpu: 250m
            memory: 512Mi
            nvidia.com/gpu: 1 # Maximum number of GPUs that can be used
        imagePullSecrets:
        - name: default-secret

```

## 17.6.9 NPU Scheduling

You can use NPUs in CCE containers.

### Prerequisites

- An NPU node has been ready for use. For details, see [Buying a Node](#).
- The huawei-npu has been installed. For details, see [huawei-npu](#).

### Using NPUs

Create a workload and request NPUs. You can specify the number of NPUs as follows:

```

kind: Deployment
apiVersion: apps/v1
metadata:
  name: npu-test
  namespace: default
spec:
  replicas: 1
  selector:
    matchLabels:
      app: npu-test
  template:
    metadata:
      labels:
        app: npu-test
    spec:
      containers:
      - name: container-0
        image: nginx:perl
        resources:
          limits:
            cpu: 250m
            huawei.com/ascend-310: '1'

```

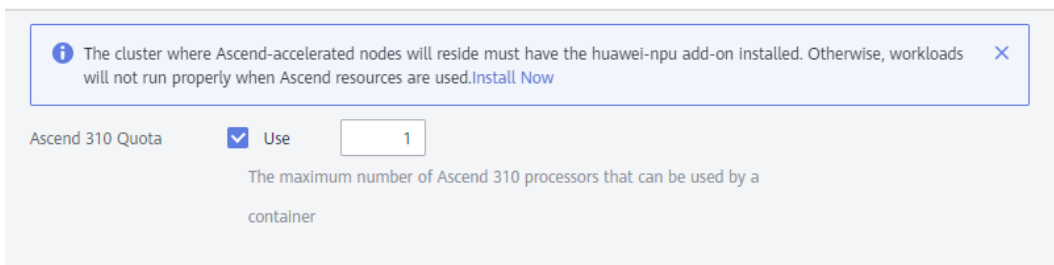
```
memory: 512Mi
requests:
  cpu: 250m
  huawei.com/ascend-310: '1'
memory: 512Mi
imagePullSecrets:
- name: default-secret
```

Specify the number of NPUs to be requested in **huawei.com/ascend-310**.

After **huawei.com/ascend-310** is specified, workloads will not be scheduled to nodes without NPUs. If NPUs are insufficient, a Kubernetes event similar to "0/2 nodes are available: 2 Insufficient huawei.com/ascend-310." will be reported.

To use NPUs on the CCE console, select the Ascend 310 quota and specify the number of Ascend chips to be used when creating a workload.

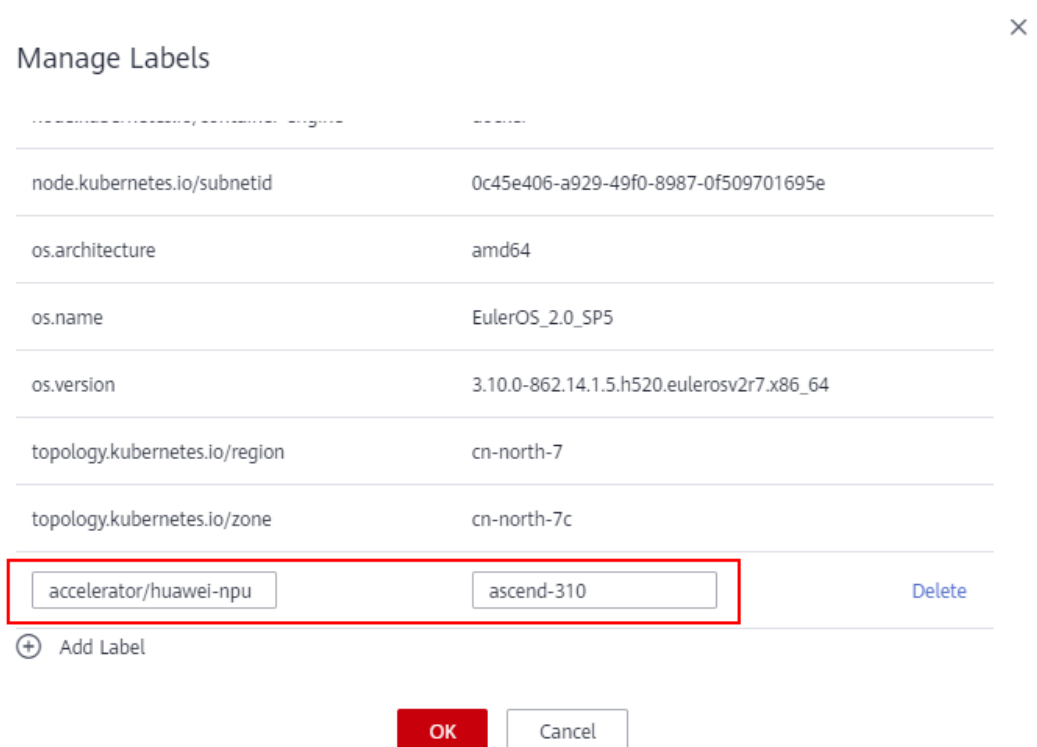
**Figure 17-62** Using NPUs



## NPU Node Labels

CCE will label NPU-enabled nodes that are ready to use.

**Figure 17-63** NPU node labels



When using NPUs, you can enable the affinity between pods and nodes based on labels so that the pods can be scheduled to the correct nodes.

```
kind: Deployment
apiVersion: apps/v1
metadata:
  name: npu-test
  namespace: default
spec:
  replicas: 1
  selector:
    matchLabels:
      app: npu-test
  template:
    metadata:
      labels:
        app: npu-test
    spec:
      nodeSelector:
        accelerator/huawei-npu: ascend-310
      containers:
        - name: container-0
          image: nginx:perl
          resources:
            limits:
              cpu: 250m
              huawei.com/ascend-310: '1'
              memory: 512Mi
            requests:
              cpu: 250m
              huawei.com/ascend-310: '1'
              memory: 512Mi
      imagePullSecrets:
        - name: default-secret
```

## 17.6.10 Managing Workloads and Jobs

### Scenario

After a workload is created, you can scale, upgrade, monitor, roll back, or delete the workload, as well as edit its YAML file.

**Table 17-49** Workload/Job management

Operation	Description
<b>Logging</b>	You can view logs of Deployments, StatefulSets, DaemonSets, and jobs.
<b>Upgrade</b>	You can replace images or image tags to quickly upgrade Deployments, StatefulSets, and DaemonSets without interrupting services.
<b>Editing a YAML file</b>	You can modify and download the YAML files of Deployments, StatefulSets, DaemonSets, and pods on the CCE console. YAML files of jobs and cron jobs can only be viewed, copied, and downloaded.
<b>Scaling</b>	A workload can be automatically resized according to scaling policies, freeing you from the efforts to manually adjust resources for fluctuating service traffic. This saves you big on both resources and labors.

Operation	Description
<a href="#">Monitoring</a>	You can view the CPU and memory usage of Deployments, DaemonSets, and pods on the CCE console to determine the resource specifications you may need.
<a href="#">Rollback</a>	Only Deployments can be rolled back.
<a href="#">Pausing</a>	Only Deployments can be paused.
<a href="#">Resuming</a>	Only Deployments can be resumed.
<a href="#">Labeling</a>	Labels are key-value pairs and can be attached to workloads for affinity and anti-affinity scheduling.
<a href="#">Deletion</a>	You can delete a workload or job that is no longer needed. Deleted workloads or jobs cannot be recovered.
<a href="#">Access settings</a>	You can determine how your workloads can be accessed. For details, see <a href="#">Overview</a> .
<a href="#">Scheduling policies</a>	CCE supports custom and simple scheduling policies. <b>Custom scheduling policies</b> allow you to customize node affinity, workload affinity, and workload anti-affinity. <b>Simple scheduling policies</b> allow easy and convenient scheduling.
<a href="#">Event</a>	CCE provides event names, event types, number of occurrences, Kubernetes events, first occurrence time, and last occurrence time by workload or pod.

## Viewing Logs

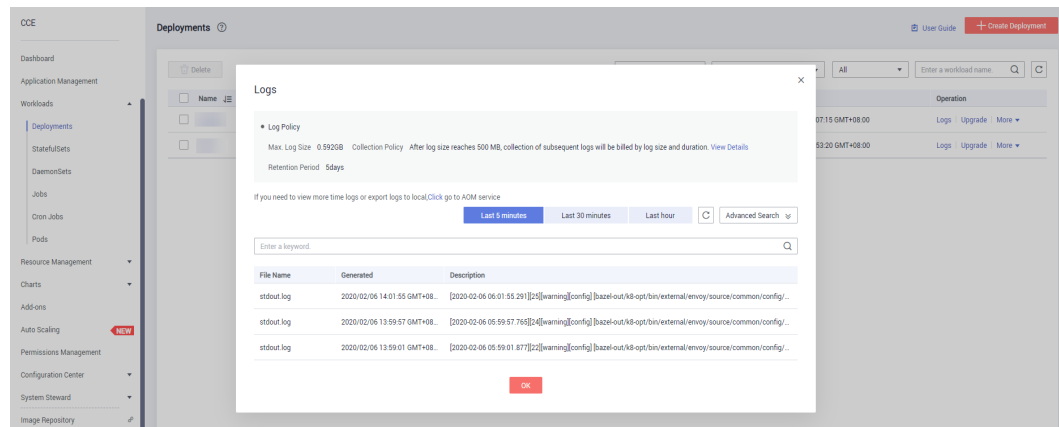
You can view logs of Deployments, StatefulSets, DaemonSets, and jobs. This section uses a Deployment as an example to describe how to view logs.

**Step 1** Log in to the CCE console. In the navigation pane, choose **Workloads > Deployments**.

**Step 2** In the same row as the workload you will view, click **Logs**.

In the displayed **Logs** window, view the logs generated in the last 5 minutes, 30 minutes, or 1 hour.

Figure 17-64 Viewing logs of a workload



----End

## Upgrading a Workload

You can replace images or image tags to quickly upgrade Deployments, StatefulSets, and DaemonSets without interrupting services.

This section uses a Deployment as an example to describe how to upgrade a workload.

Before replacing an image or image version, upload the new image to the SWR service.

**Step 1** Log in to the CCE console. In the navigation pane, choose **Workloads** > **Deployments**, and click **Upgrade** for the Deployment to be upgraded.

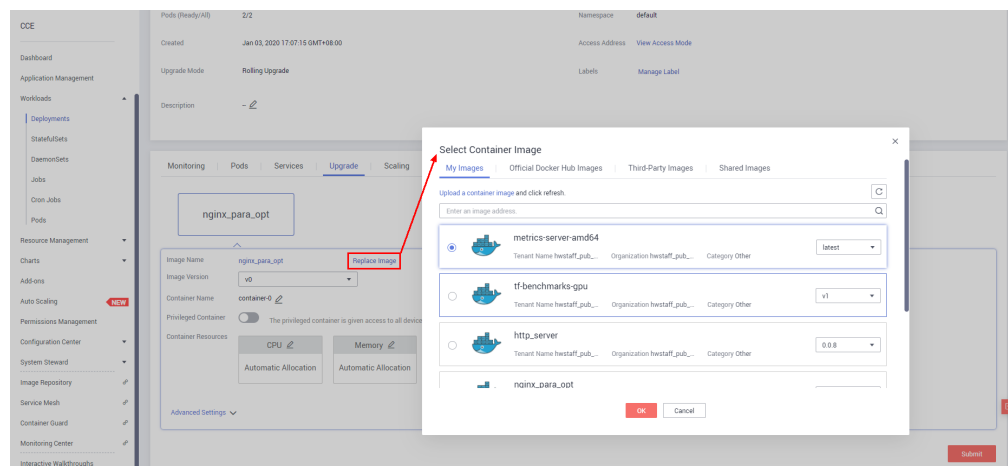
### NOTE

- Workloads cannot be upgraded in batches.
- Before performing an in-place StatefulSet upgrade, you must manually delete old pods. Otherwise, the upgrade status is always displayed as **Upgrading**.

**Step 2** Upgrade the Deployment.

- **Image Name:** To replace the Deployment image, click **Replace Image** and select a new image.

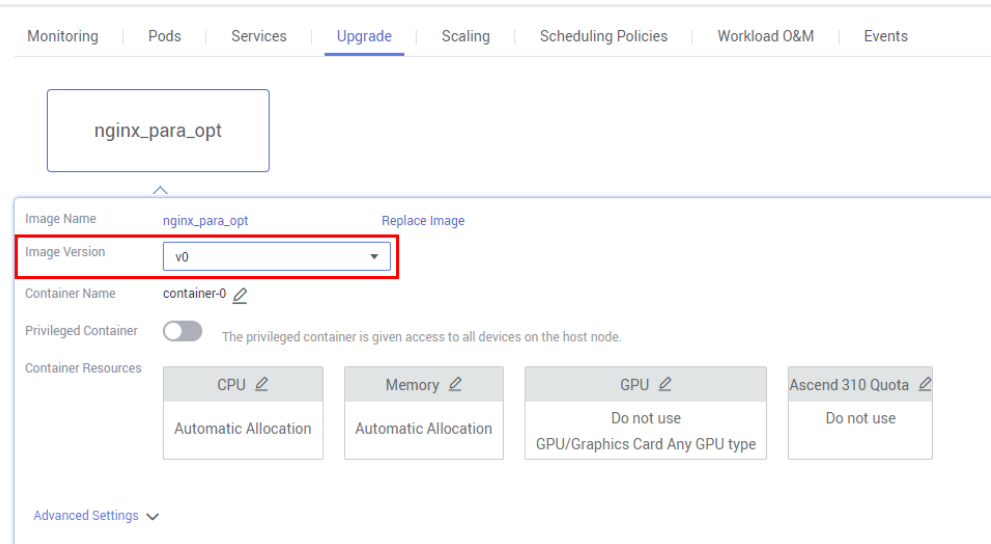
Figure 17-65 Replacing an image






- **Image Version:** To replace the Deployment image version, select a new version from the **Image Version** drop-down list.

**Figure 17-66** Replacing an image version



- **Container Name:** To change the container name, click  next to **Container Name** and enter a new name.
- **Privileged Container:** After this function is enabled, the container can access all devices on the host.
- **Container Resources:** You can set the CPU, memory and GPU quotas.
- **Advanced Settings:**
  - **Lifecycle:** Commands for starting and running containers can be set.
    - **Start Command:** executed when the workload is started. For details, see [Setting Container Startup Commands](#).
    - **Post-Start:** executed after the workload runs successfully. For more information, see [Setting Container Lifecycle Parameters](#).
    - **Pre-Stop:** executed to delete logs or temporary files before the workload ends. For more information, see [Setting Container Lifecycle Parameters](#).
  - **Health Check:** CCE provides two types of probes: liveness probe and readiness probe. They are used to determine whether containers and user services are running properly. For more information, see [Setting Health Check for a Container](#).
    - **Liveness Probe:** used to restart the unhealthy container.
    - **Readiness Probe:** used to change the container to the unready state when detecting that the container is unhealthy. In this way, service traffic will not be directed to the container.
  - **Environment Variables:** Environment variables can be added to a container. In general, environment variables are used to set parameters. On the **Environment Variables** tab page, click **Add Environment Variable**. Currently, three types of environment variables are supported:

- **Added manually:** Set **Variable Name** and **Variable Value/Reference**.
- **Added from Secret:** Set **Variable Name** and select the desired secret name and data. A secret must be created in advance. For details, see [Creating a Secret](#).
- **Added from ConfigMap:** Set **Variable Name** and select the desired ConfigMap name and data. A ConfigMap must be created in advance. For details, see [Creating a ConfigMap](#).

 **NOTE**

To edit an environment variable that has been set, click **Edit**. To delete an environment variable that has been set, click **Delete**.

- **Data Storage:** Data storage can be mounted to containers for persistent storage and high disk I/O. Local disks and cloud storage volumes are supported. For details, see [Storage \(CSI\)](#).

 **NOTE**

You can add data storage volumes only when creating a StatefulSet.

- **Security Context:** Container permissions can be configured to protect CCE and other containers from being affected.  
Enter the user ID to set container permissions and prevent systems and other containers from being affected.
- **Log Policies:** Log collection policies and log directory can be configured to collect container logs for unified management and analysis. For details, see [Container Logs](#).

**Step 3** Click **Submit**.

----End

## Editing a YAML file

You can modify and download the YAML files of Deployments, StatefulSets, DaemonSets, and pods on the CCE console. YAML files of jobs and cron jobs can only be viewed, copied, and downloaded. This section uses a Deployment as an example to describe how to edit the YAML file.

**Step 1** Log in to the CCE console. In the navigation pane, choose **Workloads > Deployments**.

**Step 2** In the same row as the workload you will edit, choose **Operation > More > Edit YAML**. In the **Edit YAML** window, edit the YAML file of the current workload.

**Step 3** Click **Edit** and then **OK** to save the changes.

**Step 4** (Optional) In the **Edit YAML** window, click **Download** to download the YAML file.

----End

## Scaling a Workload

A workload can be automatically resized according to custom scaling policies, freeing you from the efforts to manually adjust the amount of resources for

fluctuating service traffic. This saves you big on both resources and labors. This section uses a Deployment as an example to describe how to scale a workload.


- Step 1** Log in to the CCE console. In the navigation pane, choose **Workloads > Deployments**.
- Step 2** In the same row as the workload for which you will add a scaling policy, choose **Operation > More > Scaling**.
- Step 3** On the **Scaling** tab page, add or edit scaling policies. Scaling policies are classified as auto and manual scaling policies.

For details, see [Scaling a Workload](#).

----End

## Monitoring a Workload

You can view the CPU and memory usage of Deployments, DaemonSets, and pods on the CCE console to determine the resource specifications you may need. This section uses a Deployment as an example to describe how to monitor a workload.

- Step 1** Log in to the CCE console. In the navigation pane, choose **Workloads > Deployments**.
- Step 2** Click the name of the Deployment to be monitored. On the displayed Deployment details page, click the **Monitoring** tab to view CPU usage and memory usage of the Deployment.
- Step 3** Click the **Pods** tab. Click  next to a pod to be monitored and click **Monitoring**.
- Step 4** Check CPU usage and memory usage of the pod.

- CPU usage

The horizontal axis indicates time while the vertical axis indicates the CPU usage. The green line indicates the CPU usage while the red line indicates the CPU usage limit.

### NOTE

It takes some time to calculate CPU usage. Therefore, when CPU and memory usage are displayed for the first time, CPU usage is displayed about one minute later than memory usage.

CPU and memory usage are displayed only for pods in the running state.

- Memory usage

The horizontal axis indicates time while the vertical axis indicates the memory usage. The green line indicates the memory usage while the red line indicates the memory usage limit.

### NOTE

Memory usage is displayed only for a running pod.

----End

## Rolling Back a Workload (Available Only for Deployments)

CCE records the release history of all Deployments. You can roll back a Deployment to a specified version.

- Step 1** Log in to the CCE console. In the navigation pane, choose **Workloads > Deployments**.
- Step 2** In the same row as the Deployment you will roll back, choose **Operation > More > Roll Back**.
- Step 3** In the **Roll Back to This Version** drop-down list, select the version to which you will roll back the Deployment. Then, click **OK**.

----End

## Pausing a Workload (Available Only for Deployments)

You can pause Deployments. After a Deployment is paused, the upgrade command can be successfully issued but will not be applied to the pods.

If you are performing a rolling upgrade, the rolling upgrade stops after the pause command is issued. In this case, the new and old pods coexist.

- Step 1** Log in to the CCE console. In the navigation pane, choose **Workloads > Deployments**.
- Step 2** In the same row as the Deployment you will pause, choose **Operation > More > Pause**.
- Step 3** In the displayed **Pause Workload** dialog box, click **OK**.
- Step 4** Click **OK**.

---

**NOTICE**

Deployments in the paused state cannot be rolled back.

---

----End

## Resuming a Workload (Available Only for Deployments)

You can resume paused Deployments. After a Deployment is resumed, it can be upgraded or rolled back. Its pods will inherit the latest updates of the Deployment. If they are inconsistent, the pods are upgraded automatically according to the latest information of the Deployment.

- Step 1** Log in to the CCE console. In the navigation pane, choose **Workloads > Deployments**.
- Step 2** In the same row as the Deployment you will resume, choose **Operation > More > Resume**.
- Step 3** In the displayed **Resume Workload** dialog box, click **OK**.

----End

## Managing Labels

Labels are key-value pairs and can be attached to workloads. Workload labels are often used for affinity and anti-affinity scheduling. You can add labels to multiple workloads or a specified workload.

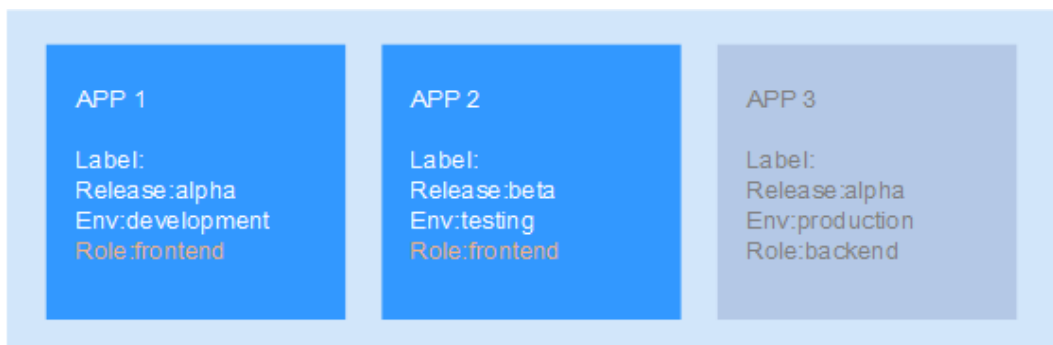
You can manage the labels of Deployments, StatefulSets, and DaemonSets based on service requirements. This section uses Deployments as an example to describe how to manage labels.

In the following figure, three labels (release, env, and role) are defined for workload APP 1, APP 2, and APP 3. The values of these labels vary with workload.

- Label of APP 1: [release:alpha;env:development;role:frontend]
- Label of APP 2: [release:beta;env:testing;role:frontend]
- Label of APP 3: [release:alpha;env:production;role:backend]

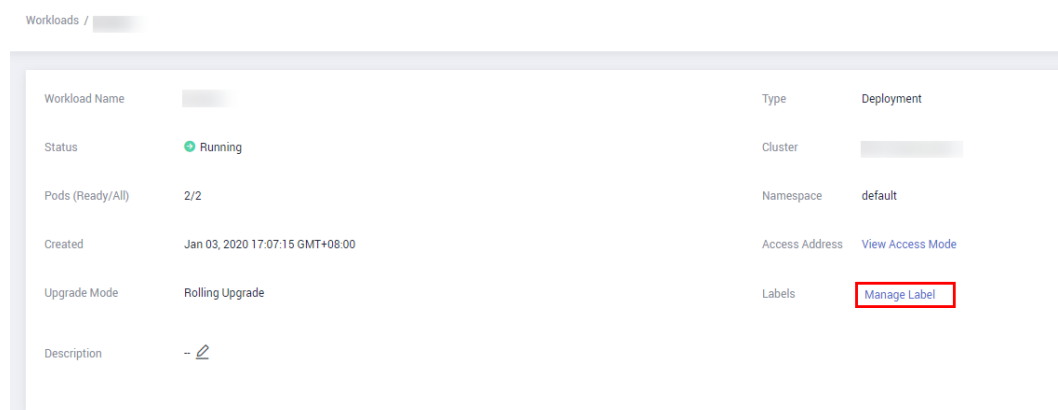
If you set **key** to **role** and **value** to **frontend** when using workload scheduling or another function, APP 1 and APP 2 will be selected.

Figure 17-67 Label example



- Step 1** Log in to the CCE console. In the navigation pane, choose **Workloads > Deployments**.
- Step 2** Click the name of the workload whose labels will be managed.
- Step 3** On the workload details page, click **Manage Label**. In the displayed dialog box, click **Add Label**. Enter the label key and value, and click **OK**.

Figure 17-68 Managing labels



 **NOTE**

A key-value pair must contain 1 to 63 characters starting and ending with a letter or digit. Only letters, digits, hyphens (-), underscores (\_), and periods (.) are allowed.

----End

## Deleting a Workload/Job

You can delete a workload or job that is no longer needed. Deleted workloads or jobs cannot be recovered. Exercise caution when you perform this operation. This section uses a Deployment as an example to describe how to delete a workload.

**Step 1** Log in to the CCE console. In the navigation pane, choose **Workloads > Deployments**.

**Step 2** In the same row as the workload you will delete, choose **Operation > More > Delete**.

Read the system prompts carefully. A workload cannot be recovered after it is deleted. Exercise caution when performing this operation.

**Step 3** Click **Yes**.

 **NOTE**

- If the node where the pod is located is unavailable or shut down and the workload cannot be deleted, you can forcibly delete the pod from the pod list on the workload details page.
- Ensure that the storage volumes to be deleted are not used by other workloads. If these volumes are imported or have snapshots, you can only unbind them.

----End

## Events

On the workload details page, click the **Events** or **Pods** tab to view the events, event types, number of occurrences, Kubernetes events, first occurrence time, and last occurrence time.

 **NOTE**

Event data will be retained for one hour and then automatically deleted.

### 17.6.11 Scaling a Workload

After scaling policies are defined, pods can be automatically added or deleted based on resource changes, fixed time, and fixed periods. You do not need to manually adjust resource allocation to cope with service changes and data traffic spikes.

- **Auto scaling:** You can set metric-based, scheduled, and periodic policies. After configuration, pods can be automatically added or deleted based on resource changes or the specified schedule.
- **Manual scaling:** Pods are immediately added or deleted after the configuration is complete.

 NOTE

**Scaling policy priority:** If you do not manually adjust the number of pods, auto scaling policies will take effect for resource scheduling. If **manual scaling** is triggered, auto scaling policies will be temporarily invalid.

## Auto Scaling - HPA/CustomedHPA

HPA or CustomedHPA policies can be used for auto scaling. **You can view all policies or perform more operations in [Auto Scaling](#).**

## Auto Scaling - AOM

You can define auto scaling policies as required, which can intelligently adjust resources in response to service changes and data traffic spikes.

Auto scaling can be backed by Application Operations Management (AOM), but not for clusters of v1.17 and later.

Currently, CCE supports the following types of auto scaling policies:

**Metric-based policy:** After a workload is created, pods will be automatically scaled when the workload's CPU or memory usage exceeds or falls below a preset limit.

**Scheduled policy:** scaling at a specified time. Scheduled auto scaling is applicable flash sales, premier shopping events, and other regular events that bring a high burst of traffic load.

**Periodic policy:** scaling at a specified time on a daily, weekly, or monthly basis. Periodic scheduling is applicable to scenarios where traffic changes periodically.

- **Metric-based policy:** Supports auto scaling of a workload based on the CPU/memory usage.
  - a. Log in to the CCE console. In the navigation pane, choose **Workloads > Deployments** or **StatefulSets**. In the same row as the target workload, choose **More > Scaling**.
  - b. In the **Auto Scaling** area, click **Add Scaling Policy**.
  - c. Set the policy parameters as listed in [Table 17-50](#).

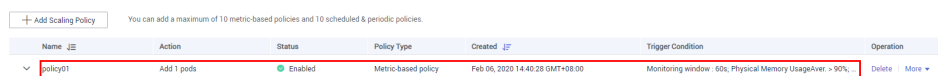
**Table 17-50** Parameters for adding a metric-based policy

Parameter	Description
Policy Name	Enter the name of the scaling policy. The policy name must be 1 to 64 characters in length and start with a letter. Only letters, digits, underscores (_), and hyphens (-) are allowed.

Parameter	Description
Policy Type	Set this parameter to <b>Metric-based policy</b> . The alarm policy is triggered based on historical data. The system checks whether the indicators set by the user in the monitoring window meet the triggering conditions <b>every minute</b> . If the triggering conditions are met for N consecutive periods, the system performs the action specified by the policy.
Metric	Set the metrics that describe the resource performance data or status. <ul style="list-style-type: none"> <li>• <b>CPU Usage</b>: CPU usage of the measured object. The value is the percentage of the used CPU cores to the total CPU cores.</li> <li>• <b>Physical Memory Usage</b>: percentage of the physical memory size used by the measured object to the physical memory size that the measured object has applied for.</li> </ul>
Trigger Condition	The value can be higher (>) or lower (<) than a threshold. When the usage of the preceding metrics reaches the specified value, the scaling policy is triggered.  For example, if <b>Metric</b> is set to <b>CPU Usage</b> and this parameter is set to <b>&gt; 70%</b> , the scaling policy is triggered when the CPU usage exceeds 70%.
Monitoring window	Size of the data aggregation window. If the value is set to <b>60</b> , metric statistics are collected every 60 seconds.
Threshold Crossings	Number of consecutive times that the threshold is reached within the monitoring window. The calculation cycle is fixed at one minute.  If the parameter is set to <b>3</b> , the action is triggered if threshold is reached for three consecutive measurement periods.
Action	Action executed after a policy is triggered. Two actions are available: add or reduce pods.

- d. Click **OK**.
- e. In the **Auto Scaling** area, check that the policy has been started.

**Figure 17-69** Viewing a metric-based policy

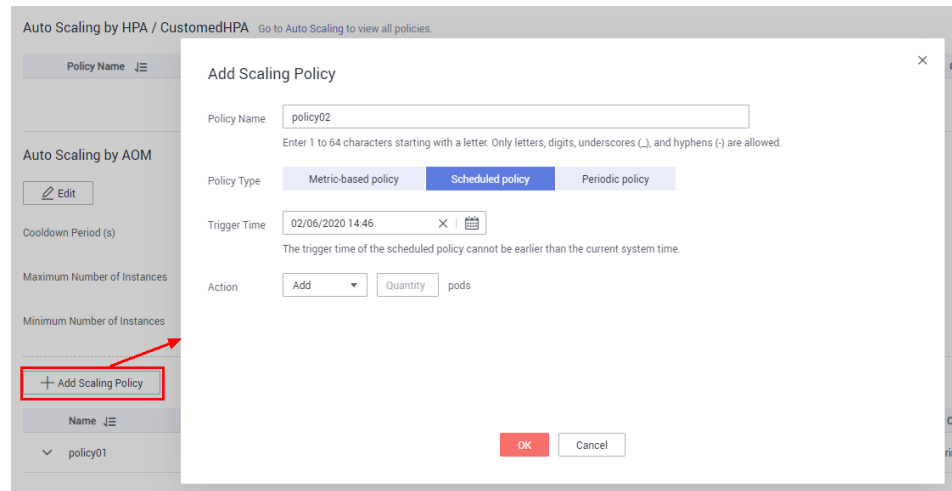




When the trigger condition is met, the auto scaling policy starts automatically.

- **Scheduled policy:** scaling at a specified time.
  - a. In the **Auto Scaling** area, click **Add Scaling Policy**. Select **Scheduled policy**.

**Figure 17-70** Scheduled Policy



**Table 17-51** Parameters for adding a scheduled policy

Parameter	Description
Policy Name	Enter the name of the scaling policy. The policy name must be 1 to 64 characters in length and start with a letter. Only letters, digits, underscores (_), and hyphens (-) are allowed.
Policy Type	Set this parameter to <b>Scheduled policy</b> .
Trigger Time	Time at which the policy is enforced.
Action	Action executed after a policy is triggered. Three actions are available: add pods, reduce pods, and set the number of pods.

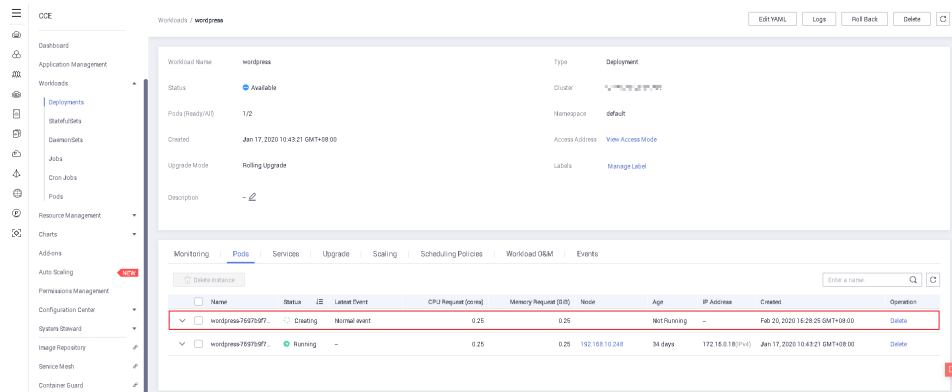
- b. Click **OK**.
- c. In the **Auto Scaling** area, check that the policy has been started.

**Figure 17-71** Viewing a scheduled policy



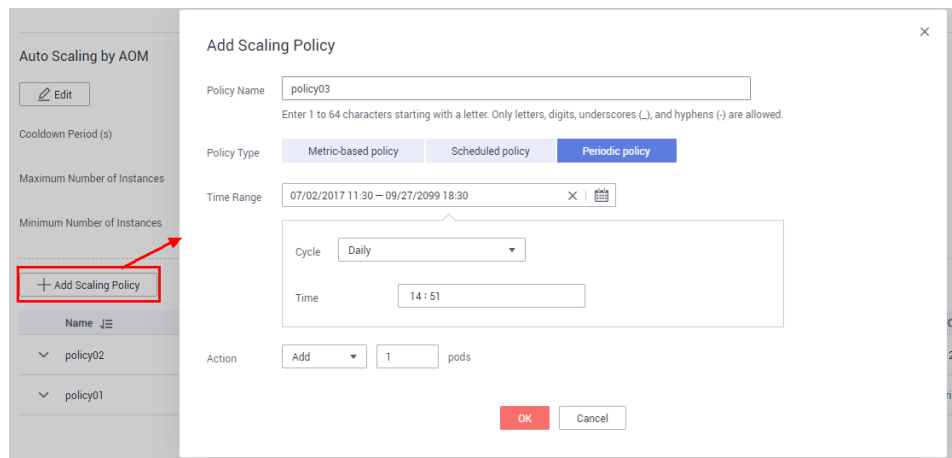
When the trigger time is reached, you can see on the **Pods** tab page that the auto scaling policy has taken effect.

**Figure 17-72** Auto scaling having taken effect



- **Periodic policy:** scaling at a specified time on a daily, weekly, or monthly basis.
  - a. In the **Auto Scaling** area, click **Add Scaling Policy**. Select **Periodic policy**.

**Figure 17-73** Periodic policy



**Table 17-52** Parameters for adding a periodic policy

Parameter	Description
Policy Name	Enter the name of the scaling policy. The policy name must be 1 to 64 characters in length and start with a letter. Only letters, digits, underscores (_), and hyphens (-) are allowed.
Policy Type	Set this parameter to <b>Periodic policy</b> .
Time Range	Specify the time for triggering the policy.
Action	Action executed after a policy is triggered. Three actions are available: add pods, reduce pods, and set the number of pods.

- b. Click **OK**.

- c. In the **Auto Scaling** area, check that the policy has been started.

**Figure 17-74** Viewing a periodic policy

Name	Action	Status	Policy Type	Created	Trigger Condition	Operation
policy03	Add 1 pods	Enabled	Periodic policy	Feb 06, 2020 14:52:43 GMT+08:00	Jul 02, 2017 11:30 To Sep 27, 2099 18:30 Daily 14:51	Delete

When the trigger condition is met, the auto scaling policy starts automatically.

## Manual Scaling

**Step 1** Log in to the CCE console. In the navigation pane, choose **Workloads > Deployments** or **StatefulSets**. In the same row as the target workload, choose **More > Scaling**.

**Step 2** In the **Manual Scaling** area, click and change the number of pods to, for example, **3**. Then, click **Save**. The scaling takes effect immediately.

**Figure 17-75** Changing the number of pods

Manual Scaling

Pods  Save Cancel

**Step 3** On the **Pods** tab page, check that a new pod is being created. When the pod status becomes **Running**, pod scaling is complete.

**Figure 17-76** Manual scaling

Name	Status	Latest Event	CPU Request (cores)	Memory Request (GiB)	Node	Age	IP Address	Created	Operation
	Running		--	--	192.168.50.222	33 days	172.17.2.12	Jan 03, 2020 17:07:15 GMT+08:00	Delete
	Creating	InstanceVolume mou...	--	--	192.168.50.149	Not Running	--	Feb 06, 2020 15:04:55 GMT+08:00	Delete
	Running		--	--	192.168.50.34	33 days	172.17.1.29	Jan 03, 2020 17:07:15 GMT+08:00	Delete

----End

## 17.6.12 Configuring a Container

### 17.6.12.1 Using a Third-Party Image

#### Scenario

CCE allows you to create workloads using images pulled from third-party image repositories.

Generally, a third-party image repository can be accessed only after authentication (using your account and password). CCE uses the secret-based authentication to

pull images. Therefore, you need to create a secret for an image repository before pulling images from the repository.

## Prerequisites

The node where the workload is running is accessible from public networks. You can access public networks through [LoadBalancer](#).

## Using the Console

**Step 1** Create a secret for accessing a third-party image repository.

In the navigation pane, choose **Configuration Center > Secret**, and click **Create Secret**. **Type** must be set to **kubernetes.io/dockerconfigjson**. For details, see [Creating a Secret](#).

Enter the user name and password used to access the third-party image repository.

**Step 2** Create a workload. For details, see [Creating a Deployment](#) or [Creating a StatefulSet](#). If the workload will be created from a third-party image, set the image parameters as follows:

1. Set **Secret Authentication** to **Yes**.
2. Select the secret created in step [Step 1](#).
3. Enter the image address.

**Step 3** Click **Create**.

----End

## Using kubectl

**Step 1** Use kubectl to connect to the cluster. For details, see [Connecting to a Cluster Using kubectl](#).

**Step 2** Create a secret of the dockercfg type using kubectl.

```
kubectl create secret docker-registry myregistrykey --docker-server=DOCKER_REGISTRY_SERVER --docker-username=DOCKER_USER --docker-password=DOCKER_PASSWORD --docker-email=DOCKER_EMAIL
```

In the preceding commands, **myregistrykey** indicates the secret name, and other parameters are described as follows:

- **DOCKER\_REGISTRY\_SERVER**: address of a third-party image repository, for example, **www.3rdregistry.com** or **10.10.10.10:443**
- **DOCKER\_USER**: account used for logging in to a third-party image repository
- **DOCKER\_PASSWORD**: password used for logging in to a third-party image repository
- **DOCKER\_EMAIL**: email of a third-party image repository

**Step 3** Use a third-party image to create a workload.

A dockercfg secret is used for authentication when you obtain a private image. The following is an example of using the myregistrykey for authentication.

```
apiVersion: v1  
kind: Pod
```

```
metadata:
  name: foo
  namespace: default
spec:
  containers:
  - name: foo
    image: www.3rdregistry.com/janedoe/awesomeapp:v1
    imagePullSecrets:
    - name: myregistrykey          #Use the created secret.
```

----End

## 17.6.12.2 Setting Container Specifications

### Scenario

CCE allows you to set resource limits for added containers during workload creation. You can request and limit the CPU and memory quotas used by each pod in the workload.

### Meanings

For **CPU** and **Memory**, the meanings of **Request** and **Limit** are as follows:

- If **Request** is selected, the system schedules the pod to the node that meets the requirements for workload deployment based on the request value.
- If **Request** is deselected, the system schedules the pod to a random node for workload deployment.
- If **Limit** is selected, the system limits the resources used by the workload based on the preset value.
- If **Limit** is deselected, the system does not limit the resources used by the pod. If the memory resources used by the pod exceed the memory allocated to the node, the workload or node may be unavailable.

#### NOTE

When creating a workload, you are advised to set the upper and lower limits of CPU and memory resources. If the upper and lower resource limits are not set for a workload, a resource leak of this workload will make resources unavailable for other workloads deployed on the same node. In addition, workloads that do not have upper and lower resource limits cannot be accurately monitored.

For **GPU** quotas, the meanings of **Use** and **Any GPU type** are as follows:

- If **Use** is selected, the system schedules the pod to a node that meets the requirements for workload deployment based on the configured value.
- **Any GPU type** is selected by default and cannot be deselected. This option indicates that the resources used by pods are not limited.

### Configuration Description

- CPU quotas:

**Table 17-53** Description of CPU quotas

Parameter	Description
CPU request	Minimum number of CPU cores required by a container. Resources are scheduled for the container based on this value. The container can be scheduled to this node only when the total available CPU on the node is greater than or equal to the number of containerized CPU applications.
CPU limit	Maximum number of CPU cores available for a container.

### Recommended configuration

Actual available CPU of a node  $\geq$  Sum of CPU limits of all containers on the current node  $\geq$  Sum of CPU requests of all containers on the current node. You can view the actual available CPUs of a node on the CCE console (**Resource Management > Nodes > Allocatable**).

- Memory quotas:

**Table 17-54** Description of memory quotas

Parameter	Description
Memory request	Minimum amount of memory required by a container. Resources are scheduled for the container based on this value. The container can be scheduled to this node only when the total available memory on the node is greater than or equal to the number of containerized memory applications.
Memory Limit	Maximum amount of memory available for a container. When the memory usage exceeds the configured memory limit, the instance may be restarted, which affects the normal use of the workload.

### Recommended configuration

Actual available memory of a node  $\geq$  Sum of memory limits of all containers on the current node  $\geq$  Sum of memory requests of all containers on the current node. You can view the actual available memory of a node on the CCE console (**Resource Management > Nodes > Allocatable**).

#### NOTE

The allocatable resources are calculated based on the resource request value (**Request**), which indicates the upper limit of resources that can be requested by pods on this node, but does not indicate the actual available resources of the node. The calculation formula is as follows:

- Allocatable CPU = Total CPU – Requested CPU of all pods – Reserved CPU for other resources
- Allocatable memory = Total memory – Requested memory of all pods – Reserved memory for other resources

## Example

Assume that a cluster contains a node with 4 cores and 8 GB. A workload containing two pods has been deployed on the cluster. The resources of the two pods (pods 1 and 2) are as follows: {CPU request, CPU limit, memory request, memory limit} = {1 core, 2 cores, 2 GB, 2 GB}.

The CPU and memory usage of the node is as follows:

- Allocatable CPU = 4 cores - (1 core requested by pod 1 + 1 core requested by pod 2) = 2 cores
- Allocatable memory = 8 GB - (2 GB requested by pod 1 + 2 GB requested by pod 2) = 4 GB

Therefore, the remaining 2 cores and 4 GB can be used by the next new pod.

### 17.6.12.3 Setting Container Lifecycle Parameters

#### Scenario

CCE provides callback functions for the lifecycle management of containerized applications. For example, if you want a container to perform a certain operation before stopping, you can register a hook function.

CCE provides the following lifecycle callback functions:

- **Start Command:** executed to start a container. For details, see [Setting Container Startup Commands](#).
- **Post-Start:** executed immediately after a container is started. For details, see [Post-Start Processing](#).
- **Pre-Stop:** executed before a container is stopped. The pre-stop processing function helps you ensure that the services running on the pods can be completed in advance in the case of pod upgrade or deletion. For details, see [Pre-Stop Processing](#).

#### Commands and Parameters Used to Run a Container

A Docker image has metadata that stores image information. If lifecycle commands and arguments are not set, CCE runs the default commands and arguments, that is, Docker instructions **ENTRYPOINT** and **CMD**, provided during image creation.

If the commands and arguments used to run a container are set during application creation, the default commands **ENTRYPOINT** and **CMD** are overwritten during image build. The rules are as follows:

**Table 17-55** Commands and parameters used to run a container

Image Entrypoint	Image CMD	Command to Run a Container	Parameters to Run a Container	Command Executed
[touch]	[/root/test]	Not set	Not set	[touch /root/test]

Image Entrypoint	Image CMD	Command to Run a Container	Parameters to Run a Container	Command Executed
[touch]	[/root/test]	[mkdir]	Not set	[mkdir]
[touch]	[/root/test]	Not set	[/opt/test]	[touch /opt/test]
[touch]	[/root/test]	[mkdir]	[/opt/test]	[mkdir /opt/test]

## Startup Commands

By default, the default command during image start. To run a specific command or rewrite the default image value, you must perform specific settings: For details, see [Setting Container Startup Commands](#).

## Post-Start Processing

- Step 1** Log in to the CCE console. Expand **Lifecycle** when adding a container during workload creation.
- Step 2** Set the post-start processing parameters, as listed in [Table 17-56](#).

**Table 17-56** Post-start processing parameters

Parameter	Description
CLI	<p>Set commands to be executed in the container for post-start processing. The command format is <b>Command Args[1] Args[2]...</b> <b>Command</b> is a system command or a user-defined executable program. If no path is specified, an executable program in the default path will be selected. If multiple commands need to be executed, write the commands into a script for execution. <b>Commands that are executed in the background or asynchronously are not supported.</b></p> <p>Example command:</p> <pre>exec: command: - /install.sh - install_agent</pre> <p>Enter <b>/install install_agent</b> in the script. This command indicates that <b>install.sh</b> will be executed after the container is created successfully.</p>



Parameter	Description
HTTP request	<p>Send an HTTP request for post-start processing. The related parameters are described as follows:</p> <ul style="list-style-type: none"> <li>• <b>Path:</b> (optional) request URL.</li> <li>• <b>Port:</b> (mandatory) request port.</li> <li>• <b>Host Address:</b> (optional) IP address of the request. The default value is the IP address of the node where the container resides.</li> </ul>

----End

## Pre-Stop Processing

**Step 1** When creating a workload and adding a container, expand **Lifecycle**.

**Step 2** Set **pre-stop** parameters, as shown in [Table 17-56](#).

**Table 17-57** Pre-stop parameters

Parameter	Description
CLI	<p>Set commands to be executed in the container for pre-stop processing. The command format is <b>Command Args[1] Args[2]...</b> <b>Command</b> is a system command or a user-defined executable program. If no path is specified, an executable program in the default path will be selected. If multiple commands need to be executed, write the commands into a script for execution.</p> <p>Example command:</p> <pre>exec: command: - /uninstall.sh - uninstall_agent</pre> <p>Enter <b>/uninstall uninstall_agent</b> in the script. This command indicates that the <b>uninstall.sh</b> script will be executed before the container completes its execution and stops running.</p>
HTTP request	<p>Send an HTTP request for pre-stop processing. The related parameters are described as follows:</p> <ul style="list-style-type: none"> <li>• <b>Path:</b> (optional) request URL.</li> <li>• <b>Port:</b> (mandatory) request port.</li> <li>• <b>Host Address:</b> (optional) IP address of the request. The default value is the IP address of the node where the container resides.</li> </ul>

----End

## Container Restart Policy

The **restartPolicy** field is used to specify the pod restart policy. The restart policy type can be **Always**, **OnFailure**, or **Never**. The default value is **Always**.

When **restartPolicy** is used, containers are restarted only through kubelet on the same node.

Restart Policy	Description
Always	When a container fails, kubelet automatically restarts the container.
OnFailure	When the container stops running and the exit code is not 0, kubelet automatically restarts the container.
Never	kubelet does not restart the container regardless of the container running status.

### NOTE

Controllers that can manage pods include ReplicaSet Controllers, jobs, DaemonSets, and kubelet (static pod).

- ReplicaSet Controller and DaemonSet: The policy must be set to **Always** to ensure that containers run continuously.
- Job: The policy can be set to **OnFailure** or **Never** to ensure that containers are not restarted after being executed.
- kubelet will restart a pod whenever it fails, regardless of the value of **restartPolicy**. In addition, no health check is performed on the pod.

## Example YAML for Setting the Container Lifecycle

This section uses Nginx as an example to describe how to set the container lifecycle.

**Step 1** Use `kubectl` to connect to the cluster. For details, see [Connecting to a Cluster Using kubectl](#).

**Step 2** Create and edit the `nginx-deployment.yaml` file. `nginx-deployment.yaml` is an example file name, and you can change it as required.

### **vi nginx-deployment.yaml**

In the following configuration file, the **postStart** command is defined to run the **install.sh** command in the `/bin/bash` directory. **preStop** is defined to run the **uninstall.sh** command.

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx
spec:
  replicas: 1
  selector:
    matchLabels:
      app: nginx
  strategy:
```

```

type: RollingUpdate
template:
  metadata:
    labels:
      app: nginx
  spec:
    restartPolicy: Always          #Restart policy
    containers:
      - image: nginx
        command:
          - sleep 3600            #Startup command
        imagePullPolicy: Always
        lifecycle:
          postStart:
            exec:
              command:
                - /bin/bash
                - install.sh      #Post-start command
          preStop:
            exec:
              command:
                - /bin/bash
                - uninstall.sh    #Pre-stop command
    name: nginx
    imagePullSecrets:
      - name: default-secret

```

----End

## 17.6.12.4 Setting Container Startup Commands

### Scenario

When creating a workload or job, you can use an image to specify the processes running in the container.

By default, the image runs the default command. To run a specific command or rewrite the default image value, you must perform the following settings:

- **Working directory:** working directory of the command.  
If the working directory is not specified in the image or on the console, the default value is `/`.
- **Command:** command that controls the running of an image.
- **Args:** parameters transferred to the running command.

---

#### NOTICE

After a container is started, do not modify configurations in the container. If configurations in the container are modified (for example, passwords, certificates, and environment variables of a containerized application are added to the container), the configurations will be lost after the container restarts and container services will become abnormal. An example scenario of container restart is pod rescheduling due to node anomalies.

Configurations must be imported to a container as arguments. Otherwise, configurations will be lost after the container restarts.

---

## Commands and Arguments Used to Run a Container

A Docker image has metadata that stores image information. If lifecycle commands and arguments are not set, CCE runs the default commands and arguments, that is, Docker instructions **ENTRYPOINT** and **CMD**, provided during image creation.

If the commands and arguments used to run a container are set during application creation, the default commands **ENTRYPOINT** and **CMD** are overwritten during image build. The rules are as follows:

**Table 17-58** Commands and parameters used to run a container

Image Entrypoint	Image CMD	Command to Run a Container	Args to Run a Container	Command Executed
[touch]	[/root/test]	Not set	Not set	[touch /root/test]
[touch]	[/root/test]	[mkdir]	Not set	[mkdir]
[touch]	[/root/test]	Not set	[/opt/test]	[touch /opt/test]
[touch]	[/root/test]	[mkdir]	[/opt/test]	[mkdir /opt/test]

## Setting the Startup Command

**Step 1** Log in to the CCE console. Expand **Lifecycle** when adding a container during workload or job creation.

**Step 2** Enter the running command and parameters, as shown in [Table 17-59](#).

 **NOTE**

- The current startup command is provided as a string array and corresponds to the Entrypoint startup command of Docker. The format is as follows: ["executable", "param1", "param2",,..]. For details about how to start Kubernetes containers, click [here](#).
- The lifecycle of a container is the same as that of the startup command. That is, the lifecycle of the container ends after the command is executed.

**Table 17-59** Container startup command

Configuration Item	Procedure
Command	<p>Enter an executable command, for example, <b>/run/server</b>.</p> <p>If there are multiple commands, separate them with spaces. If the command contains a space, you need to add a quotation mark ("").</p> <p><b>NOTE</b> If there are multiple commands, you are advised to run the <b>/bin/sh</b> or other shell commands. Other commands are used as parameters.</p>
Args	<p>Enter the argument that controls the container running command, for example, <b>--port=8080</b>.</p> <p>If there are multiple arguments, separate them in different lines.</p>

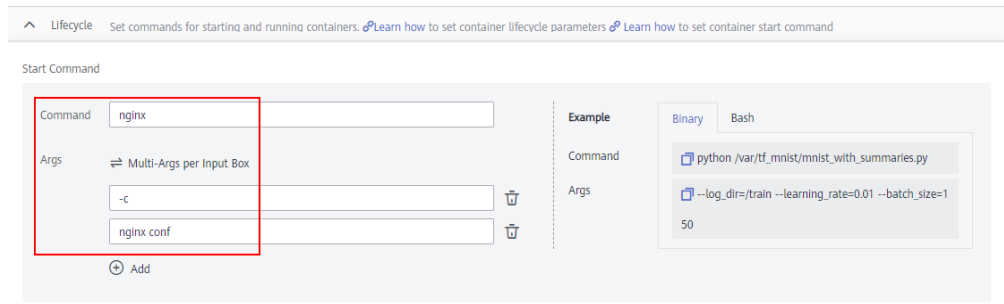
The following uses Nginx as an example to describe three typical application scenarios of the container startup command:

Example code:

```
nginx -c nginx.conf
```

- Scenario 1: Both the **command** and **arguments** are set.

**Figure 17-77** Setting the startup command and arguments



Example YAML file:

```
command:
  - nginx
args:
  - '-c'
  - nginx.conf
```

- Scenario 2: Only the **command** is set.

**NOTE**

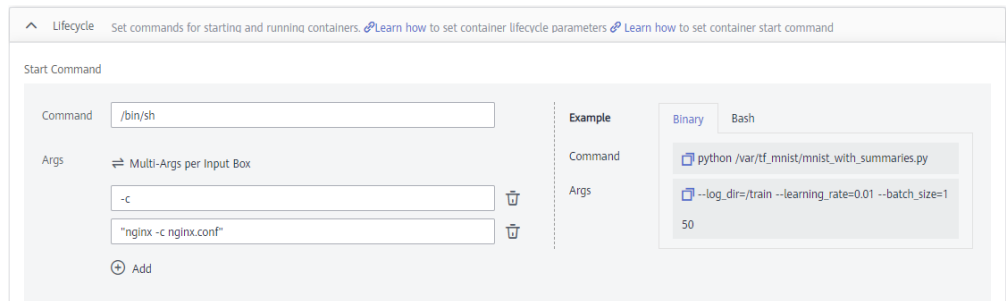
A command must be enclosed in double quotes. If no double quotes are added, the command is split into multiple commands based on space character.

Example YAML file:

```
command:
  - nginx -c nginx.conf
args:
```

- Scenario 3: Only **arguments** are set.

**Figure 17-78** Setting startup arguments



**NOTE**

If the container startup command is not added to the system path, run the **/bin/sh** command to execute the container startup command. The container startup command must be enclosed in double quotes.

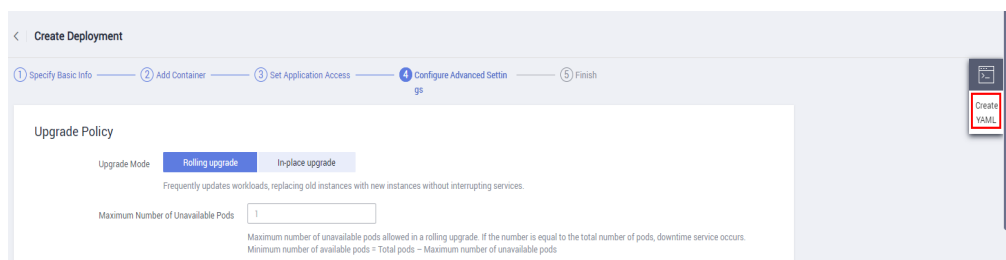
Example YAML file:

```
command:
  - /bin/sh
args:
  - '-c'
  - "'nginx -c nginx.conf'"
```

**Step 3** Check or modify the YAML file.

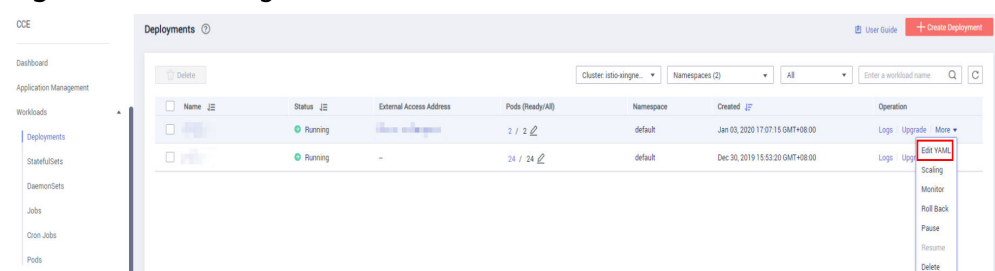
- When creating a workload, in the **Configure Advanced Settings** step, click **YAML** on the right.

**Figure 17-79** Checking or editing a YAML file



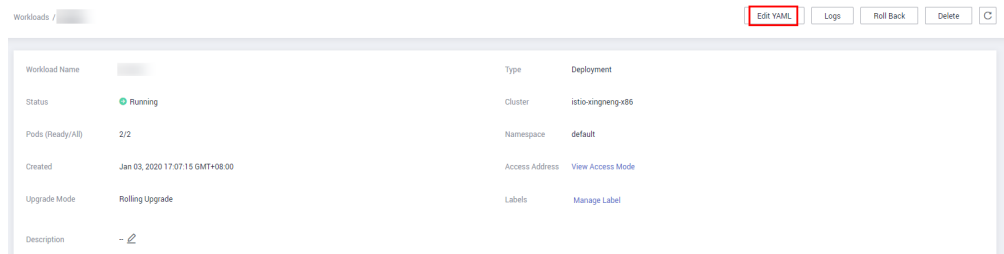
- After the workload is created, go to the workload list. In the same row as the workload, choose **More > Edit YAML**.

**Figure 17-80** Editing a YAML file



- After the workload is created, go to the workload details page. On the displayed page, click **Edit YAML** in the upper right corner.

**Figure 17-81** Editing a YAML file



----End

## Example YAML for Setting Container Startup Commands

This section uses Nginx as an example to describe how to set container startup commands using kubectl.

Use kubectl to connect to the cluster. For details, see [Connecting to a Cluster Using kubectl](#). See [Using kubectl to create a Deployment](#) or [Using kubectl to create a StatefulSet](#). For more details on how to set container startup commands, see [official Kubernetes documentation](#).

```
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
          command:
            - sleep
            - '3600'                #Startup command
          imagePullPolicy: Always
      lifecycle:
        postStart:
          exec:
            command:
              - /bin/bash
              - install.sh          #Post-start command
        preStop:
          exec:
            command:
              - /bin/bash
              - uninstall.sh      #Pre-stop command
      name: nginx
      imagePullSecrets:
        - name: default-secret
```

## 17.6.12.5 Setting Health Check for a Container

### Scenario

Health check regularly checks the health status of containers during container running. If the health check function is not configured, a pod cannot detect service exceptions or automatically restart the service to restore it. This will result in a situation where the pod status is normal but the service in the pod is abnormal.

CCE provides the following health check probes:

- **Liveness probe:** checks whether a container is still alive. It is similar to the **ps** command that checks whether a process exists. If the liveness check of a container fails, the cluster restarts the container. If the liveness check is successful, no operation is executed.
- **Readiness probe:** checks whether a container is ready to process user requests. Upon that the container is detected unready, service traffic will not be directed to the container. It may take a long time for some applications to start up before they can provide services. This is because that they need to load disk data or rely on startup of an external module. In this case, the application process is running, but the application cannot provide services. To address this issue, this health check probe is used. If the container readiness check fails, the cluster masks all requests sent to the container. If the container readiness check is successful, the container can be accessed.

### Health Check Methods

- **HTTP request**

This health check mode is applicable to containers that provide HTTP/HTTPS services. The cluster periodically initiates an HTTP/HTTPS GET request to such containers. If the return code of the HTTP/HTTPS response is within 200–399, the probe is successful. Otherwise, the probe fails. In this health check mode, you must specify a container listening port and an HTTP/HTTPS request path.

For example, for a container that provides HTTP services, the HTTP check path is **/health-check**, the port is 80, and the host address is optional (which defaults to the container IP address). Here, 172.16.0.186 is used as an example, and we can get such a request: GET http://172.16.0.186:80/health-check. The cluster periodically initiates this request to the container.



**Figure 17-82** HTTP request-based check

Advanced Settings ^

Lifecycle | **Health Check** | Environment Variables | Data Storage | Security Context | Log Policies

Regular checks performed to determine the health status of containers and workloads.

Liveness Probe | Readiness Probe

Checks whether the container is running.

Check Method  HTTP request  TCP port  CLI [Reset](#)

Path

Port

Host Address

Protocol  HTTP  HTTPS

Initial Delay (s)

Timeout (s)

- **TCP port**

For a container that provides TCP communication services, the cluster periodically establishes a TCP connection to the container. If the connection is successful, the probe is successful. Otherwise, the probe fails. In this health check mode, you must specify a container listening port.

For example, if you have a Nginx container with service port 80, after you specify TCP port 80 for container listening, the cluster will periodically initiate a TCP connection to port 80 of the container. If the connection is successful, the probe is successful. Otherwise, the probe fails.

**Figure 17-83** TCP port-based check

The screenshot shows the 'Health Check' configuration page. At the top, there are tabs for 'Lifecycle', 'Health Check', 'Environment Variables', 'Data Storage', 'Security Context', and 'Log Policies'. Below the tabs, there is a description: 'Regular checks performed to determine the health status of containers and workloads.' There are two main sections: 'Liveness Probe' (selected) and 'Readiness Probe'. Under 'Liveness Probe', there is a description: 'Checks whether the container is running.' The 'Check Method' is set to 'TCP port' (highlighted with a red box), with other options being 'HTTP request', 'CLI', and 'Reset'. Below this, there are three input fields: 'TCP Port' (with placeholder text 'TCP port to which a probe will connect'), 'Initial Delay (s)' (with placeholder text 'Delay between container startup and the first probe'), and 'Timeout (s)' (with placeholder text 'Number of seconds after which the probe times out').

- **CLI**

CLI is an efficient tool for health check. When using the CLI, you must specify an executable command in a container. The cluster periodically runs the command in the container. If the command output is 0, the health check is successful. Otherwise, the health check fails.

The CLI mode can be used to replace the HTTP request-based and TCP port-based health check.

- For a TCP port, you can write a program script to connect to a container port. If the connection is successful, the script returns **0**. Otherwise, the script returns **-1**.
- For an HTTP request, you can write a program script to run the **wget** command for a container.

**wget http://127.0.0.1:80/health-check**

Check the return code of the response. If the return code is within 200–399, the script returns **0**. Otherwise, the script returns **-1**.

**Figure 17-84** CLI-based check

The screenshot shows the 'Health Check' configuration page. At the top, there are tabs for 'Lifecycle', 'Health Check', 'Environment Variables', 'Data Storage', 'Security Context', and 'Log Policies'. Below the tabs, there is a description: 'Regular checks performed to determine the health status of containers and workloads.' There are two main sections: 'Liveness Probe' (selected) and 'Readiness Probe'. Under 'Liveness Probe', there is a description: 'Checks whether the container is running.' The 'Check Method' is set to 'CLI' (highlighted with a red box), with other options being 'HTTP request', 'TCP port', and 'Reset'. Below this, there are three input fields: 'Command' (with value 'wget http://127.0.0.1:80/health-check'), 'Initial Delay (s)' (with value '30'), and 'Timeout (s)' (with value '10').

**NOTICE**

- Put the program to be executed in the container image so that the program can be executed.
- If the command to be executed is a shell script, do not directly specify the script as the command, but add a script parser. For example, if the script is `/data/scripts/health_check.sh`, you must specify `sh/data/scripts/health_check.sh` for command execution. The reason is that the cluster is not in the terminal environment when executing programs in a container.

## Common Parameter Description

**Table 17-60** Common parameter description

Parameter	Description
Initial Delay (s)	Check delay time in seconds. Set this parameter according to the normal startup time of services. For example, if this parameter is set to 30, the health check will be started 30 seconds after the container is started. The time is reserved for containerized services to start.
Timeout (s)	Timeout duration. Unit: second. For example, if this parameter is set to <b>10</b> , the timeout wait time for performing a health check is 10s. If the wait time elapses, the health check is regarded as a failure. If the parameter is left blank or set to <b>0</b> , the default timeout time is 1s.

### 17.6.12.6 Setting an Environment Variable

#### Scenario

An environment variable is a variable whose value can affect the way a running container will behave. You can modify environment variables even after workloads are deployed, increasing flexibility in workload configuration.

The function of setting environment variables on CCE is the same as that of specifying **ENV** in a Dockerfile.

CCE provides three ways to add environment variables: Manually add environment variables, import environment variables from a secret, and import environment variables from a configMap.

**NOTICE**

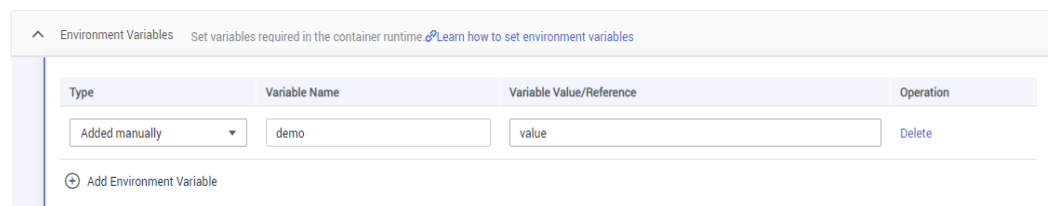
After a container is started, do not modify configurations in the container. If configurations in the container are modified (for example, passwords, certificates, and environment variables of a containerized application are added to the container), the configurations will be lost after the container restarts and container services will become abnormal. An example scenario of container restart is pod rescheduling due to node anomalies.

Configurations must be imported to a container as arguments. Otherwise, configurations will be lost after the container restarts.

## Manually Adding Environment Variables

- Step 1** When creating a workload, add a container image. Then, expand **Environment Variables** and click **Add Environment Variables**.
- Step 2** Configure the following parameters as required:
  - **Type:** Set this to **Added manually**.
  - **Variable Name:** Enter a variable name, for example, demo.
  - **Variable Value/Reference:** Enter a variable value, for example, value.

**Figure 17-85** Manually adding environment variables

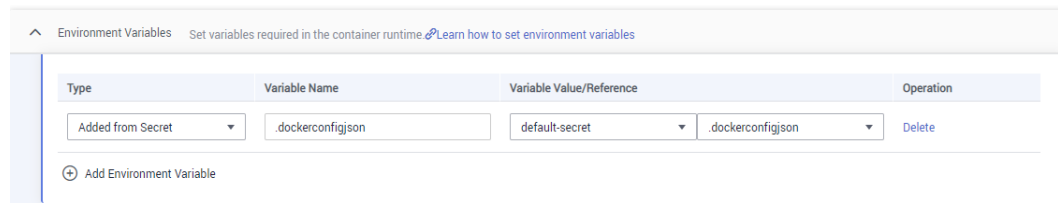


----End

## Importing Environment Variables from a Secret

- Step 1** You need to create a key first. For details, see [Creating a Secret](#).
- Step 2** When creating a workload, add a container image. Then, expand **Environment Variables** and click **Add Environment Variables**.
- Step 3** Configure the following parameters as required:
  - **Type:** Set this to **Added from Secret**.
  - **Variable Name:** Enter a variable name.
  - **Variable Value/Reference:** Select the corresponding secret name and key.

**Figure 17-86** Importing environment variables from a secret

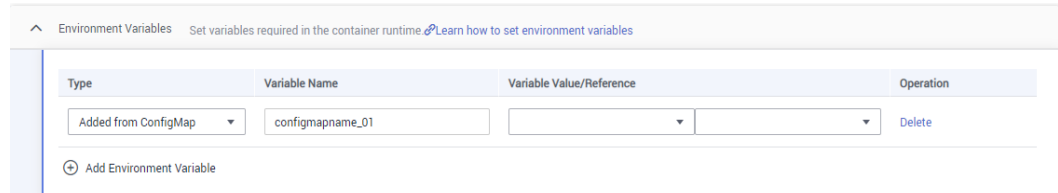


----End

## Importing Environment Variables from a ConfigMap

- Step 1** Create a ConfigMap first. For details, see [Creating a ConfigMap](#).
- Step 2** When creating a workload, add a container image. Then, expand **Environment Variables** and click **Add Environment Variables**.
- Step 3** Configure the following parameters as required:
  - **Type:** Set this to **Added from ConfigMap**.
  - **Variable Name:** Enter a variable name.
  - **Variable Value/Reference:** Select the corresponding ConfigMap name and key.

**Figure 17-87** Importing environment variables from a ConfigMap



----End

### 17.6.12.7 Enabling ICMP Security Group Rules

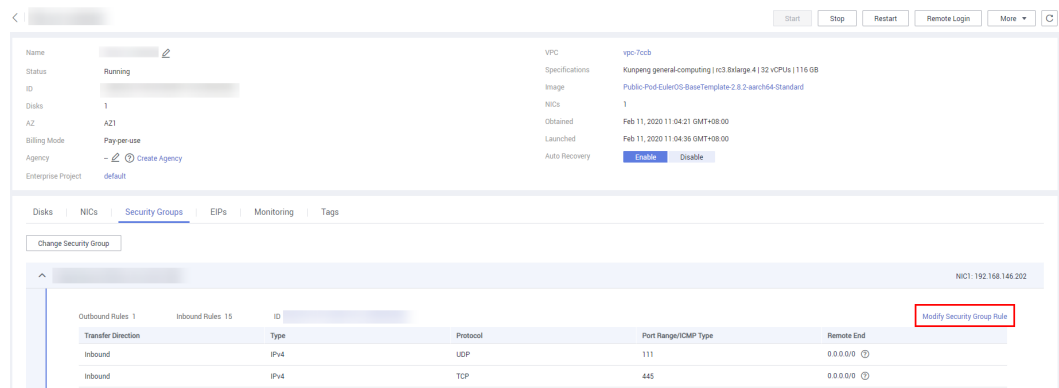
#### Scenario

If a workload uses UDP for both load balancing and health check, enable ICMP security group rules for the backend servers.

#### Procedure

- Step 1** Log in to the Elastic Cloud Server (ECS) console, find the ECS corresponding to any node where the workload runs, and click the ECS name. The ECS details page is displayed.
- Step 2** On the **Security Groups** tab page, click **Modify Security Group Rule**.

**Figure 17-88** Modifying security group rules

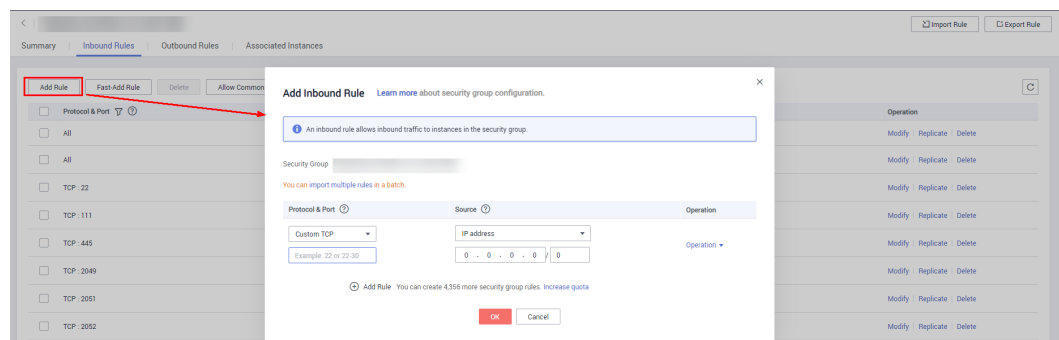


**Step 3** On the page displayed, click the **Inbound Rules** tab and click **Add Rule** to add an inbound rule for ECS. Then, click **OK**. For details about how to add an inbound rule for ECS, see [Figure 17-89](#).

**NOTE**

- You only need to add security group rules to any node where the workload runs.
- The security group must have rules to allow access from the CIDR block 100.125.0.0/16.

**Figure 17-89** Adding a security group rule



----End

### 17.6.12.8 Configuring an Image Pull Policy

When a workload is created, the container image is pulled from the image repository to the node. The image is also pulled when the workload is restarted or upgraded.

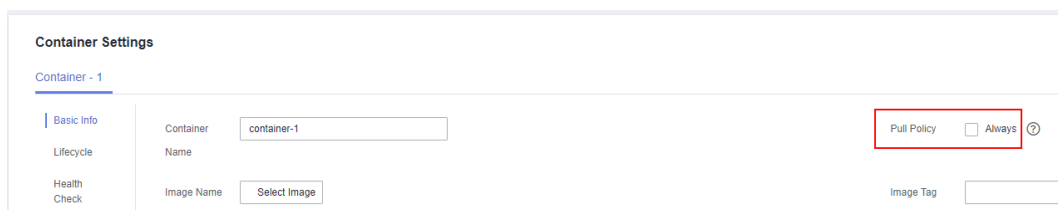
By default, **imagePullPolicy** is set to **IfNotPresent**, indicating that if the image exists on the node, the existing image is used. If the image does not exist on the node, the image is pulled from the image repository.

The image pull policy can also be set to **Always**, indicating that the image is pulled from the image repository and overwrites the image on the node regardless of whether the image exists on the node.

```
apiVersion: v1
kind: Pod
metadata:
  name: nginx
spec:
```

```
containers:
- image: nginx:alpine
  name: container-0
resources:
  limits:
    cpu: 100m
    memory: 200Mi
  requests:
    cpu: 100m
    memory: 200Mi
  imagePullPolicy: Always
imagePullSecrets:
- name: default-secret
```

You can also set the image pull policy when creating a workload on the CCE console. As shown in the following figure, if you select **Always**, the image is always pulled. If you do not select it, the policy will be **IfNotPresent**, which means that the image is not pulled.



### NOTICE

You are advised to use a new tag each time you create an image. If you do not update the tag but only update the image, when **Pull Policy** is set to **IfNotPresent**, CCE considers that an image with the tag already exists on the current node and will not pull the image again.

## 17.6.12.9 Configuring Time Zone Synchronization

When creating a workload, you can configure containers to use the same time zone as the node. You can enable time zone synchronization when creating a workload.

Time Zone Synchronization  If this setting is enabled, the same time zone will be used for both containers and nodes.

The time zone synchronization function depends on the local disk (hostPath) mounted to the container. After time zone synchronization is enabled, **/etc/localtime** of the node is mounted to **/etc/localtime** of the container in HostPath mode, in this way, the node and container use the same time zone configuration file.

```
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
```

### 17.6.12.10 DNS Configuration

Every Kubernetes cluster has a built-in DNS add-on (CoreDNS) to provide domain name resolution for workloads in the cluster. When handling a high concurrency of DNS queries, CoreDNS may encounter a performance bottleneck, that is, it may fail occasionally to fulfill DNS queries. There are cases when Kubernetes workloads initiate unnecessary DNS queries. This makes DNS overloaded if there are many concurrent DNS queries. Tuning DNS configuration for workloads will reduce the risks of DNS query failures to some extent.

For more information about DNS, see [coredns \(System Resource Add-on, Mandatory\)](#).

## DNS Configuration Items

Run the `cat /etc/resolv.conf` command on a Linux node or container to view the DNS resolver configuration file. The following is an example DNS resolver configuration of a container in a Kubernetes cluster:

```
nameserver 10.247.x.x
search default.svc.cluster.local svc.cluster.local cluster.local
options ndots:5
```

### Configuration Options

- **nameserver:** an IP address list of a name server that the resolver will query. If this parameter is set to 10.247.x.x, the resolver will query the kube-dns/CoreDNS. If this parameter is set to another IP address, the resolver will query a cloud or on-premises DNS server.
- **search:** a search list for host-name lookup. When a domain name cannot be resolved, DNS queries will be attempted combining the domain name with each domain in the search list in turn until a match is found or all domains in the search list are tried. For CCE clusters, the search list is currently limited to three domains per container. When a nonexistent domain name is being resolved, eight DNS queries will be initiated because each domain name (including those in the search list) will be queried twice, one for IPv4 and the other for IPv6.
- **options:** options that allow certain internal resolver variables to be modified. Common options include timeout and ndots.



The value **ndots:5** means that if a domain name has fewer than 5 dots (.), DNS queries will be attempted by combining the domain name with each domain in the search list in turn. If no match is found after all the domains in the search list are tried, the domain name is then used for DNS query. If the domain name has 5 or more than 5 dots, it will be tried first for DNS query. In case that the domain name cannot be resolved, DNS queries will be attempted by combining the domain name with each domain in the search list in turn.

For example, the domain name **www.\*\*\*.com** has only two dots (smaller than the value of **ndots**), and therefore the sequence of DNS queries is as follows: **www.\*\*\*.com.default.svc.cluster.local**, **www.\*\*\*.com.svc.cluster.local**, **www.\*\*\*.com.cluster.local**, and **www.\*\*\*.com**. This means that at least seven DNS queries will be initiated before the domain name is resolved into an IP address. It is clear that when many unnecessary DNS queries will be initiated to access an external domain name. There is room for improvement in workload's DNS configuration.

#### NOTE

For more information about configuration options in the resolver configuration file used by Linux operating systems, visit <http://man7.org/linux/man-pages/man5/resolv.conf.5.html>.

## Configuring DNS Using the Workload YAML

When creating a workload using a YAML file, you can configure the DNS settings in the YAML. The following is an example for an Nginx application:

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx
  namespace: default
spec:
  replicas: 1
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
        - name: container-1
          image: nginx:latest
          imagePullPolicy: IfNotPresent
      imagePullSecrets:
        - name: default-secret
      dnsPolicy: None
      dnsConfig:
        options:
          - name: ndots
            value: '5'
          - name: timeout
            value: '3'
        nameservers:
          - 10.2.3.4
        searches:
          - my.dns.search.suffix
```

- **dnsPolicy**

The **dnsPolicy** field is used to configure a DNS policy for an application. The default value is **ClusterFirst**. The following table lists **dnsPolicy** configurations.

**Table 17-61** dnsPolicy

Parameter	Description
ClusterFirst (default value)	Custom DNS configuration added to the default DNS configuration. By default, the application connects to CoreDNS (CoreDNS of the CCE cluster connects to the DNS on the cloud by default). The custom dnsConfig will be added to the default DNS parameters. Containers can resolve both the cluster-internal domain names registered by a Service and the external domain names exposed to public networks. The search list ( <b>search</b> option) and <b>ndots: 5</b> are present in the DNS configuration file. Therefore, when accessing an external domain name and a long cluster-internal domain name (for example, <code>kubernetes.default.svc.cluster.local</code> ), the search list will usually be traversed first, resulting in at least six invalid DNS queries. The issue of invalid DNS queries disappears only when a short cluster-internal domain name (for example, <code>kubernetes</code> ) is being accessed.
ClusterFirstWithHostNet	By default, the applications configured with the host network are interconnected with the DNS configuration of the node where the pod is located. The DNS configuration is specified in the DNS file that the kubelet <b>--resolv-conf</b> parameter points to. In this case, the CCE cluster uses the DNS on the cloud. If workloads need to use Kube-DNS/ CoreDNS of the cluster, set <b>dnsPolicy</b> to <b>ClusterFirstWithHostNet</b> and container's DNS configuration file is the same as ClusterFirst, in which invalid DNS queries still exist. ... spec: containers: - image: nginx:latest imagePullPolicy: IfNotPresent name: container-1 restartPolicy: Always <b>hostNetwork: true</b> <b>dnsPolicy: ClusterFirstWithHostNet</b>
Default	The DNS configuration of the node where the pod is located is inherited, and the custom DNS configuration is added to the inherited configuration. Container's DNS configuration file is the DNS configuration file that the kubelet's <b>--resolv-conf</b> flag points to. In this case, a cloud DNS is used for CCE clusters. Both <b>search</b> and <b>options</b> fields are left unspecified. This configuration can only resolve the external domain names registered with the Internet, and not cluster-internal domain names. This configuration is free from the issue of invalid DNS queries.

Parameter	Description
None	The default DNS configuration is replaced by the custom DNS configuration, and only the custom DNS configuration is used. If <b>dnsPolicy</b> is set to <b>None</b> , the <b>dnsConfig</b> field must be specified because all DNS settings are supposed to be provided using the <b>dnsConfig</b> field.

 NOTE

If the **dnsPolicy** field is not specified, the default value is **ClusterFirst** instead of **Default**.

- **dnsConfig**

The **dnsConfig** field is used to configure DNS parameters for workloads. The configured parameters are merged to the DNS configuration file generated according to **dnsPolicy**. If **dnsPolicy** is set to **None**, the workload's DNS configuration file is specified by the **dnsConfig** field. If **dnsPolicy** is not set to **None**, the DNS parameters configured in **dnsConfig** are added to the DNS configuration file generated according to **dnsPolicy**.

**Table 17-62** dnsConfig

Parameter	Description
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 in <b>dnsPolicy</b> . Duplicate entries are removed.
nameservers	A list of IP addresses that will be used as DNS servers. If workload's <b>dnsPolicy</b> is set to <b>None</b> , the list must contain at least one IP address, otherwise this property is optional. The servers listed will be combined to the nameservers generated from the specified DNS policy in <b>dnsPolicy</b> with duplicate addresses removed. <b>NOTE</b> A maximum of three DNS addresses can be configured for a nameserver in the container DNS configuration file. <ul style="list-style-type: none"> <li>• If <b>dnsPolicy</b> is set to <b>ClusterFirst</b> and the cluster uses <b>CoreDNS</b>, you can add two custom DNS addresses in addition to the CoreDNS address. Excess DNS addresses are invalid.</li> <li>• If <b>dnsPolicy</b> is set to <b>ClusterFirst</b> and the cluster uses <b>CoreDNS</b> and <b>NodeLocal DNSCache</b>, you can add one custom DNS address in addition to the CoreDNS and NodeLocal DNSCache addresses. Excess DNS addresses are invalid.</li> </ul>

Parameter	Description
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 search domain names generated from the chosen DNS policy in <b>dnsPolicy</b> . Duplicate domain names are removed. Kubernetes allows for at most 6 search domains.

## Configuration Examples

The following example describes how to configure DNS for workloads.

- **Use Case 1: Using Kube-DNS/CoreDNS Built in Kubernetes Clusters**

### Scenario

Kubernetes in-cluster Kube-DNS/CoreDNS applies to resolving only cluster-internal domain names or cluster-internal domain names + external domain names. This is the default DNS for workloads.

### Example:

```
apiVersion: v1
kind: Pod
metadata:
  namespace: default
  name: dns-example
spec:
  containers:
  - name: test
    image: nginx:alpine
  dnsPolicy: ClusterFirst
  imagePullSecrets:
  - name: default-secret
```

Container's DNS configuration file:

```
nameserver 10.247.3.10
search default.svc.cluster.local svc.cluster.local cluster.local
options ndots:5
```

- **Use Case 2: Using a Cloud DNS**

### Scenario

A DNS cannot resolve cluster-internal domain names and therefore applies to the scenario where workloads access only external domain names registered with the Internet.

### Example:

```
apiVersion: v1
kind: Pod
metadata:
  namespace: default
  name: dns-example
spec:
  containers:
  - name: test
    image: nginx:alpine
  dnsPolicy: Default # The DNS configuration file that the kubelet --resolv-conf parameter points to is used. In this case, the CCE cluster uses the DNS on the cloud.
  imagePullSecrets:
  - name: default-secret
```

Container's DNS configuration file:

```
nameserver 100.125.x.x
```

- **Use Case 3: Using Kube-DNS/CoreDNS for Workloads Running with hostNetwork**

### Scenario

By default, a DNS is used for workloads running with hostNetwork. If workloads need to use Kube-DNS/CoreDNS, set **dnsPolicy** to **ClusterFirstWithHostNet**.

### Example:

```
apiVersion: v1
kind: Pod
metadata:
  name: nginx
spec:
  hostNetwork: true
  dnsPolicy: ClusterFirstWithHostNet
  containers:
  - name: nginx
    image: nginx:alpine
    ports:
    - containerPort: 80
  imagePullSecrets:
  - name: default-secret
```

Container's DNS configuration file:

```
nameserver 10.247.3.10
search default.svc.cluster.local svc.cluster.local cluster.local
options ndots:5
```

- **Use Case 4: Customizing Application's DNS Configuration**

### Scenario

You can flexibly customize the DNS configuration file for applications. Using **dnsPolicy** and **dnsConfig** together can address almost all scenarios, including the scenarios in which an on-premises DNS will be used, multiple DNSs will be cascaded, and DNS configuration options will be modified.

### Example 1: Using Your On-Premises DNS

Set **dnsPolicy** to **None** so application's DNS configuration file is generated based on **dnsConfig**.

```
apiVersion: v1
kind: Pod
metadata:
  namespace: default
  name: dns-example
spec:
  containers:
  - name: test
    image: nginx:alpine
  dnsPolicy: "None"
  dnsConfig:
    nameservers:
    - 10.2.3.4 # IP address of your on-premises DNS
    searches:
    - ns1.svc.cluster.local
    - my.dns.search.suffix
    options:
    - name: ndots
      value: "2"
    - name: timeout
      value: "3"
  imagePullSecrets:
  - name: default-secret
```

Container's DNS configuration file:

```
nameserver 10.2.3.4
search ns1.svc.cluster.local my.dns.search.suffix
options timeout:3 ndots:2
```

### Example 2: Modifying the ndots Option in the DNS Configuration File to Reduce Invalid DNS Queries

Set **dnsPolicy** to a value other than **None** so the DNS parameters configured in **dnsConfig** are added to the DNS configuration file generated based on **dnsPolicy**.

```
apiVersion: v1
kind: Pod
metadata:
  namespace: default
  name: dns-example
spec:
  containers:
  - name: test
    image: nginx:alpine
    dnsPolicy: "ClusterFirst"
  dnsConfig:
    options:
    - name: ndots
      value: "2" # The ndots:5 option in the DNS configuration file generated based on the
ClusterFirst policy is changed to ndots:2.
  imagePullSecrets:
  - name: default-secret
```

Container's DNS configuration file:

```
nameserver 10.247.3.10
search default.svc.cluster.local svc.cluster.local cluster.local
options ndots:2
```

### Example 3: Using Multiple DNSs in Serial Sequence

```
apiVersion: v1
kind: Pod
metadata:
  namespace: default
  name: dns-example
spec:
  containers:
  - name: test
    image: nginx:alpine
    dnsPolicy: ClusterFirst # Added DNS configuration. The cluster connects to CoreDNS by default.
  dnsConfig:
    nameservers:
    - 10.2.3.4 # IP address of your on-premises DNS
  imagePullSecrets:
  - name: default-secret
```

#### NOTE

A maximum of three DNS addresses can be configured for a nameserver in the container DNS configuration file.

- If **dnsPolicy** is set to **ClusterFirst** and the cluster uses **CoreDNS**, you can add two custom DNS addresses in addition to the CoreDNS address. Excess DNS addresses are invalid.
- If **dnsPolicy** is set to **ClusterFirst** and the cluster uses **CoreDNS** and **NodeLocal DNSCache**, you can add one custom DNS address in addition to the CoreDNS and NodeLocal DNSCache addresses. Excess DNS addresses are invalid.

Container's DNS configuration file:

```
nameserver 10.247.3.10 10.2.3.4
search default.svc.cluster.local svc.cluster.local cluster.local
options ndots:5
```

## 17.6.12.11 Pod Scale-in Priorities

### Scale-in Priorities

Pod scale-in is performed based on the following priorities:

1. Pods that are not scheduled
2. Pod Pending < Pod Unknown < Pod Running
3. Not ready < Ready
4. Pod with a specific label (cce.io/priordeletion)
5. Doubled up < Not doubled up (nodes with more pods are first downsized)
6. Pods that take a long time to be ready for scale-in
7. Pods that are frequently restarted
8. Pods with an earlier **createtime**

### Preferential Scale-in of Old Pods

- Where such scale-in **does not work**:  
According to the preceding priorities, assume that the number of nodes is 2 (the same specifications) and the number of pods in a Deployment is 5 (or any odd number), three pods distributed on node A and two on node B. If the number of pods is reduced to 3, **Rule 5** is performed first, and the system determines that the node to be scaled in is node A. Then **Rule 6** is performed. After an old pod is deleted, it is found that no old pod exists and the scale-in node remains node A. In this case, a new pod will be deleted.
- **Where such scale-in works**: Assume that the number of nodes is 2 (the same specifications) and the number of Deployment pods is 6 (or any even number). These pods are distributed evenly on nodes A and B. If the number of pods is reduced to 4, the system determines that the **Rule 5** is not met. In this case, both nodes A and B will be scaled in. Then, the system performs **Rule 6** to sort the pods on the two nodes and deletes two old pods.

## 17.6.13 Configuring QoS Rate Limiting for Inter-Pod Access

### Scenario

Bandwidth preemption occurs between different containers deployed on the same node, which may cause service jitter. You can configure QoS rate limiting for inter-pod access to prevent this problem.

### Notes and Constraints

You can configure limiting for the container tunnel network, VPC network, and Cloud Native 2.0 Network models. The latter two models must comply with the following restrictions:

- Only clusters later than v1.19.10 are supported.
- Only common containers (runC as the container runtime) are supported. Secure containers (Kata as the container runtime) are not supported.
- Only the rate of access from pods to pods is limited. Access to nodes and external access are not affected.

- The upper bandwidth limit must be the smaller value between the upper limit of the server model bandwidth and 4.3 Gbit/s.
- Currently, only rate limit at the Mbit/s or higher level is supported.

## Procedure

You can add annotations to a pod to specify its egress and ingress bandwidth.

```
apiVersion: v1
kind: Pod
metadata:
  annotations:
    kubernetes.io/ingress-bandwidth: 100M
    kubernetes.io/egress-bandwidth: 100M
...
```

- **kubernetes.io/ingress-bandwidth**: ingress bandwidth of the pod
- **kubernetes.io/egress-bandwidth**: egress bandwidth of the pod

If these two parameters are not specified, the bandwidth is not limited.

## 17.6.14 Adding Pod Annotations

CCE allows you to add annotations to a YAML file to realize some advanced pod functions. The following table describes the annotations you can add.

**Table 17-63** Pod Annotation

Annotation	Description	Default Value
kubernetes.AOM.log.stdout	Standard output parameter. If not specified, the standard log output of all containers is reported to AOM. You can collect stdout logs from certain containers or ignore them at all. Example: <ul style="list-style-type: none"> <li>• Collecting none of the stdout logs: kubernetes.AOM.log.stdout: '[]'</li> <li>• Collecting stdout logs of container-1 and container-2: kubernetes.AOM.log.stdout: '["container-1","container-2"]'</li> </ul>	-
metrics.alpha.kubernetes.io/custom-endpoints	Parameter for reporting AOM monitoring metrics that you specify. For details, see <a href="#">Custom Monitoring</a> .	-



Annotation	Description	Default Value
prometheus.io/scrape	Parameter for reporting Prometheus metrics. If the value is <b>true</b> , the current workload reports the monitoring metrics. For details, see <a href="#">Monitoring by Using the prometheus Add-on</a> .	-
prometheus.io/path	URL for Prometheus to collect data. For details, see <a href="#">Monitoring by Using the prometheus Add-on</a> .	/metrics
prometheus.io/port	Endpoint port number for Prometheus to collect data. For details, see <a href="#">Monitoring by Using the prometheus Add-on</a> .	-
prometheus.io/scheme	Protocol used by Prometheus to collect data. The value can be <b>http</b> or <b>https</b> . For details, see <a href="#">Monitoring by Using the prometheus Add-on</a> .	-

## 17.7 Affinity and Anti-Affinity Scheduling

### 17.7.1 Scheduling Policy Overview

CCE supports custom and simple scheduling policies. A custom scheduling policy allows you to customize node affinity, workload affinity, and workload anti-affinity to meet higher requirements. A simple scheduling policy provides a simple and convenient scheduling mode with sufficient functions.

#### Custom Scheduling Policies

You can configure node affinity, workload affinity, and workload anti-affinity in custom scheduling policies.

- [Node Affinity](#)
- [Workload Affinity](#)
- [Workload Anti-Affinity](#)

#### NOTE

Custom scheduling policies depend on node labels and pod labels. You can use default labels or customize labels as required.

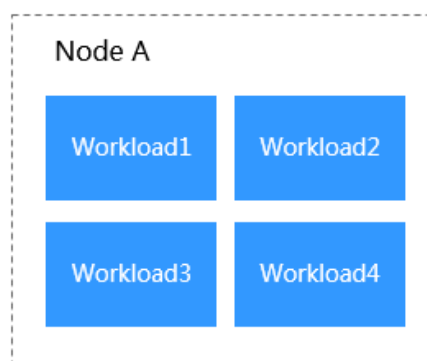
## Simple Scheduling Policies

A simple scheduling policy allows you to configure affinity between workloads and AZs, between workloads and nodes, and between workloads.

- **Workload-AZ affinity:** Multiple AZ-based scheduling policies (including affinity and anti-affinity policies) can be configured. However, scheduling is performed as long as one of the scheduling policies is met.
  - **Affinity between workloads and AZs:** [Workload-AZ Affinity](#)
  - **Anti-affinity between workloads and AZs:** [Workload-AZ Anti-Affinity](#)
- **Workload-node affinity:** Multiple node-based scheduling policies (including affinity and anti-affinity scheduling) can be configured. However, scheduling is performed as long as one of the scheduling policies is met. For example, if a cluster contains nodes A, B, and C and two scheduling policies are set (one policy defines node A as an affinity node and the other policy defines node B as an anti-affinity node), then the workload can be scheduled to any node other than B.
  - **Affinity between workloads and nodes:** [Workload-Node Affinity](#)
  - **Anti-affinity between workloads and nodes:** [Workload-Node Anti-Affinity](#)
- **Workload-workload affinity:** Multiple workload-based scheduling policies can be configured, but the labels in these policies must belong to the same workload.
  - **Affinity between workloads:** For details, see [Workload-Workload Affinity](#). You can deploy workloads on the same node to reduce consumption of network resources.

[Figure 17-90](#) shows an example of affinity deployment, in which all workloads are deployed on the same node.

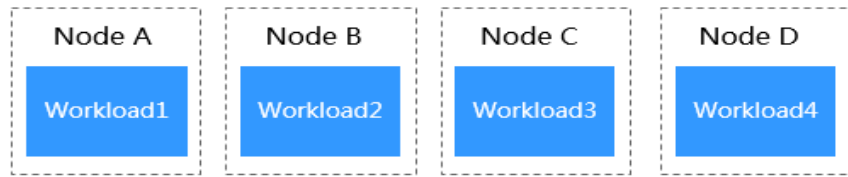
**Figure 17-90** Affinity between workloads



- **Anti-affinity between workloads:** For details, see [Workload-Workload Anti-Affinity](#). Constraining multiple instances of the same workload from being deployed on the same node reduces the impact of system breakdowns. Anti-affinity deployment is also recommended for workloads that may interfere with each other.

[Figure 17-91](#) shows an example of anti-affinity deployment, in which four workloads are deployed on four different nodes.

Figure 17-91 Anti-affinity between workloads



### NOTICE

When setting workload-workload affinity and workload-node affinity, ensure that the affinity relationships do not contradict each other; otherwise, workload deployment will fail.

For example, Workload 3 will fail to be deployed when the following conditions are met:

- Anti-affinity is configured for Workload 1 and Workload 2. Workload 1 is deployed on **Node A** and Workload 2 is deployed on **Node B**.
- Affinity is configured between Workload 2 and Workload 3, but the target node on which Workload 3 is to be deployed is **Node C** or **Node A**.

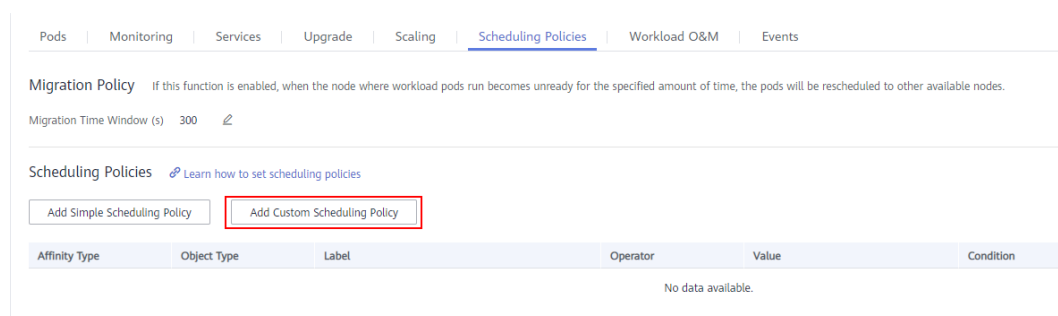
## 17.7.2 Custom Scheduling Policies

### 17.7.2.1 Node Affinity

#### Using the CCE Console

- Step 1** Log in to the CCE console and choose **Workloads > Deployments** or **Workloads > StatefulSets** in the navigation pane.
- Step 2** Click a workload name in the Deployment or StatefulSet list. On the displayed workload details page, click the **Scheduling Policies** tab and then click **Add Custom Scheduling Policy**.

Figure 17-92 Adding a custom scheduling policy



- Step 3** In the **Node Affinity** area, you can specify node labels to meet required or preferred rules in scheduling.

**Table 17-64** Node affinity settings

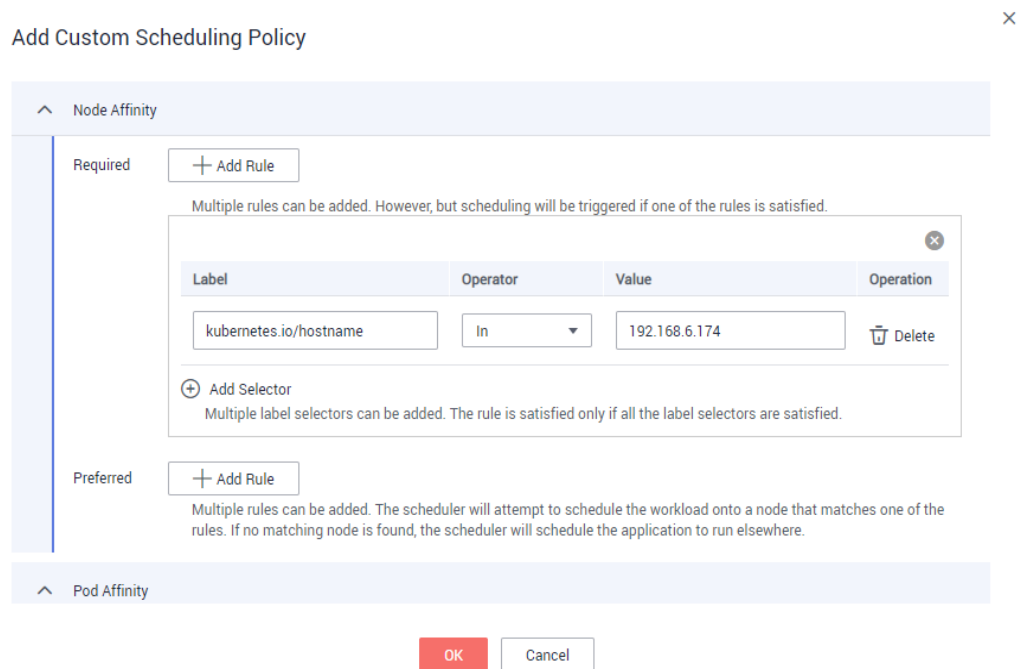
Parameter	Description
Required	It specifies a rule that must be met in scheduling. It corresponds to <b>requiredDuringSchedulingIgnoredDuringExecution</b> in Kubernetes. You can click <b>Add Rule</b> to add multiple required rules. A pod will be scheduled on a node that meets any of the rules configured.
Preferred	It specifies a preference in scheduling. It corresponds to <b>preferredDuringSchedulingIgnoredDuringExecution</b> in Kubernetes. You can click <b>Add Rule</b> to add multiple preferred rules. The scheduler will try to enforce the rules but will not guarantee. If the scheduler cannot satisfy any one of the rules, the pod will still be scheduled.

**Step 4** Set a rule according to the following table. You can click **Add Selector** to configure multiple selectors for a rule.

**Table 17-65** Selector settings

Parameter	Description
Weight	<ul style="list-style-type: none"> <li>This parameter is unavailable for a required rule.</li> <li>Set the weight of a preferred rule. A higher weight indicates a higher priority.</li> </ul>
Label	Node label. You can use the default label or customize a label.
Operator	The following relations are supported: <b>In</b> , <b>NotIn</b> , <b>Exists</b> , <b>DoesNotExist</b> , <b>Gt</b> , and <b>Lt</b>
Value	Tag value. Operators <b>In</b> and <b>NotIn</b> allow one or more label values. Values are separated with colons (;). Operators <b>Exists</b> and <b>DoesNotExist</b> are used to determine whether a label exists, and do not require a label value. If you set the operator to <b>Gt</b> or <b>Lt</b> for a label, the label value must be greater than or less than a certain integer.
Operation	You can click <b>Delete</b> to delete a selector.
Add Selector	A selector corresponds to <b>matchExpressions</b> in Kubernetes. You can click <b>Add Selector</b> to add multiple selectors for a scheduling rule. The rule is applied in scheduling only when all its selectors are satisfied.

**Figure 17-93** Node affinity scheduling policy



----End

## Using kubectl

This section uses Nginx as an example to describe how to configure node affinity.

### Prerequisites

A workload that uses the nginx container image has been deployed on a node.

### Procedure

Set **Label** to **kubernetes.io/hostname**, add affinity nodes, and set the operator to **In**. Then, click **OK**.

YAML file of the workload with node affinity:

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx
  namespace: default
spec:
  replicas: 2
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - image: nginx
        imagePullPolicy: Always
        name: nginx
        imagePullSecrets:
```

```
- name: default-secret
affinity:
  nodeAffinity:
    requiredDuringSchedulingIgnoredDuringExecution:
      nodeSelectorTerms:
        - matchExpressions:
            - key: kubernetes.io/hostname
              operator: In
              values:
                - 192.168.6.174
```

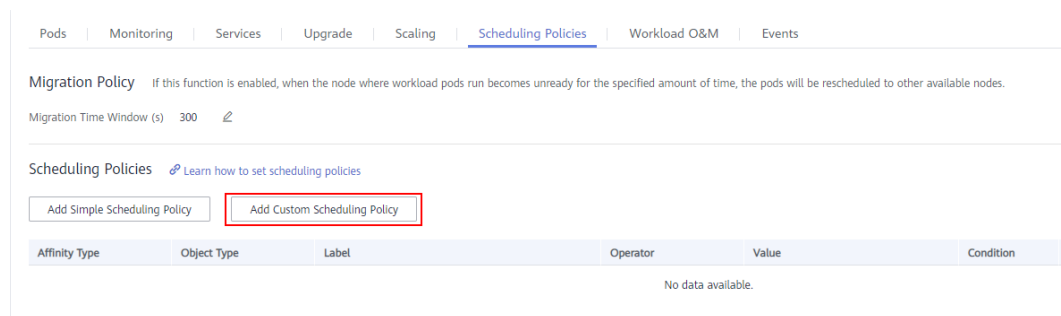
## 17.7.2.2 Workload Affinity

### Using the CCE Console

Workload affinity determines the pods as which the target workload will be deployed in the same topology domain.

- Step 1** Log in to the CCE console and choose **Workloads > Deployments** or **Workloads > StatefulSets** in the navigation pane.
- Step 2** Click a workload name in the Deployment or StatefulSet list. On the displayed workload details page, click the **Scheduling Policies** tab and then click **Add Custom Scheduling Policy**.

**Figure 17-94** Adding a custom scheduling policy



- Step 3** In the **Pod Affinity** area, set the namespace, topology key, and the label requirements to be met.

There are two types of pod affinity rules: **Required** (hard rule) and **Preferred** (soft rule). The label operators include **In**, **NotIn**, **Exists**, and **DoesNotExist**.

**Table 17-66** Pod affinity settings

Parameter	Description
Required	It specifies a rule that must be met in scheduling. It corresponds to <b>requiredDuringSchedulingIgnoredDuringExecution</b> in Kubernetes. You can click <b>Add Rule</b> to add multiple required rules. Ensure that all the labels specified in the rules must be in the same workload. Each rule requires a namespace and topology key.

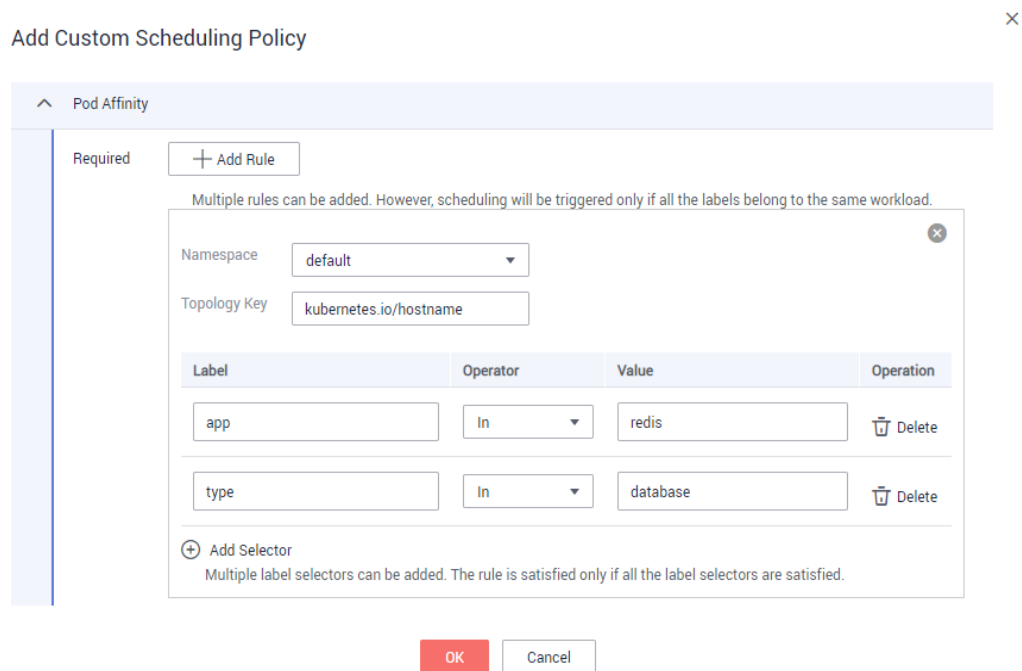
Parameter	Description
Preferred	It specifies a preference in scheduling. It corresponds to <b>preferredDuringSchedulingIgnoredDuringExecution</b> in Kubernetes. You can click <b>Add Rule</b> to add multiple preferred rules. The scheduler will try to enforce the rules but will not guarantee. If the scheduler cannot satisfy any one of the rules, the pod will still be scheduled.

**Step 4** Set a rule according to the following table. You can click **Add Selector** to configure multiple selectors for a rule.

**Table 17-67** Selector settings

Parameter	Description
Weight	<ul style="list-style-type: none"> <li>This parameter is unavailable for a required rule.</li> <li>Set the weight of a preferred rule. A higher weight indicates a higher priority.</li> </ul>
Namespace	By default, the namespace of the current pod is used. You can also use another namespace.
Topology Key	Key of the worker node label that the system uses to denote a topology domain in which scheduling can be performed. Default and custom node labels can be used.
Label	Label of the workload. You can customize the label name.
Operator	The following relations are supported: <b>In</b> , <b>NotIn</b> , <b>Exists</b> , and <b>DoesNotExist</b>
Value	Tag value. Operators <b>In</b> and <b>NotIn</b> allow one or more label values. Values are separated with colons (;). Operators <b>Exists</b> and <b>DoesNotExist</b> are used to determine whether a label exists, and do not require a label value.
Operation	You can click <b>Delete</b> to delete a selector.
Add Selector	A selector corresponds to <b>matchExpressions</b> in Kubernetes. You can click <b>Add Selector</b> to add multiple selectors for a scheduling rule. The rule is applied in scheduling only when all its selectors are satisfied.

**Figure 17-95** Pod affinity scheduling policy



----End

## Using kubectl

This section uses Nginx as an example to describe how to configure pod affinity.

### Prerequisites

A workload that uses the nginx container image has been deployed on a node.

### Procedure

Set **Namespace** to **default** and **Topology Key** to the built-in node label **kubernetes.io/hostname**, which means that the scheduling scope is a node. Set labels **app** and **type** and their value to **redis** and **database**, respectively. Set **Operator** to **In** and click **OK**.

The YAML of the workload with pod affinity is as follows:

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx
  namespace: default
spec:
  replicas: 2
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
        - image: nginx
          imagePullPolicy: Always
```



```

name: nginx
imagePullSecrets:
- name: default-secret
affinity:
  nodeAffinity: {}
  podAffinity:
    requiredDuringSchedulingIgnoredDuringExecution:
      - labelSelector:
          matchExpressions:
            - key: app
              operator: In
              values:
                - redis
            - key: type
              operator: In
              values:
                - database
      namespaces:
        - default
    topologyKey: kubernetes.io/hostname
    
```

#### NOTICE

In this example, only when a candidate workload (for example, workload A) with both labels **app=redis** and **type=database** is found can the workload Nginx be successfully scheduled to the node of the candidate workload.

### 17.7.2.3 Workload Anti-Affinity

#### Using the CCE Console

Workload anti-affinity determines the pods from which the target workload will be deployed in a different topology domain.

- Step 1** Log in to the CCE console and choose **Workloads > Deployments** or **Workloads > StatefulSets** in the navigation pane.
- Step 2** Click a workload name in the Deployment or StatefulSet list. On the displayed workload details page, click the **Scheduling Policies** tab and then click **Add Custom Scheduling Policy**.
- Step 3** In the **Pod Anti-Affinity** area, set the namespace, topology key, and the label requirements to be met.

There are two types of pod anti-affinity rules: **Required** (hard rule) and **Preferred** (soft rule), and the label operators include **In**, **NotIn**, **Exists**, and **DoesNotExist**.

**Table 17-68** Workload anti-affinity settings

Parameter	Description
Required	It specifies a rule that must be met in scheduling. It corresponds to <b>requiredDuringSchedulingIgnoredDuringExecution</b> in Kubernetes. You can add multiple required rules. Ensure that all the labels specified in the rules must be in the same workload. Each rule requires a namespace and topology key.

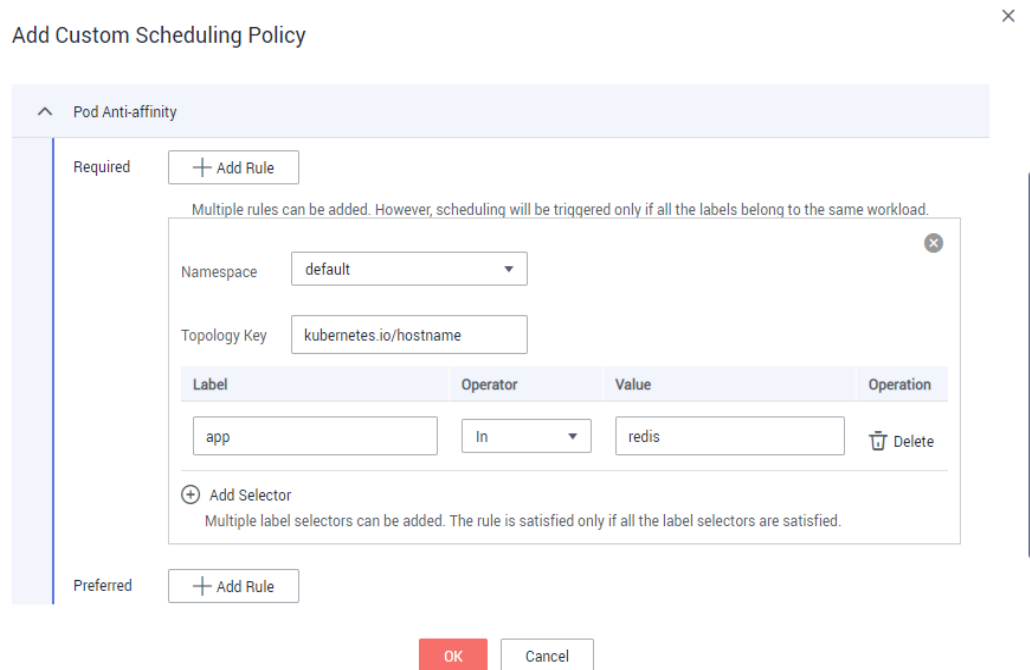
Parameter	Description
Preferred	It specifies a preference in scheduling. It corresponds to <b>preferredDuringSchedulingIgnoredDuringExecution</b> in Kubernetes. You can add multiple preferred rules. The scheduler will try to enforce the rules but will not guarantee. If the scheduler cannot satisfy any one of the rules, the pod will still be scheduled.

**Step 4** Set a rule according to the following table. You can click **Add Selector** to configure multiple selectors for a rule.

**Table 17-69** Selector settings

Parameter	Description
Weight	<ul style="list-style-type: none"> <li>This parameter is unavailable for a required rule.</li> <li>Set the weight of a preferred rule. A higher weight indicates a higher priority.</li> </ul>
Namespace	By default, the namespace of the current pod is used. You can also use another namespace.
Topology Key	Key of the worker node label that the system uses to denote a topology domain in which scheduling can be performed. Default and custom node labels can be used.
Label	Label of the workload. You can customize the label name.
Operator	The following relations are supported: <b>In</b> , <b>NotIn</b> , <b>Exists</b> , and <b>DoesNotExist</b>
Value	Tag value. Operators <b>In</b> and <b>NotIn</b> allow one or more label values. Values are separated with colons (;). Operators <b>Exists</b> and <b>DoesNotExist</b> are used to determine whether a label exists, and do not require a label value.
Operation	You can click <b>Delete</b> to delete a selector.
Add Selector	A selector corresponds to <b>matchExpressions</b> in Kubernetes. You can click <b>Add Selector</b> to add multiple selectors for a scheduling rule. The rule is applied in scheduling only when all its selectors are satisfied.

**Figure 17-96** Pod anti-affinity scheduling policy



----End

## Using kubectl

This section uses Nginx as an example to describe how to configure pod anti-affinity.

### Prerequisites

A workload that uses the nginx container image has been deployed on a node.

### Procedure

Set **Namespace** to **default** and **Topology Key** to the built-in node label **kubernetes.io/hostname**, which means that the scheduling scope is a node. Set the label **app** and its value to **redis**. Set **Operator** to **In** and click **OK**.

The YAML of the workload with pod anti-affinity:


```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx
  namespace: default
spec:
  replicas: 2
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
        - image: nginx
          imagePullPolicy: Always
```

```
name: nginx
imagePullSecrets:
- name: default-secret
affinity:
nodeAffinity: {}
podAffinity: {}
podAntiAffinity:
requiredDuringSchedulingIgnoredDuringExecution:
- labelSelector:
matchExpressions:
- key: app
operator: In
values:
- redis
namespaces:
- default
topologyKey: kubernetes.io/hostname
```

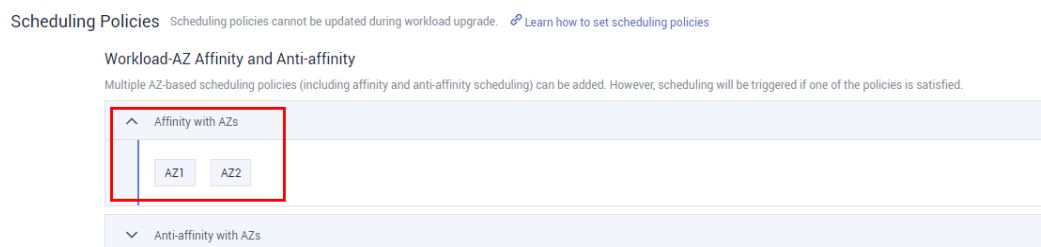
## 17.7.3 Simple Scheduling Policies

### 17.7.3.1 Workload-AZ Affinity

#### Using the CCE Console

**Step 1** When [Creating a Deployment](#) or [Creating a StatefulSet](#), in the **Scheduling Policies** area on the **Configure Advanced Settings** page, click  next to **Workload-AZ Affinity and Anti-affinity** > **Affinity with AZs**.

**Figure 17-97** Scheduling policies



**Step 2** Select the AZ in which you want to deploy the workload.

The created workload will be deployed in the selected AZ.

----End

#### Using kubectl

This section uses an Nginx workload as an example to describe how to create a workload using kubectl.

##### Prerequisites

The ECS where the kubectl client runs has been connected to your cluster. For details, see [Connecting to a Cluster Using kubectl](#).

##### Procedure

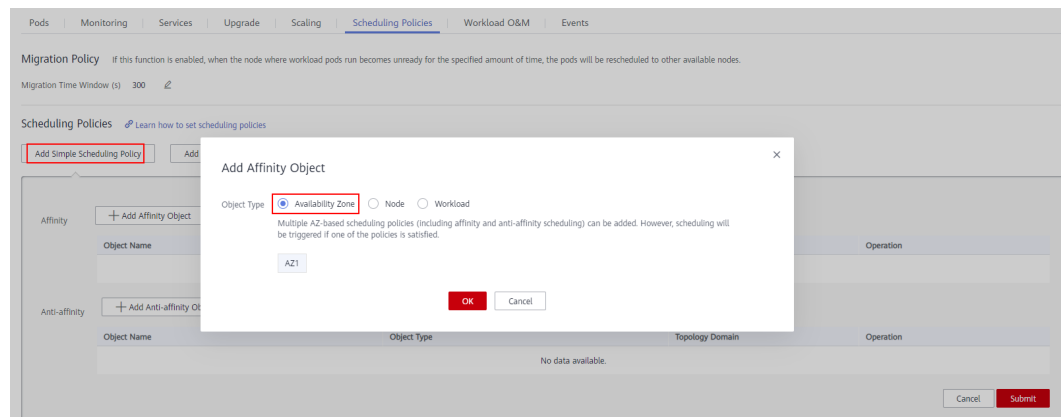
When [using kubectl to create a Deployment](#) or [using kubectl to create a StatefulSet](#), configure workload-AZ affinity. The following is an example YAML file for workload-AZ affinity.

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: az-in-deployment
spec:
  replicas: 1
  selector:
    matchLabels:
      app: az-in-deployment
  strategy:
    type: RollingUpdate
  template:
    metadata:
      labels:
        app: az-in-deployment
    spec:
      containers:
        - image: nginx
          imagePullPolicy: Always
          name: nginx
      imagePullSecrets:
        - name: default-secret
      affinity:
        nodeAffinity:
          requiredDuringSchedulingIgnoredDuringExecution:
            nodeSelectorTerms:
              - matchExpressions:
                  - key: failure-domain.beta.kubernetes.io/zone #node's label key
                    operator: In
                    values:
                      - az1 #node's key value
```

## Setting the Object Type After Creating a Workload

- Step 1** Log in to the CCE console and choose **Workloads > Deployments** or **Workloads > StatefulSets** in the navigation pane.
- Step 2** Click the name of the workload for which you will add a scheduling policy. On the workload details page, choose **Scheduling Policies > Add Simple Scheduling Policy > Add Affinity Object**.

**Figure 17-98** Adding an affinity object – Availability Zone



- Step 3** Set **Object Type** to **Availability Zone**, and select the AZ in which the workload is eligible to be deployed. The workload will be deployed in the selected AZ.

 NOTE

This method can be used to add, edit, or delete scheduling policies.

----End

### 17.7.3.2 Workload-AZ Anti-Affinity

#### Using the CCE Console


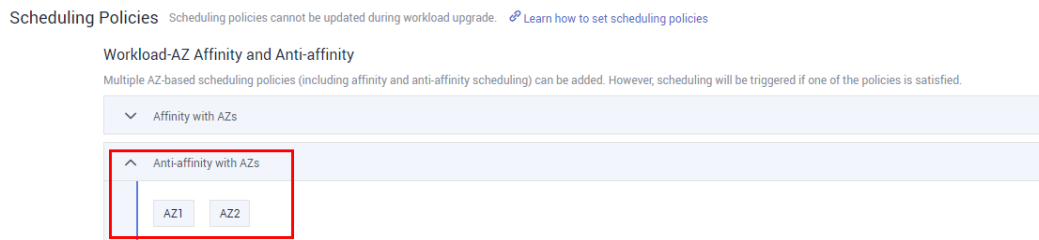
**Step 1** When [Creating a Deployment](#) or [Creating a StatefulSet](#), in the **Scheduling Policies** area on the **Configure Advanced Settings** page, click  next to **Workload-AZ Affinity and Anti-affinity > Anti-affinity with AZs**.

Figure 17-99 Scheduling policies



**Step 2** Select an AZ in which the workload is ineligible to be deployed.

The created workload is not deployed on the selected AZ.

----End

#### Using kubectl

This section uses Nginx as an example to describe how to create a workload using kubectl.

##### Prerequisites

The ECS where the kubectl client runs has been connected to your cluster. For details, see [Connecting to a Cluster Using kubectl](#).

##### Procedure

When [using kubectl to create a Deployment](#) or [using kubectl to create a StatefulSet](#), configure workload-AZ anti-affinity. The following is an example YAML file for workload-AZ anti-affinity.

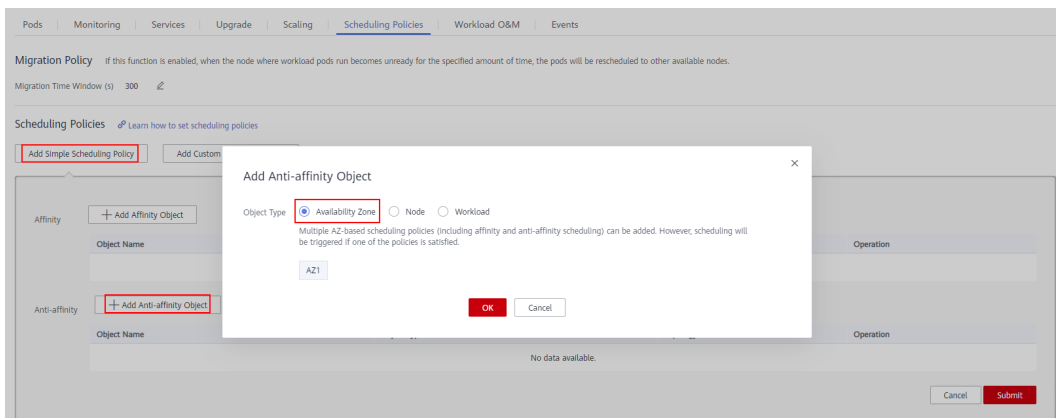
```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx
spec:
  replicas: 1
  selector:
    matchLabels:
      app: nginx
  strategy:
    type: RollingUpdate
  template:
```

```
metadata:  
  labels:  
    app: nginx  
spec:  
  containers:  
  - image: nginx  
    imagePullPolicy: Always  
    name: nginx  
  imagePullSecrets:  
  - name: default-secret  
  affinity:  
    nodeAffinity:  
      requiredDuringSchedulingIgnoredDuringExecution:  
        nodeSelectorTerms:  
        - matchExpressions:  
          - key: failure-domain.beta.kubernetes.io/zone      #node's label key  
            operator: NotIn  
            values:  
            - az1      #node's key value
```

## Setting the Object Type After Creating a Workload

- Step 1** Log in to the CCE console and choose **Workloads > Deployments** or **Workloads > StatefulSets** in the navigation pane.
- Step 2** Click the name of the workload for which you will add a scheduling policy. On the workload details page, choose **Scheduling Policies > Add Simple Scheduling Policy > Add Anti-affinity Object**.

Figure 17-100 Adding an anti-affinity object – Availability Zone



- Step 3** Set **Object Type** to **Availability Zone** and select the AZ in which the workload is ineligible to be deployed. The workload will be constrained from being deployed in the selected AZ.

**NOTE**

This method can be used to add, edit, or delete scheduling policies.

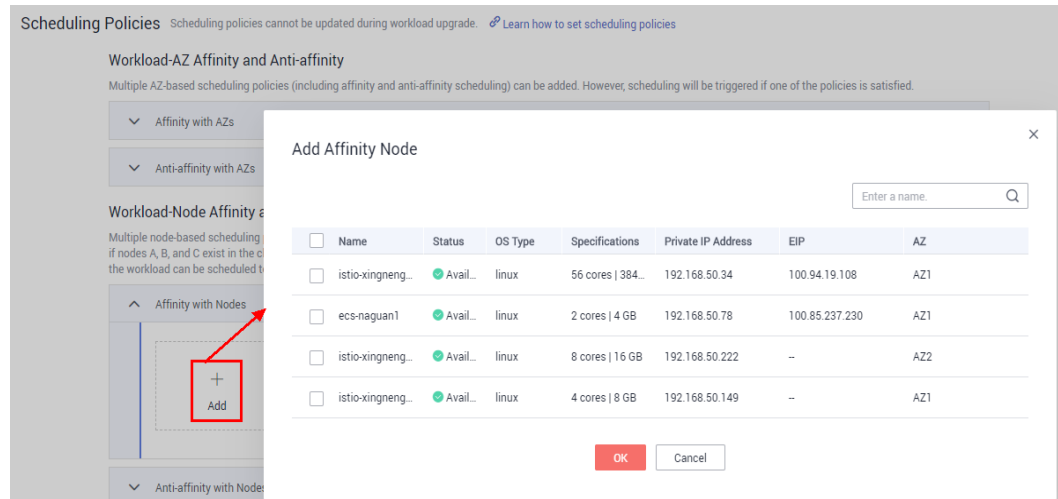
----End

### 17.7.3.3 Workload-Node Affinity

#### Using the CCE Console

**Step 1** When [Creating a Deployment](#) or [Creating a StatefulSet](#), in the **Scheduling Policies** area on the **Configure Advanced Settings** page, choose **Workload-Node Affinity and Anti-affinity** > **Affinity with Nodes** > **Add**.

**Figure 17-101** Affinity with nodes



**Step 2** Select the node on which you want to deploy the workload, and click **OK**.

If you select multiple nodes, the system automatically chooses one of them during workload deployment.

----End

#### Using kubectl

This section uses an Nginx workload as an example to describe how to create a workload using kubectl.

##### Prerequisites

The ECS where the kubectl client runs has been connected to your cluster. For details, see [Connecting to a Cluster Using kubectl](#).

##### Procedure

When [using kubectl to create a Deployment](#) or [using kubectl to create a StatefulSet](#), configure workload-node affinity. The following is an example YAML file for workload-node affinity.

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx
spec:
  replicas: 1
  selector:
    matchLabels:
      app: nginx
```



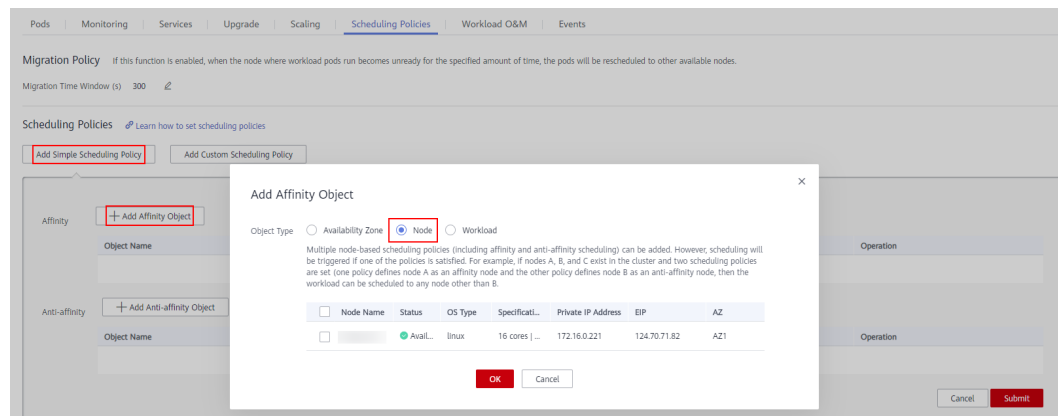
```

strategy:
  type: RollingUpdate
template:
  metadata:
    labels:
      app: nginx
  spec:
    containers:
      - image: nginx
        imagePullPolicy: Always
        name: nginx
    imagePullSecrets:
      - name: default-secret
    affinity:
      nodeAffinity:
        requiredDuringSchedulingIgnoredDuringExecution:
          nodeSelectorTerms:
            - matchExpressions:
                - key: nodeName          #node's label key
                  operator: In
                  values:
                    - test-node-1        #node's label value
    
```

## Setting the Object Type After Creating a Workload

- Step 1** Log in to the CCE console and choose **Workloads > Deployments** or **Workloads > StatefulSets** in the navigation pane.
- Step 2** Click the name of the workload for which you will add a scheduling policy. On the workload details page, choose **Scheduling Policies > Add Simple Scheduling Policy > Add Affinity Object**.
- Step 3** Set **Object Type** to **Node** and select the node where the workload is to be deployed. The workload will be deployed on the selected node.

**Figure 17-102** Adding an affinity object - Node



**NOTE**

This method can be used to add, edit, or delete scheduling policies.

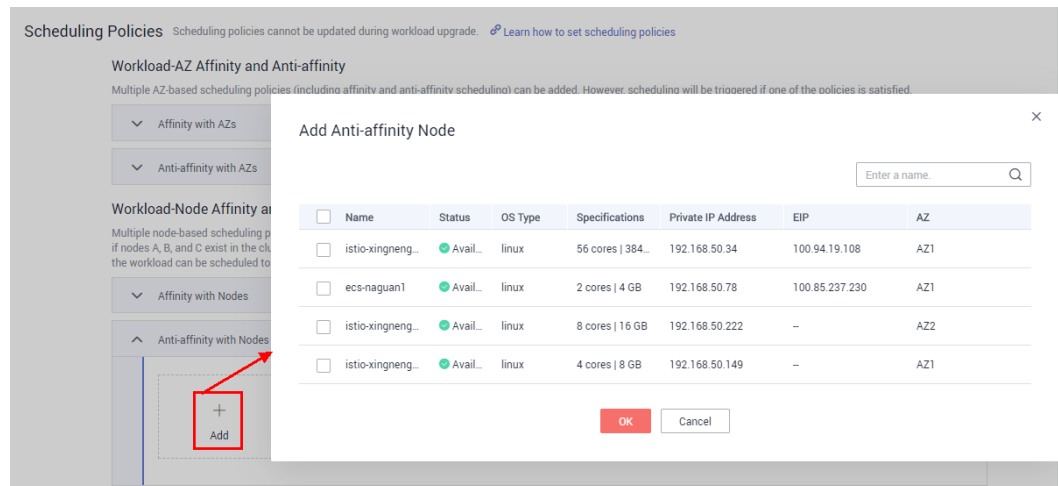
----End

### 17.7.3.4 Workload-Node Anti-Affinity

#### Using the CCE Console

- Step 1** When [Creating a Deployment](#) or [Creating a StatefulSet](#), in the **Scheduling Policies** area on the **Configure Advanced Settings** page, choose **Workload-Node Affinity and Anti-affinity** > **Anti-affinity with Nodes** > **Add**.

**Figure 17-103** Anti-affinity with nodes



- Step 2** Select the node on which the workload is ineligible to be deployed, and click **OK**. If you select multiple nodes, the workload will not be deployed on these nodes.

----End

#### Using kubectl

This section uses Nginx as an example to describe how to create a workload using kubectl.

##### Prerequisites

The ECS where the kubectl client runs has been connected to your cluster. For details, see [Connecting to a Cluster Using kubectl](#).

##### Procedure

When [using kubectl to create a Deployment](#) or [using kubectl to create a StatefulSet](#), configure workload-node affinity. The following is an example YAML file for workload-node anti-affinity.

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx
spec:
  replicas: 1
  selector:
    matchLabels:
      app: nginx
  strategy:
    type: RollingUpdate
```

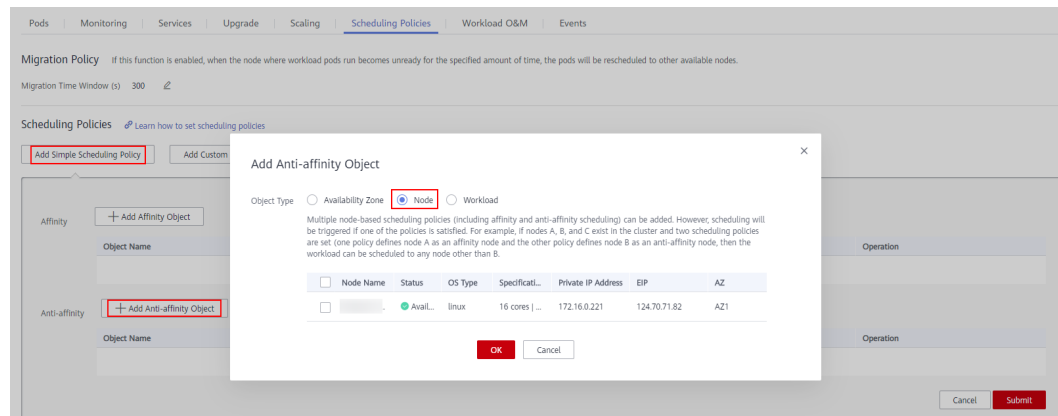
```

template:
  metadata:
    labels:
      app: nginx
  spec:
    containers:
      - image: nginx
        imagePullPolicy: Always
        name: nginx
    imagePullSecrets:
      - name: default-secret
    affinity:
      nodeAffinity:
        requiredDuringSchedulingIgnoredDuringExecution:
          nodeSelectorTerms:
            - matchExpressions:
              - key: nodeName          #node's label key
                operator: NotIn       #Indicates that the workload will not be deployed on the node.
                values:
              - test-node-1          #node's label value
    
```

## Setting the Object Type After Creating a Workload

- Step 1** Log in to the CCE console and choose **Workloads > Deployments** or **Workloads > StatefulSets** in the navigation pane.
- Step 2** Click the name of the workload for which you will add a scheduling policy. On the workload details page, choose **Scheduling Policies > Add Simple Scheduling Policy > Add Anti-affinity Object**.
- Step 3** Set **Object Type** to **Node** and select the node on which the workload is ineligible to be deployed. The workload will be constrained from being deployed on the selected node.

**Figure 17-104** Adding an anti-affinity object - Node



**NOTE**

This method can be used to add, edit, or delete scheduling policies.

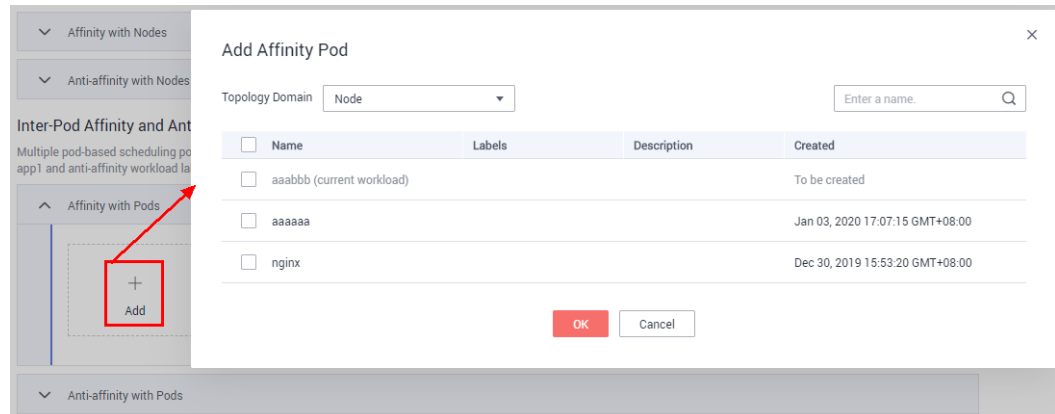
----End

### 17.7.3.5 Workload-Workload Affinity

#### Using the CCE Console

- Step 1** When [Creating a Deployment](#) or [Creating a StatefulSet](#), in the **Scheduling Policies** area on the **Configure Advanced Settings** page, choose **Inter-Pod Affinity and Anti-affinity** > **Affinity with Pods** > **Add**.

**Figure 17-105** Affinity between pods



- Step 2** Select the workloads that will be co-located with the current workload on the same node, and click **OK**.

The workload to be created will be deployed on the same node as the selected affinity workloads.

----End

#### Using kubectl

This section uses Nginx as an example to describe how to create a workload using kubectl.

##### Prerequisites

The ECS where the kubectl client runs has been connected to your cluster. For details, see [Connecting to a Cluster Using kubectl](#).

##### Procedure

When [using kubectl to create a Deployment](#) or [using kubectl to create a StatefulSet](#), configure workload-workload affinity. The following is an example YAML file for workload-workload affinity.

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx
spec:
  replicas: 1
  selector:
    matchLabels:
      app: nginx
  strategy:
    type: RollingUpdate
```

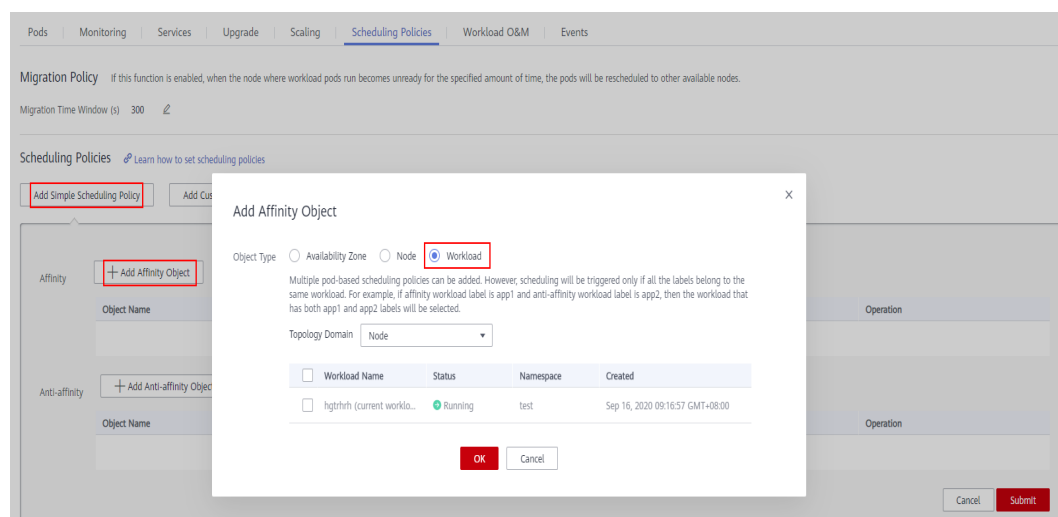
```

template:
  metadata:
    labels:
      app: nginx
  spec:
    containers:
      - image: nginx
        imagePullPolicy: Always
        name: nginx
    imagePullSecrets:
      - name: default-secret
  affinity:
    podAffinity:
      requiredDuringSchedulingIgnoredDuringExecution:
        nodeSelectorTerms:
          - matchExpressions:
              - key: app      #workload's label key
                operator: In
              values:
                - test      #workload's label value
    
```

## Setting the Object Type After Creating a Workload

- Step 1** Log in to the CCE console and choose **Workloads > Deployments** or **Workloads > StatefulSets** in the navigation pane.
- Step 2** Click the name of the workload for which you will add a scheduling policy. On the workload details page, choose **Scheduling Policies > Add Simple Scheduling Policy > Add Affinity Object**.
- Step 3** Set **Object Type** to **Workload** and select the workloads to be deployed on the same node as the created workload. The created workload and the selected workloads will be deployed on the same node.

**Figure 17-106** Adding an affinity object – Workload



**NOTE**

This method can be used to add, edit, or delete scheduling policies.

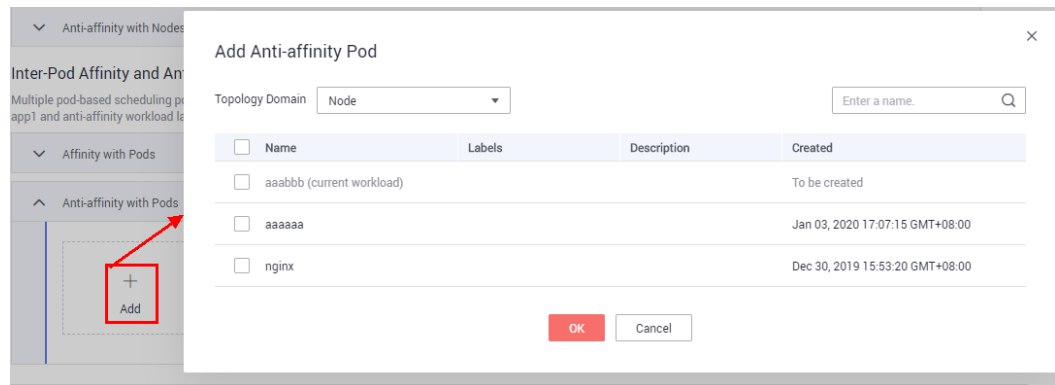
----End

### 17.7.3.6 Workload-Workload Anti-Affinity

#### Using the CCE Console

- Step 1** When [Creating a Deployment](#) or [Creating a StatefulSet](#), in the **Scheduling Policies** area on the **Configure Advanced Settings** page, choose **Inter-Pod Affinity and Anti-affinity** > **Anti-affinity with Pods** > **Add**.

**Figure 17-107** Anti-affinity between pods



- Step 2** Select the workloads to which you want to deploy the target workload on a different node, and click **OK**.

The workload to be created and the selected workloads will be deployed on different nodes.

----End

#### Using kubectl

This section uses Nginx as an example to describe how to create a workload using kubectl.

##### Prerequisites

The ECS where the kubectl client runs has been connected to your cluster. For details, see [Connecting to a Cluster Using kubectl](#).

##### Procedure

When [using kubectl to create a Deployment](#) or [using kubectl to create a StatefulSet](#), configure workload-workload anti-affinity. The following is an example YAML file for workload-workload anti-affinity.

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx
spec:
  replicas: 1
  selector:
    matchLabels:
      app: nginx
  strategy:
    type: RollingUpdate
  template:
```

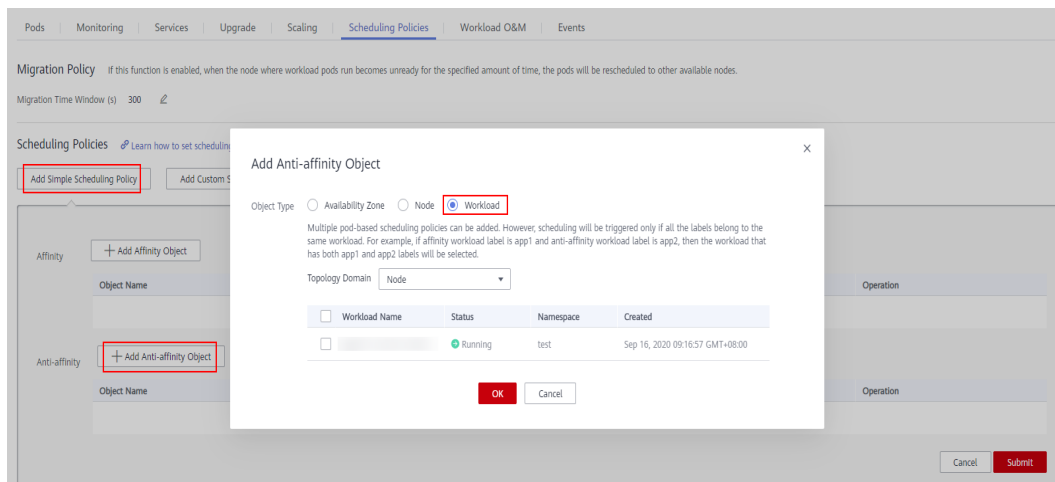
```

metadata:
  labels:
    app: nginx
spec:
  containers:
  - image: nginx
    imagePullPolicy: Always
    name: nginx
  imagePullSecrets:
  - name: default-secret
  affinity:
    podAffinity:
      requiredDuringSchedulingIgnoredDuringExecution:
        nodeSelectorTerms:
        - matchExpressions:
          - key: app      #workload's label key
            operator: NotIn
              values:
            - test      #workload's label value
  
```

## Setting the Object Type After Creating a Workload

- Step 1** Log in to the CCE console and choose **Workloads > Deployments** or **Workloads > StatefulSets** in the navigation pane.
- Step 2** Click the name of the workload for which you will add a scheduling policy. On the workload details page, choose **Scheduling Policies > Add Simple Scheduling Policy > Add Anti-affinity Object**.
- Step 3** Set **Object Type** to **Workload** and select the workloads to be deployed on a different node from the created workload. The created workload and the selected workloads will be deployed on different nodes.

**Figure 17-108** Adding an anti-affinity object – Workload



**NOTE**

This method can be used to add, edit, or delete scheduling policies.

-----End

## 17.8 Networking

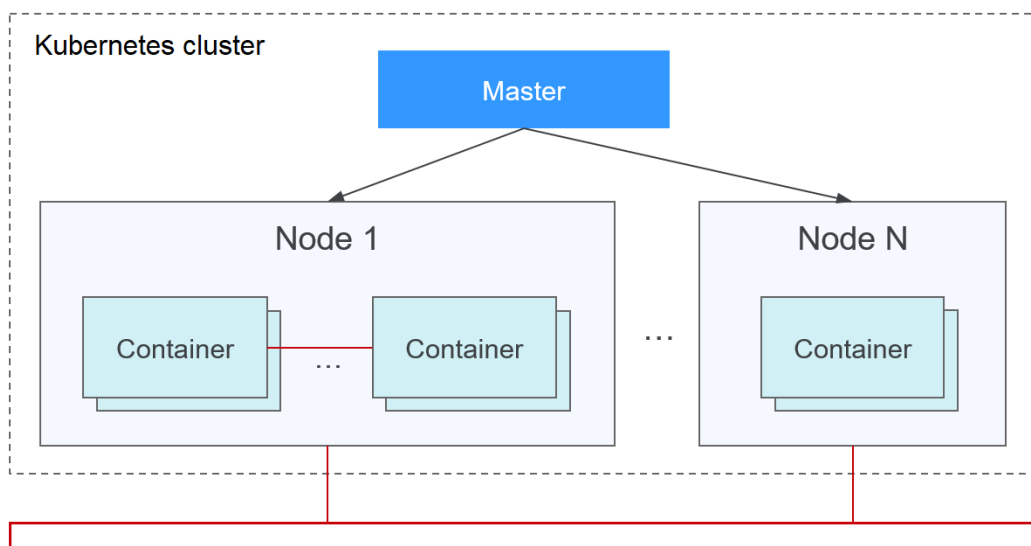
## 17.8.1 Overview

You can learn about a cluster network from the following two aspects:

- What is a cluster network like? A cluster consists of multiple nodes, and pods (or containers) are running on the nodes. Nodes and containers need to communicate with each other. For details about the cluster network types and their functions, see [Cluster Network Structure](#).
- How is pod access implemented in a cluster? Accessing a pod or container is a process of accessing services of a user. Kubernetes provides [Service](#) and [Ingress](#) to address pod access issues. This section summarizes common network access scenarios. You can select the proper scenario based on site requirements. For details about the network access scenarios, see [Access Scenarios](#).

### Cluster Network Structure

All nodes in the cluster are located in a VPC and use the VPC network. The container network is managed by dedicated network add-ons.



- **Node Network**  
A node network assigns IP addresses to hosts (nodes in the figure above) in a cluster. You need to select a VPC subnet as the node network of the CCE cluster. The number of available IP addresses in a subnet determines the maximum number of nodes (including master nodes and worker nodes) that can be created in a cluster. This quantity is also affected by the container network. For details, see the container network model.
- **Container Network**  
A container network assigns IP addresses to containers in a cluster. CCE inherits the IP-Per-Pod-Per-Network network model of Kubernetes. That is, each pod has an independent IP address on a network plane and all containers in a pod share the same network namespace. All pods in a cluster exist in a directly connected flat network. They can access each other through their IP addresses without using NAT. Kubernetes only provides a network mechanism for pods, but does not directly configure pod networks. The



configuration of pod networks is implemented by specific container network add-ons. The container network add-ons are responsible for configuring networks for pods and managing container IP addresses.

Currently, CCE supports the following container network models:

- Container tunnel network: The container tunnel network is constructed on but independent of the node network through tunnel encapsulation. This network model uses VXLAN to encapsulate Ethernet packets into UDP packets and transmits them in tunnels. Open vSwitch serves as the backend virtual switch.
- VPC network: The VPC network uses VPC routing to integrate with the underlying network. This network model is applicable to performance-intensive 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. This networking model is free from tunnel encapsulation overhead and outperforms the container tunnel network model. In addition, as VPC routing includes routes to node IP addresses and the container CIDR block, container pods in the cluster can be directly accessed from outside the cluster.

The performance, networking scale, and application scenarios of a container network vary according to the container network model. For details about the functions and features of different container network models, see [Overview](#).

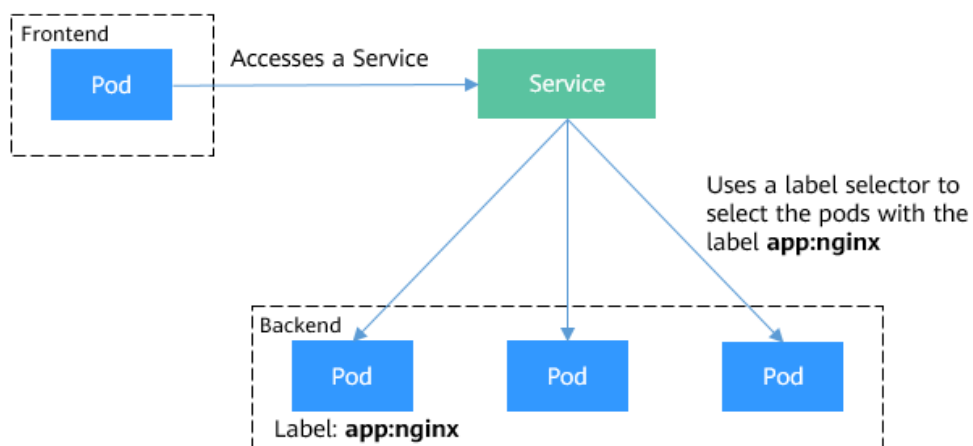
- **Service Network**

Service is also a Kubernetes object. Each Service has a fixed IP address. When creating a cluster on CCE, you can specify the Service CIDR block. The Service CIDR block cannot overlap with the node or container CIDR block. The Service CIDR block can be used only within a cluster.

## Service

A Service is used for pod access. With a fixed IP address, a Service forwards access traffic to pods and performs load balancing for these pods.

**Figure 17-109** Accessing pods through a Service



You can configure the following types of Services:

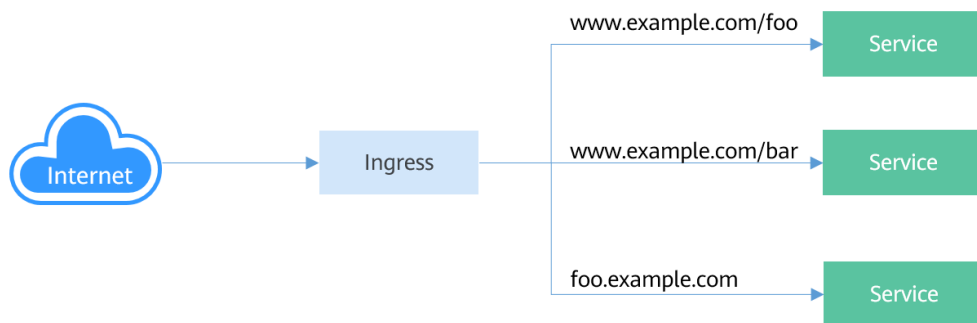
- ClusterIP: used to make the Service only reachable from within a cluster.
- NodePort: used for access from outside a cluster. A NodePort Service is accessed through the port on the node.
- LoadBalancer: used for access from outside a cluster. It is an extension of NodePort, to which a load balancer routes, and external systems only need to access the load balancer.

For details about the Service, see [Overview](#).

## Ingress

Services forward requests using layer-4 TCP and UDP protocols. Ingresses forward requests using layer-7 HTTP and HTTPS protocols. Domain names and paths can be used to achieve finer granularities.

**Figure 17-110** Ingress and Service



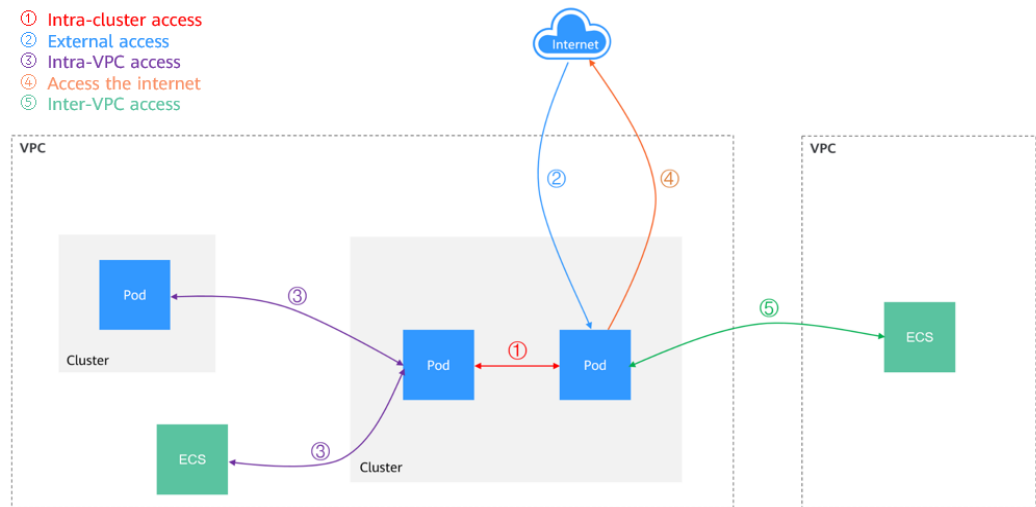
For details about the ingress, see [Overview](#).

## Access Scenarios

Workload access scenarios can be categorized as follows:

- Intra-cluster access: A ClusterIP Service is used for workloads in the same cluster to access each other.
- Access from outside a cluster: A Service (NodePort or LoadBalancer type) or an ingress is recommended for a workload outside a cluster to access workloads in the cluster.
  - Access through the internet requires an EIP to be bound the node or load balancer.
  - Access through an intranet uses only the intranet IP address of the node or load balancer. If workloads are located in different VPCs, a peering connection is required to enable communication between different VPCs.
- External access initiated by a workload:
  - Accessing an intranet: The workload accesses the intranet address, but the implementation method varies depending on container network models. Ensure that the peer security group allows the access requests from the container CIDR block.
  - Accessing a public network: You need to assign an EIP to the node where the workload runs, or configure SNAT rules through the NAT gateway.

**Figure 17-111** Network access diagram



## 17.8.2 Container Network Models

### 17.8.2.1 Overview

The container network assigns IP addresses to pods in a cluster and provides networking services. In CCE, you can select the following network models for your cluster:

- Container tunnel network
- VPC network

### Network Model Comparison

**Table 17-70** describes the differences of network models supported by CCE.

**⚠ CAUTION**

After a cluster is created, the network model cannot be changed.

**Table 17-70** Network model comparison

Dimension	Tunnel Network	VPC Network
Core technology	OVS	IPVlan and VPC route
Applicable clusters	CCE cluster	CCE cluster
Network isolation	Yes. For details, see <a href="#">Configuring Network Policies to Restrict Pod Access</a> .	No

Dimension	Tunnel Network	VPC Network
Passthrough networking	No	No
IP address management	<ul style="list-style-type: none"> <li>The container CIDR block is allocated separately.</li> <li>CIDR blocks are divided by node and can be dynamically allocated (CIDR blocks can be dynamically added after being allocated.)</li> </ul>	<ul style="list-style-type: none"> <li>The container CIDR block is allocated separately.</li> <li>CIDR blocks are divided by node and statically allocated (the CIDR block cannot be changed after a node is created).</li> </ul>
Network performance	Performance loss due to VXLAN encapsulation	No tunnel encapsulation. Cross-node packets are forwarded through VPC routers, delivering performance equivalent to that of the host network.
Networking scale	A maximum of 2,000 nodes are supported.	By default, 200 nodes are supported. Each time a node is added to the cluster, a route is added to the VPC routing table. Therefore, the cluster scale is limited by the VPC route table.
Application scenarios	<ul style="list-style-type: none"> <li>Common container service scenarios</li> <li>Scenarios that do not have high requirements on network latency and bandwidth</li> </ul>	<ul style="list-style-type: none"> <li>Scenarios that have high requirements on network latency and bandwidth</li> <li>Containers can communicate with VMs using a microservice registration framework, such as Dubbo and CSE.</li> </ul>

#### NOTICE

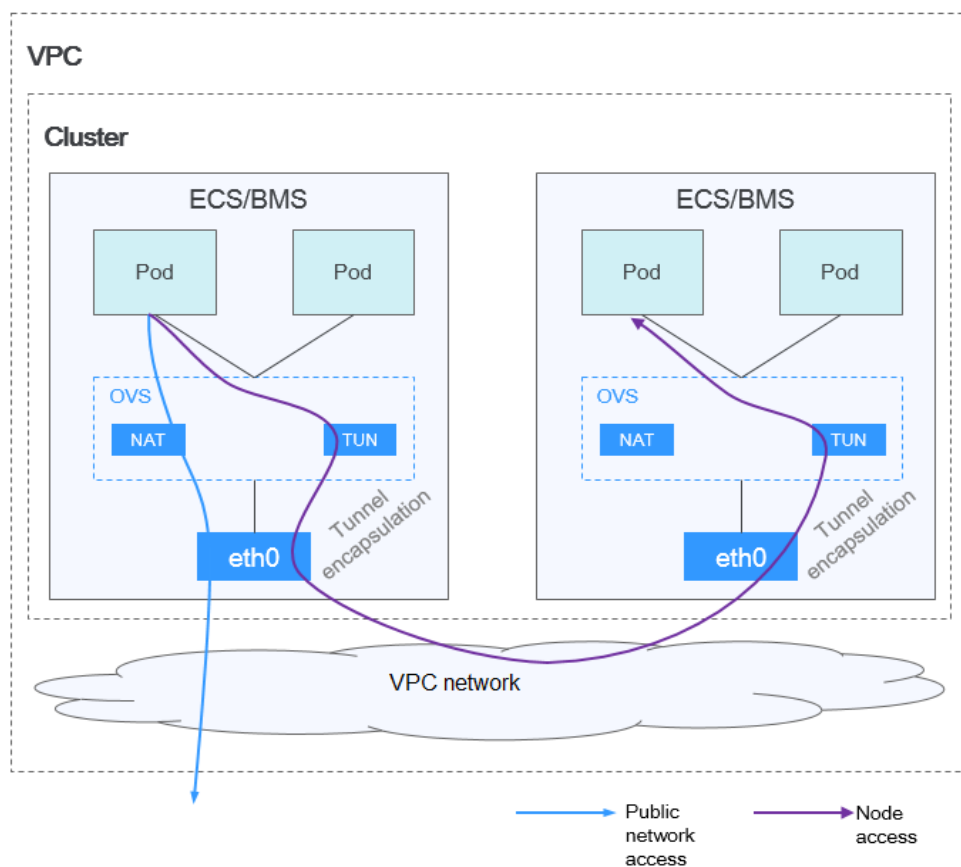
- The scale of a cluster that uses the VPC network model is limited by the custom routes of the VPC. Therefore, you need to estimate the number of required nodes before creating a cluster.
- By default, VPC routing network supports direct communication between containers and hosts in the same VPC. If 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.

## 17.8.2.2 Container Tunnel Network

### Container Tunnel Network Model

The container tunnel network is constructed on but independent of the node network through tunnel encapsulation. This network model uses VXLAN to encapsulate Ethernet packets into UDP packets and transmits them in tunnels. Open vSwitch serves as the backend virtual switch. Though at some costs of performance, packet encapsulation and tunnel transmission enable higher interoperability and compatibility with advanced features (such as network policy-based isolation) for most common scenarios.

Figure 17-112 Container tunnel network



#### Pod-to-pod communication

- On the same node: Packets are directly forwarded via the OVS bridge on the node.
- Across nodes: Packets are encapsulated in the OVS bridge and then forwarded to the peer node.

### Advantages and Disadvantages

#### Advantages

- The container network is decoupled from the node network and is not limited by the VPC quotas and response speed (such as the number of VPC routes, number of elastic ENIs, and creation speed).
- Network isolation is supported. For details, see [Configuring Network Policies to Restrict Pod Access](#).
- Bandwidth limits are supported.
- Large-scale networking is supported.

#### Disadvantages

- High encapsulation overhead, complex networking, and low performance
- Failure to use the load balancing and security group capabilities provided by the VPC
- External networks cannot be directly connected to container IP addresses.

### Applicable Scenarios

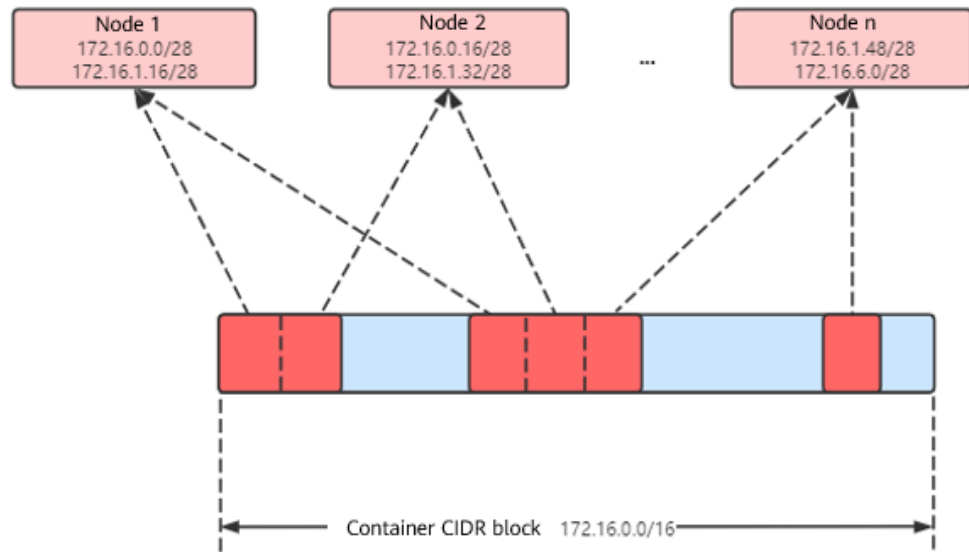
- Low requirements on performance: As the container tunnel network requires additional VXLAN tunnel encapsulation, it has about 5% to 15% of performance loss when compared with the other two container network models. Therefore, the container tunnel network is applicable to the scenarios that do not have 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 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.

### Container IP Address Management

The container tunnel network allocates container IP addresses according to the following rules:

- The container CIDR block is allocated separately, which is irrelevant to the node CIDR block.
- IP addresses are allocated by node. One or more CIDR blocks with a fixed size (16 by default) are allocated to each node in a cluster from the container CIDR block.
- When the IP addresses on a node are used up, you can apply for a new CIDR block.
- The container CIDR block cyclically allocates CIDR blocks to new nodes or existing nodes in sequence.
- Pods scheduled to a node are cyclically allocated IP addresses from one or more CIDR blocks allocated to the node.

**Figure 17-113** IP address allocation of the container tunnel network



Maximum number of nodes that can be created in the cluster using the container tunnel network = Number of IP addresses in the container CIDR block / Size of the IP CIDR block allocated to the node by the container CIDR block at a time (16 by default)

For example, if the container CIDR block is 172.16.0.0/16, the number of IP addresses is 65536. If 16 IP addresses are allocated to a node at a time, a maximum of 4096 (65536/16) nodes can be created in the cluster. This is an extreme case. If 4096 nodes are created, a maximum of 16 pods can be created for each node because only 16 IP CIDR block\’s are allocated to each node. In addition, the number of nodes that can be created in a cluster also depends on the node network and cluster scale.

## Recommendation for CIDR Block Planning

As described in [Cluster Network Structure](#), network addresses in a cluster can be divided into three parts: node network, container network, and service network. When planning network addresses, consider the following aspects:

- The three CIDR blocks cannot overlap. Otherwise, a conflict occurs.
- Ensure that each CIDR block has sufficient IP addresses.
  - The IP addresses in the node CIDR block must match the cluster scale. Otherwise, nodes cannot be created due to insufficient IP addresses.
  - The IP addresses in the container CIDR block must match the service scale. Otherwise, pods cannot be created due to insufficient IP addresses. The number of pods that can be created on each node also depends on other parameter settings.

In the following configuration, the cluster has 200 nodes, and the network model is the container tunnel network.

In this case, the number of available IP addresses in the selected node subnet must be greater than 200. Otherwise, nodes cannot be created due to insufficient IP addresses.

The container CIDR block is 10.0.0.0/16, and the number of available IP addresses is 65533. These IP addresses can be allocated to a maximum of 4096 nodes. (16 IP addresses are allocated to each node at a time. For details, see [Container IP Address Management](#).)

## Example of Container Tunnel Network Access

Create a cluster that uses the container tunnel network model.

Create a Deployment on the cluster.

```
kind: Deployment
apiVersion: apps/v1
metadata:
  name: example
  namespace: default
spec:
  replicas: 4
  selector:
    matchLabels:
      app: example
  template:
    metadata:
      labels:
        app: example
    spec:
      containers:
        - name: container-0
          image: 'nginx:perl'
          resources:
            limits:
              cpu: 250m
              memory: 512Mi
            requests:
              cpu: 250m
              memory: 512Mi
      imagePullSecrets:
        - name: default-secret
```

View the created pod.

```
$ kubectl get pod -owide
NAME                READY  STATUS   RESTARTS  AGE  IP        NODE          NOMINATED NODE
READINESS GATES
example-5bdc5699b7-5rvq4  1/1    Running  0         3m28s  10.0.0.20  192.168.0.42  <none>
example-5bdc5699b7-984j9  1/1    Running  0         3m28s  10.0.0.21  192.168.0.42  <none>
example-5bdc5699b7-lfxkm  1/1    Running  0         3m28s  10.0.0.22  192.168.0.42  <none>
example-5bdc5699b7-wjcmg  1/1    Running  0         3m28s  10.0.0.52  192.168.0.64  <none>
```

In this case, the IP address of the pod cannot be directly accessed outside the cluster in the same VPC. This is a feature of the container tunnel network.

However, the pod can be accessed from a node in the cluster or in the pod. As shown in the following figure, the pod can be accessed directly from the container.

```
$ kubectl exec -it example-5bdc5699b7-5rvq4 -- curl 10.0.0.21
<!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>
<p>If you see this page, the nginx web server is successfully installed and
working. Further configuration is required.</p>

<p>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>.</p>

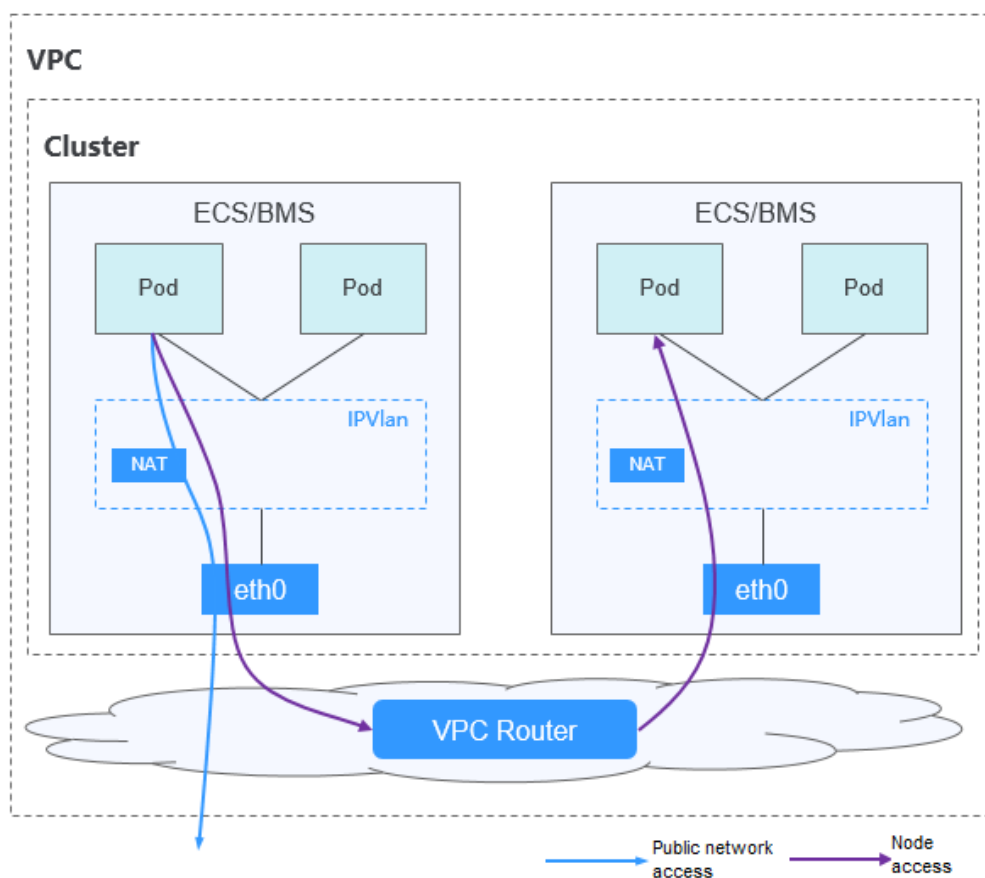
<p><em>Thank you for using nginx.</em></p>
</body>
</html>
```

### 17.8.2.3 VPC Network

#### Model Definition

The VPC network uses VPC routing to integrate with the underlying network. This network model is suitable for performance-intensive scenarios. The maximum number of nodes allowed in a cluster depends on the VPC route quota. Each node is assigned a CIDR block of a fixed size. This networking model is free from tunnel encapsulation overhead and outperforms the container tunnel network model. In addition, as VPC routing includes routes to node IP addresses and the container CIDR block, container pods in a cluster can be directly accessed from outside the cluster.

**Figure 17-114** VPC network model



### Pod-to-pod communication

- On the same node: Packets are directly forwarded through IPvlan.
- Across nodes: Packets are forwarded to the default gateway through default routes, and then to the peer node via the VPC routes.

## Advantages and Disadvantages

### Advantages

- No tunnel encapsulation is required, so network problems are easy to locate and the performance is high.
- External networks in a VPC can be directly connected to container IP addresses.

### Disadvantages

- The number of nodes is limited by the VPC route quota.
- Each node is assigned a CIDR block of a fixed size, which leads to a waste of IP addresses in the container CIDR block.
- Pods cannot directly use functionalities such as EIPs and security groups.

## Applicable Scenarios

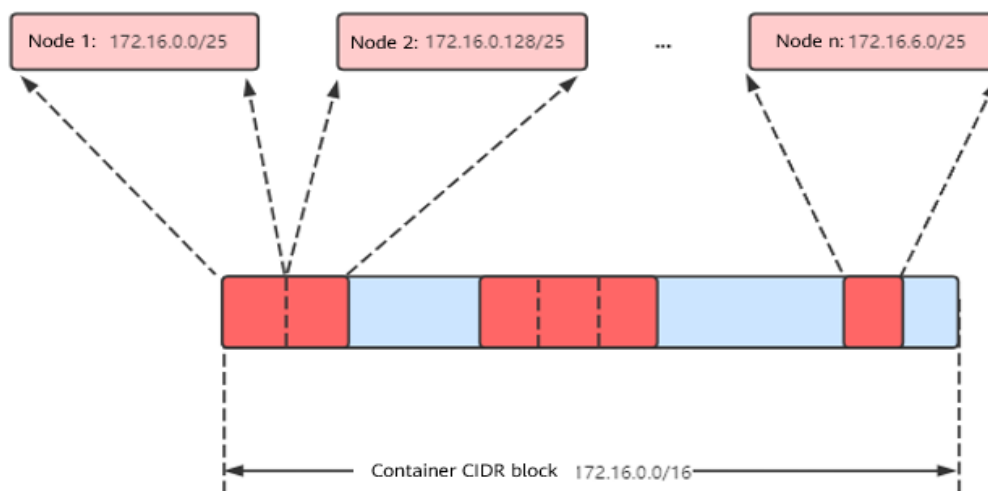
- High performance requirements: As no tunnel encapsulation is required, the VPC network model delivers the performance close to that of a VPC network when compared with the container tunnel network model. Therefore, the VPC network model is applicable to scenarios that have high requirements on performance, such as AI computing and big data computing.
- Small- and medium-scale networking: The VPC network is limited by the VPC route quota. Currently, a maximum of 200 nodes are supported by default. If there are large-scale networking requirements, you can increase the VPC route quota.

## Container IP Address Management

The VPC network allocates container IP addresses according to the following rules:

- The container CIDR block is allocated separately.
- IP addresses are allocated by node. One CIDR block with a fixed size (which is configurable) is allocated to each node in a cluster from the container CIDR block.
- The container CIDR block cyclically allocates CIDR blocks to new nodes in sequence.
- Pods scheduled to a node are cyclically allocated IP addresses from CIDR blocks allocated to the node.

**Figure 17-115** IP address management of the VPC network



Maximum number of nodes that can be created in the cluster using the VPC network = Number of IP addresses in the container CIDR block / Number of IP addresses in the CIDR block allocated to the node by the container CIDR block

For example, if the container CIDR block is 172.16.0.0/16, the number of IP addresses is 65536. The mask of the container CIDR block allocated to the node is 25. That is, the number of container IP addresses on each node is 128. Therefore, a maximum of 512 (65536/128) nodes can be created. In addition, the number of nodes that can be created in a cluster also depends on the node network and cluster scale.

## Recommendation for CIDR Block Planning

As described in [Cluster Network Structure](#), network addresses in a cluster can be divided into three parts: node network, container network, and service network. When planning network addresses, consider the following aspects:

- The three CIDR blocks cannot overlap. Otherwise, a conflict occurs.
- Ensure that each CIDR block has sufficient IP addresses.
  - The IP addresses in the node CIDR block must match the cluster scale. Otherwise, nodes cannot be created due to insufficient IP addresses.
  - The IP addresses in the container CIDR block must match the service scale. Otherwise, pods cannot be created due to insufficient IP addresses. The number of pods that can be created on each node also depends on other parameter settings.

Assume that a cluster contains 200 nodes and the network model is VPC network.

In this case, the number of available IP addresses in the selected node subnet must be greater than 200. Otherwise, nodes cannot be created due to insufficient IP addresses.

The container CIDR block is 10.0.0.0/16, and the number of available IP addresses is 65536. As described in [Container IP Address Management](#), the VPC network is allocated a CIDR block with the fixed size (using the mask to determine the maximum number of container IP addresses allocated to each node). For example, if the upper limit is 128, the cluster supports a maximum of 512 (65536/128) nodes, including the three master nodes.

## Example of VPC Network Access

Create a cluster using the VPC network model.

**Figure 17-116** Cluster network

### Network

Network Model	VPC network
VPC	vpc-asm
Subnet	subnet-asm
Service Forwarding Mode	iptables
Service Network Segment	10.247.0.0/16
Container Network Segment	10.0.0.0/16
Internal API Server Address	https://10.10.0.245:5443
Public API Server Address	Bind EIP

The cluster contains one node.

```
$ kubectl get node
NAME          STATUS    ROLES    AGE   VERSION
192.168.0.99  Ready    <none>   9d   v1.17.17-r0-CCE21.6.1.B004-17.37.5
```

Check the VPC routing table. The destination address 172.16.0.0/25 is the container CIDR block allocated to the node, and the next hop is the corresponding node. When the container IP address is accessed, the VPC route forwards the access request to the next-hop node. This indicates that the VPC network model uses VPC routes.

Create a Deployment on the cluster.

```
kind: Deployment
apiVersion: apps/v1
metadata:
  name: example
  namespace: default
spec:
  replicas: 4
  selector:
    matchLabels:
      app: example
  template:
    metadata:
      labels:
        app: example
    spec:
      containers:
        - name: container-0
          image: 'nginx:perl'
          imagePullSecrets:
            - name: default-secret
```

Check the pod.

```
$ kubectl get pod -owide
NAME                READY  STATUS   RESTARTS  AGE  IP          NODE          NOMINATED NODE
READINESS GATES
example-86b9779494-l8qrw  1/1    Running  0         14s  172.16.0.6  192.168.0.99  <none>
example-86b9779494-svs8t  1/1    Running  0         14s  172.16.0.7  192.168.0.99  <none>
example-86b9779494-x8kl5  1/1    Running  0         14s  172.16.0.5  192.168.0.99  <none>
example-86b9779494-zt627  1/1    Running  0         14s  172.16.0.8  192.168.0.99  <none>
```

In this case, the IP address of the pod can be directly accessed from a node outside the cluster in the same VPC. This is a feature of the VPC network feature.

The pod can also be accessed from a node in the same cluster or in the pod. As shown in the following figure, the pod can be accessed directly from the container.

```
$ kubectl exec -it example-86b9779494-l8qrw -- curl 172.16.0.7
<!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>
<p>If you see this page, the nginx web server is successfully installed and working. Further configuration is required.</p>
<p>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>.</p>  
<p><em>Thank you for using nginx.</em></p>  
</body>  
</html>
```

## 17.8.3 Services

### 17.8.3.1 Overview

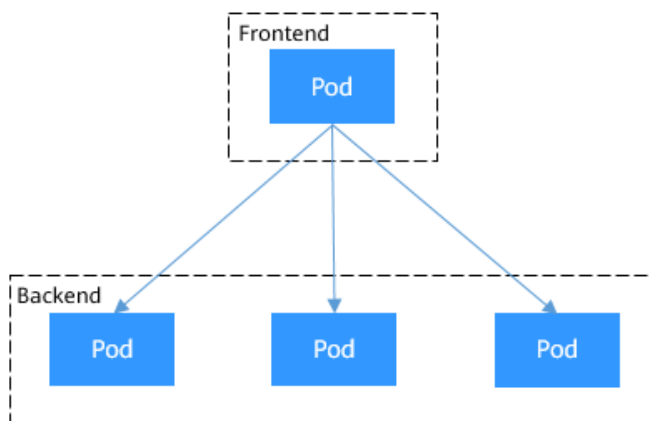
#### Direct Access to a Pod

After a pod is created, the following problems may occur if you directly access the pod:

- The pod can be deleted and recreated at any time by a controller such as a Deployment, and the result of accessing the pod becomes unpredictable.
- The IP address of the pod is allocated only after the pod is started. Before the pod is started, the IP address of the pod is unknown.
- An application is usually composed of multiple pods that run the same image. Accessing pods one by one is not efficient.

For example, an application uses Deployments to create the frontend and backend. The frontend calls the backend for computing, as shown in [Figure 17-117](#). Three pods are running in the backend, which are independent and replaceable. When a backend pod is re-created, the new pod is assigned with a new IP address, of which the frontend pod is unaware.

Figure 17-117 Inter-pod access

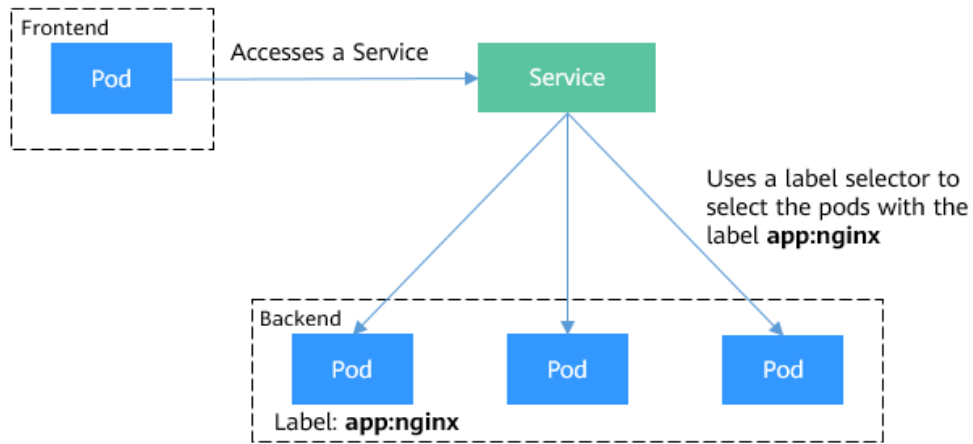


#### Using Services for Pod Access

Kubernetes Services are used to solve the preceding pod access problems. A Service has a fixed IP address. (When a CCE cluster is created, a Service CIDR block is set, which is used to allocate IP addresses to Services.) A Service forwards requests accessing the Service to pods based on labels, and at the same time, perform load balancing for these pods.

In the preceding example, a Service is added for the frontend pod to access the backend pods. In this way, the frontend pod does not need to be aware of the changes on backend pods, as shown in [Figure 17-118](#).

**Figure 17-118** Accessing pods through a Service



## Service Types

Kubernetes allows you to specify a Service of a required type. The values and actions of different types of Services are as follows:

- **ClusterIP**  
A ClusterIP Service allows workloads in the same cluster to use their cluster-internal domain names to access each other.
- **NodePort**  
A NodePort Service is exposed on each node's IP at a static port. A ClusterIP Service, to which the NodePort Service routes, is automatically created. By requesting `<NodeIP>:<NodePort>`, you can access a NodePort Service from outside the cluster.
- **LoadBalancer**  
A workload can be accessed from public networks through a load balancer, which is more secure and reliable than EIP.

### 17.8.3.2 Intra-Cluster Access (ClusterIP)

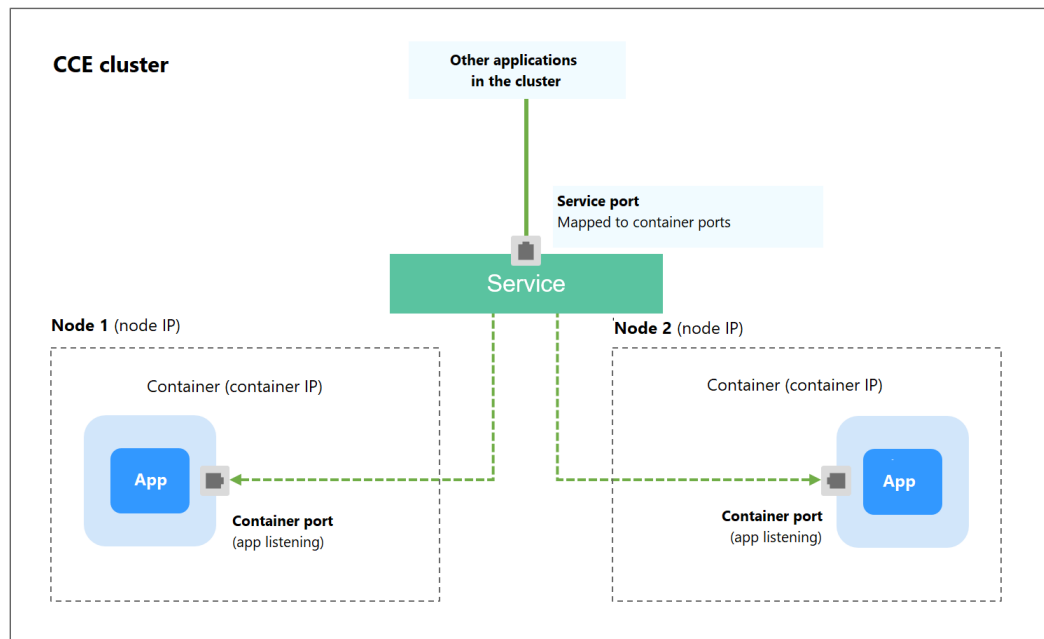
#### Scenario

ClusterIP Services allow workloads in the same cluster to use their cluster-internal domain names to access each other.

The cluster-internal domain name format is `<Service name>.<Namespace of the workload>.svc.cluster.local:<Port>`, for example, `nginx.default.svc.cluster.local:80`.

[Figure 17-119](#) shows the mapping relationships between access channels, container ports, and access ports.

**Figure 17-119** Intra-cluster access (ClusterIP)



## Adding a Service When Creating a Workload

You can set the access type (Service) when creating a workload on the CCE console.

**Step 1** In the **Set Application Access** step of **Creating a Deployment**, **Creating a StatefulSet**, or **Creating a DaemonSet**, click **Add Service** and set the following parameters:

- **Access Type:** Select **ClusterIP**.
- **Service Name:** Specify a Service name, which can be the same as the workload name.
- **Port Settings**
  - **Protocol:** protocol used by the Service.
  - **Container Port:** port on which the workload listens. The Nginx application listens on port 80.
  - **Access Port:** a port mapped to the container port at the cluster-internal IP address. The workload can be accessed at <cluster-internal IP address>:<access port>. The port number range is 1–65535.



**Figure 17-120** Adding a ClusterIP Service

Add Service
✕

Access Type:

---

Service Name:

---

Port Settings

Protocol	Container Port	Access Port	Operation
TCP	Range: 1-65535	Range: 1-65535	Delete

**Step 2** After the configuration, click **OK** and then **Next: Configure Advanced Settings**. On the page displayed, click **Create**.

**Step 3** Click **View Deployment Details** or **View StatefulSet Details**. On the **Services** tab page, obtain the access address, for example, 10.247.74.100:8080.

----End

## Adding a Service After Creating a Workload

You can set the Service after creating a workload. This has no impact on the workload status and takes effect immediately. The procedure is as follows:

**Step 1** Log in to the CCE console. In the navigation pane, choose **Workloads > Deployments**. On the workload list, click the name of the workload for which you will create a Service.

**Step 2** On the **Services** tab page, click **Add Service**.

**Step 3** On the **Create Service** page, select **ClusterIP** from the **Access Type** drop-down list.

**Step 4** Set intra-cluster access parameters.

- **Service Name:** Service name, which can be the same as the workload name.
- **Cluster Name:** name of the cluster where the workload runs. The value is inherited from the workload creation page and cannot be changed.
- **Namespace:** namespace where the workload is located. The value is inherited from the workload creation page and cannot be changed.
- **Workload:** workload for which you want to add a Service.
- **Port Settings**
  - **Protocol:** protocol used by the Service.
  - **Container Port:** port on which the workload listens. The Nginx application listens on port 80.
  - **Access Port:** port mapped to the container port at the cluster-internal IP address. The workload can be accessed at <cluster-internal IP address>:<access port>. The port number range is 1–65535.

**Step 5** Click **Create**. The ClusterIP Service will be added for the workload.

----End

## Setting the Access Type Using kubectl

You can run kubectl commands to set the access type (Service). This section uses a Nginx workload as an example to describe how to implement intra-cluster access using kubectl.

**Step 1** Use kubectl to connect to the cluster. For details, see [Connecting to a Cluster Using kubectl](#).

**Step 2** Create and edit the `nginx-deployment.yaml` and `nginx-clusterip-svc.yaml` files.

The file names are user-defined. `nginx-deployment.yaml` and `nginx-clusterip-svc.yaml` are merely example file names.

### vi nginx-deployment.yaml

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx
spec:
  replicas: 1
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
        - image: nginx:latest
          name: nginx
          imagePullSecrets:
            - name: default-secret
```

### vi nginx-clusterip-svc.yaml

```
apiVersion: v1
kind: Service
metadata:
  labels:
    app: nginx
  name: nginx-clusterip
spec:
  ports:
    - name: service0
      port: 8080          # 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 a Service to access the target container. This port is closely related
                        # to the applications running in a 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. In this example, select the pod with the app:nginx label.
    app: nginx
  type: ClusterIP      # Type of a Service. ClusterIP indicates that a Service is only reachable from within
                        # the cluster.
```

**Step 3** Create a workload.

```
kubectl create -f nginx-deployment.yaml
```

If information similar to the following is displayed, the workload has been created.

```
deployment "nginx" created
```

### kubectl get po

If information similar to the following is displayed, the workload is running.

```
NAME                READY   STATUS    RESTARTS   AGE
nginx-2601814895-znhbr 1/1     Running   0           15s
```

#### Step 4 Create a Service.

### kubectl create -f nginx-clusterip-svc.yaml

If information similar to the following is displayed, the Service is being created.

```
service "nginx-clusterip" created
```

### kubectl get svc

If information similar to the following is displayed, the Service has been created, and a cluster-internal IP address has been assigned to the Service.

```
# kubectl get svc
NAME                TYPE        CLUSTER-IP   EXTERNAL-IP  PORT(S)   AGE
kubernetes          ClusterIP   10.247.0.1    <none>        443/TCP   4d6h
nginx-clusterip     ClusterIP   10.247.74.52 <none>        8080/TCP  14m
```

#### Step 5 Access a Service.

A Service can be accessed from containers or nodes in a cluster.

Create a pod, access the pod, and run the **curl** command to access *IP address:Port* or the domain name of the Service, as shown in the following figure.

The domain name suffix can be omitted. In the same namespace, you can directly use **nginx-clusterip:8080** for access. In other namespaces, you can use **nginx-clusterip.default:8080** for access.

```
# kubectl run -i --tty --image nginx:alpine test --rm /bin/sh
If you don't see a command prompt, try pressing enter.
/ # curl 10.247.74.52:8080
<!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>
<p>If you see this page, the nginx web server is successfully installed and
working. Further configuration is required.</p>

<p>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>.</p>

<p><em>Thank you for using nginx.</em></p>
</body>
</html>
/ # curl nginx-clusterip.default.svc.cluster.local:8080
...
<h1>Welcome to nginx!</h1>
...
```

```

/ # curl nginx-clusterip.default:8080
...
<h1>Welcome to nginx!</h1>
...
/ # curl nginx-clusterip:8080
...
<h1>Welcome to nginx!</h1>
...

```

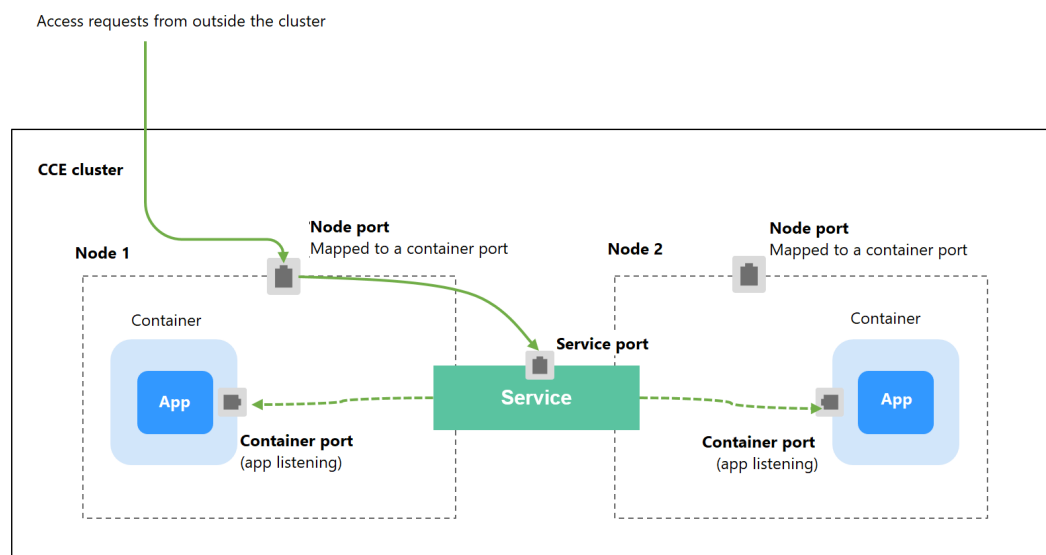
----End

### 17.8.3.3 NodePort

#### Scenario

A Service is exposed on each node's IP address at a static port (NodePort). A ClusterIP Service, to which the NodePort Service will route, is automatically created. By requesting <NodeIP>:<NodePort>, you can access a NodePort Service from outside the cluster.

Figure 17-121 NodePort access



#### Notes and Constraints

- By default, a NodePort Service is accessed within a VPC. If you need to use an EIP to access a NodePort Service through public networks, bind an EIP to the node in the cluster in advance.
- After a Service is created, if the **affinity setting** is switched from the cluster level to the node level, the connection tracing table will not be cleared. You are advised not to modify the Service affinity setting after the Service is created. If you need to modify it, create a Service again.
- The service port of a NodePort Service created on the CCE console is the same as the configured container port.

## Adding a Service When Creating a Workload

You can set the access type when creating a workload on the CCE console. An Nginx workload is used as an example.

**Step 1** In the **Set Application Access** step of [Creating a Deployment](#), [Creating a StatefulSet](#), or [Creating a DaemonSet](#), click **Add Service** and set the following parameters:

- **Access Type:** Select **NodePort**.

 **NOTE**

If you want to use an EIP to access a NodePort Service through public networks, bind an EIP to the node in the cluster in advance.

- **Service Name:** Specify a Service name, which can be the same as the workload name.
- **Service Affinity:** For details, see [externalTrafficPolicy \(Service Affinity\)](#).
  - Cluster level: The IP addresses and access ports of all nodes in a cluster can 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 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 Settings**
  - **Protocol:** protocol used by the Service.
  - **Container Port:** port on which the workload in the container image listens. The value ranges from 1 to 65535.
  - **Access Port:** node port (with a private IP address) to which the container port will be mapped. You are advised to select **Automatically generated**.
    - **Automatically generated:** The system automatically assigns a port number.
    - **Specified port:** You have to manually specify a fixed node port number in the range of 30000–32767. Ensure that the port is unique in a cluster.

**Step 2** After the configuration is complete, click **OK**.

**Step 3** Click **Next: Configure Advanced Settings**. On the page displayed, click **Create**.

**Step 4** Click **View Deployment Details** or **View StatefulSet Details**. On the **Services** tab page, obtain the access address, for example, 192.168.0.160:30358.

----End

## Adding a Service After Creating a Workload

You can set the Service after creating a workload. This has no impact on the workload status and takes effect immediately. The procedure is as follows:

**Step 1** Log in to the CCE console. In the navigation pane, choose **Workloads > Deployments**. On the workload list, click the name of the workload for which you will create a Service.

 **NOTE**

If the Service is associated with an ingress, the ingress is unavailable after the port information of the Service is updated. In this case, you need to delete and recreate the Service.

**Step 2** On the **Services** tab page, click **Add Service**.

**Step 3** On the **Create Service** page, select **NodePort** from the **Access Type** drop-down list.

 **NOTE**

If you want to use an EIP to access a NodePort Service through public networks, bind an EIP to the node in the cluster in advance.

**Step 4** Set node access parameters.

- **Service Name:** Service name, which can be the same as the workload name.
- **Cluster Name:** name of the cluster where the workload runs. The value is inherited from the workload creation page and cannot be changed.
- **Namespace:** namespace where the workload is located. The value is inherited from the workload creation page and cannot be changed.
- **Workload:** workload for which you want to add a Service. The value is inherited from the workload creation page and cannot be changed.
- **Service Affinity**
  - Cluster level: The IP addresses and access ports of all nodes in a cluster can 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 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 Settings**
  - **Protocol:** protocol used by the Service.
  - **Container Port:** port on which the workload in the container image listens. The Nginx workload listens on port 80.
  - **Access Port:** node port (with a private IP address) to which the container port will be mapped. You are advised to select **Automatically generated**.
    - **Automatically generated:** The system automatically assigns a port number.
    - **Specified port:** You have to manually specify a fixed node port number in the range of 30000–32767. Ensure that the port is unique in a cluster.

**Step 5** Click **Create**. A NodePort Service will be added for the workload.

----End

## Using kubectl

You can run kubectl commands to set the access type. This section uses a Nginx workload as an example to describe how to set a NodePort Service using kubectl.

**Step 1** Use kubectl to connect to the cluster. For details, see [Connecting to a Cluster Using kubectl](#).

**Step 2** Create and edit the `nginx-deployment.yaml` and `nginx-nodeport-svc.yaml` files.

The file names are user-defined. `nginx-deployment.yaml` and `nginx-nodeport-svc.yaml` are merely example file names.

### vi nginx-deployment.yaml

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx
spec:
  replicas: 1
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - image: nginx:latest
        name: nginx
        imagePullSecrets:
        - name: default-secret
```

### vi nginx-nodeport-svc.yaml

```
apiVersion: v1
kind: Service
metadata:
  labels:
    app: nginx
  name: nginx-nodeport
spec:
  ports:
  - name: service
    nodePort: 30000 # Node port. The value ranges from 30000 to 32767.
    port: 8080 # 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 a Service to access the target container. This port is closely related to
the applications running in a 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. In this example, select the pod with the app:nginx label.
    app: nginx
  type: NodePort # Service type. NodePort indicates that the Service is accessed through a node port.
```

**Step 3** Create a workload.

### kubectl create -f nginx-deployment.yaml

If information similar to the following is displayed, the workload has been created.

```
deployment "nginx" created
```

### kubectl get po

If information similar to the following is displayed, the workload is running.

NAME	READY	STATUS	RESTARTS	AGE
nginx-2601814895-qhxqv	1/1	Running	0	9s

#### Step 4 Create a Service.

##### **kubectl create -f nginx-nodeport-svc.yaml**

If information similar to the following is displayed, the Service is being created.

```
service "nginx-nodeport" created
```

##### **kubectl get svc**

If information similar to the following is displayed, the Service has been created.

```
# kubectl get svc
NAME          TYPE          CLUSTER-IP    EXTERNAL-IP  PORT(S)          AGE
kubernetes    ClusterIP     10.247.0.1    <none>       443/TCP          4d8h
nginx-nodeport NodePort      10.247.30.40 <none>       8080:30000/TCP  18s
```

#### Step 5 Access the Service.

By default, a NodePort Service can be accessed by using *Any node IP address:Node port*.

The Service can be accessed from a node in another cluster in the same VPC or in another pod in the cluster. If a public IP address is bound to the node, you can also use the public IP address to access the Service. Create a container in the cluster and access the container by using *Node IP address:Node port*.

```
# kubectl get node -owide
NAME          STATUS    ROLES    AGE    INTERNAL-IP  EXTERNAL-IP  OS-IMAGE          KERNEL-VERSION  CONTAINER-RUNTIME
10.100.0.136  Ready    <none>   152m   10.100.0.136 <none>       CentOS Linux 7 (Core)
3.10.0-1160.25.1.el7.x86_64 docker://18.9.0
10.100.0.5    Ready    <none>   152m   10.100.0.5   <none>       CentOS Linux 7 (Core)
3.10.0-1160.25.1.el7.x86_64 docker://18.9.0
# kubectl run -i --tty --image nginx:alpine test --rm /bin/sh
If you don't see a command prompt, try pressing enter.
/ # curl 10.100.0.136:30000
<!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>
<p>If you see this page, the nginx web server is successfully installed and
working. Further configuration is required.</p>

<p>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>.</p>

<p><em>Thank you for using nginx.</em></p>
</body>
</html>
/ #
```

----End



## externalTrafficPolicy (Service Affinity)

For a NodePort Service, requests are first sent to the node port, then the Service, and finally the pod backing the Service. The backing pod may be not located in the node receiving the requests. By default, the backend workload can be accessed from any node IP address and service port. If the pod is not on the node that receives the request, the request will be redirected to the node where the pod is located, which may cause performance loss.

**externalTrafficPolicy** is a configuration parameter of the Service.

```
apiVersion: v1
kind: Service
metadata:
  name: nginx-nodeport
spec:
  externalTrafficPolicy: local
  ports:
  - name: service
    nodePort: 30000
    port: 80
    protocol: TCP
    targetPort: 80
  selector:
    app: nginx
  type: NodePort
```

If the value of **externalTrafficPolicy** is **local**, requests sent from *Node IP address:Service port* will be forwarded only to the pod on the local node. If the node does not have a pod, the requests are suspended.

The other value of **externalTrafficPolicy** is **cluster** (default value), which indicates that requests are forwarded in a cluster.

You can set this parameter when creating a Service of the NodePort type on the CCE console.

The screenshot shows the CCE console interface for creating a Service. The '访问类型' (Access Type) dropdown is set to '节点访问 (NodePort)'. Below it, a note states: '集群下节点有绑定弹性IP, 则可以使用弹性IP访问该服务。' (If nodes in the cluster have a bound elastic IP, you can use the elastic IP to access the service). The 'Service名称' (Service Name) is 'example', '集群名称' (Cluster Name) is 'cce-ss', and '命名空间' (Namespace) is 'default'. There is a '创建命名空间' (Create Namespace) link next to the namespace dropdown. The '关联工作负载' (Associate Workload) button is '+ 选择工作负载'. The '服务亲和' (Service Affinity) section has two tabs: '集群级别' (Cluster Level) and '节点级别' (Node Level). Below the tabs, two notes are listed: 1. '集群下所有节点的IP+访问端口均可以访问到此服务关联的负载。' (All nodes in the cluster can access the workload associated with this service through their IP and access port.) 2. '服务访问会因路由跳转导致一定性能损失, 且无法获取到客户端源IP。' (Service access will cause a certain performance loss due to route redirection, and the source IP of the client cannot be obtained.)

The values of **externalTrafficPolicy** are as follows:

- **cluster**: The IP addresses and access ports of all nodes in a cluster can 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.
- **local**: Only the IP address and access port of the node where the workload is located can 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.

### 17.8.3.4 LoadBalancer

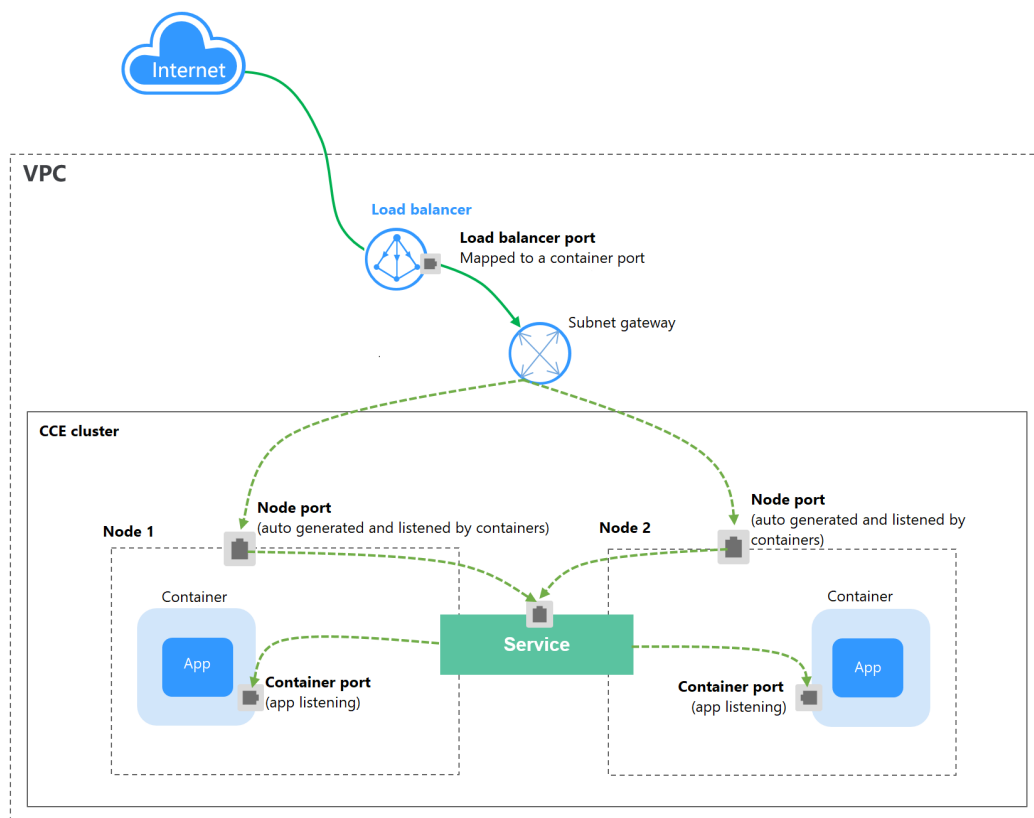
#### Scenario

A workload can be accessed from public networks through a load balancer, which is more secure and reliable than EIP.

The LoadBalancer access address is in the format of <IP address of public network load balancer>:<access port>, for example, **10.117.117.117:80**.

In this access mode, requests are transmitted through an ELB load balancer to a node and then forwarded to the destination pod through the Service.

Figure 17-122 LoadBalancer



#### Notes and Constraints

- LoadBalancer Services allow workloads to be accessed from public networks through **ELB**. This access mode has the following restrictions:
  - It is recommended that automatically created load balancers not be used by other resources. Otherwise, these load balancers cannot be completely deleted, causing residual resources.
  - Do not change the listener name for the load balancer in clusters of v1.15 and earlier. Otherwise, the load balancer cannot be accessed.

- After a Service is created, if the **affinity setting** is switched from the cluster level to the node level, the connection tracing table will not be cleared. You are advised not to modify the Service affinity setting after the Service is created. If you need to modify it, create a Service again.
- If the service affinity is set to the node level (that is, **externalTrafficPolicy** is set to **Local**), the cluster may fail to access the Service by using the ELB address. For details, see [Why a Cluster Fails to Access Services by Using the ELB Address](#).
- If you create a LoadBalancer Service on the CCE console, a random node port is automatically generated. If you use kubectl to create a LoadBalancer Service, a random node port is generated unless you specify one.
- In a CCE cluster, if the cluster-level affinity is configured for a LoadBalancer Service, requests are distributed to the node ports of each node using SNAT when entering the cluster. The number of node ports cannot exceed the number of available node ports on the node. If the Service affinity is at the node level (local), there is no such constraint.
- When the cluster service forwarding (proxy) mode is IPVS, the node IP cannot be configured as the external IP of the Service. Otherwise, the node is unavailable.

## Adding a Service When Creating a Workload

You can set the Service when creating a workload on the CCE console. An Nginx workload is used as an example.

**Step 1** In the **Set Application Access** step of [Creating a Deployment](#), [Creating a StatefulSet](#), or [Creating a DaemonSet](#), click **Add Service** and set the following parameters:

- **Access Type:** Select **LoadBalancer (ELB)**.
- **Service Name:** Specify a Service name, which can be the same as the workload name.
- **Service Affinity:**
  - Cluster level: The IP addresses and access ports of all nodes in a cluster can 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 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.

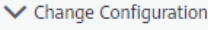
### ELB Configuration

- **Elastic Load Balancer:** A load balancer automatically distributes Internet access traffic to multiple nodes where the workload is located.

You can create **public network** or **private network** load balancers.

- **Public network:** You can select an existing public network load balancer or have the system automatically create a new one.
- **Private network:** You can select an existing private network load balancer or have the system automatically create a new private network load balancer.

The selected or created load balancer must be in the same VPC as the current cluster, and it must match the load balancer type (private or public network).

- **Enterprise Project:** Select an enterprise project in which the load balancer is created.
- **Specifications:** This field is displayed only when you select **Public network** and **Automatically created** for **Elastic Load Balancer**. You can click  to modify the name, specifications, billing mode, and bandwidth of the load balancer.
- **Algorithm Type:** Three algorithms are available: weighted round robin, weighted least connections algorithm, or source IP hash.

#### NOTE

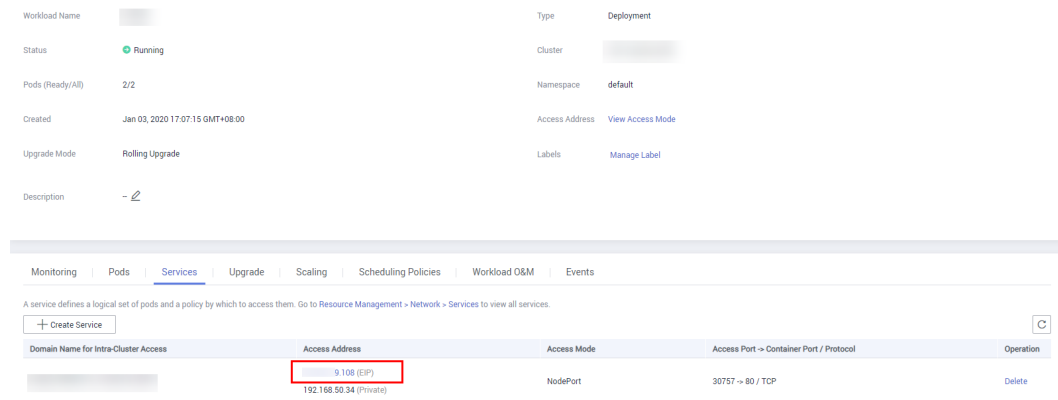
- **Weighted round robin:** Requests are forwarded to different servers based on their weights, which indicate server processing performance. Backend servers with higher weights receive proportionately more requests, whereas equal-weighted servers receive the same number of requests. This algorithm is often used for short connections, such as HTTP services.
- **Weighted least connections:** In addition to the weight assigned to each server, the number of connections processed by each backend server is also considered. Requests are forwarded to the server with the lowest connections-to-weight ratio. Building on **least connections**, the **weighted least connections** algorithm assigns a weight to each server based on their processing capability. This algorithm is often used for persistent connections, such as database connections.
- **Source IP hash:** The source IP address of each request is calculated using the hash algorithm to obtain a unique hash key, and all backend servers are numbered. The generated key allocates the client to a particular server. This enables requests from different clients to be distributed in load balancing mode and ensures that requests from the same client are forwarded to the same server. This algorithm applies to TCP connections without cookies.
- **Sticky Session:** This function is disabled by default. You can select **Based on source IP address**. Listeners ensure session stickiness based on IP addresses. Requests from the same IP address will be forwarded to the same backend server.
- **Health Check:** This function is enabled by default. The health check is for the load balancer. By default, the Service ports (Node Port and container port of the Service) are used for health check. You can also specify another port for health check. After the port is specified, a Service port (name: **cce-healthz**; protocol: **TCP**) will be added for the Service.
- **Port Settings**
  - **Protocol:** protocol used by the Service.
  - **Container Port:** port defined in the container image and on which the workload listens. The Nginx application listens on port 80.
  - **Access Port:** port mapped to the container port at the load balancer's IP address. The workload can be accessed at *<Load balancer's IP address>:<Access port>*. The port number range is 1–65535.

**Step 2** After the configuration is complete, click **OK**.

**Step 3** On the workload creation page, click **Next: Configure Advanced Settings**. On the page displayed, click **Create**.

**Step 4** After the workload is successfully created, choose **Workloads > Deployments** or **Workloads > StatefulSets** on the CCE console. Click the name of the workload to view its details. On the workload details page, click the **Services** tab and obtain the access address.

**Figure 17-123** Obtaining the load balancer IP address



**Step 5** Click the access address.

----End

## Adding a Service After Creating a Workload

You can set the Service after creating a workload. This has no impact on the workload status and takes effect immediately. The procedure is as follows:

**Step 1** Log in to the CCE console. In the navigation pane, choose **Resource Management > Network**.

**Step 2** On the **Services** tab page, click **Create Service**.

The parameters are the same as those in [Adding a Service When Creating a Workload](#).

**Step 3** Click **Create**.

----End

## Using kubectl to Create a Service (Using an Existing Load Balancer)

You can set the access type when creating a workload using kubectl. This section uses an Nginx workload as an example to describe how to add a LoadBalancer Service using kubectl.

**Step 1** Use kubectl to connect to the cluster. For details, see [Connecting to a Cluster Using kubectl](#).

**Step 2** Create and edit the **nginx-deployment.yaml** and **nginx-elb-svc.yaml** files.

The file names are user-defined. **nginx-deployment.yaml** and **nginx-elb-svc.yaml** are merely example file names.

**vi nginx-deployment.yaml**

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx
spec:
  replicas: 1
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
        - image: nginx
          name: nginx
      imagePullSecrets:
        - name: default-secret
```

### vi nginx-elb-svc.yaml

#### NOTE

Before enabling sticky session, ensure that the following conditions are met:

- The workload protocol is TCP.
- Anti-affinity has been configured between pods of the workload. That is, all pods of the workload are deployed on different nodes. For details, see [Workload-Node Anti-Affinity](#).

```
apiVersion: v1
kind: Service
metadata:
  annotations:
    kubernetes.io/elb.id: 3c7caa5a-a641-4bff-801a-feace27424b6 # Load balancer ID. Replace it with
the actual value.
    kubernetes.io/elb.class: performance # Load balancer type
  name: nginx
spec:
  ports:
    - name: service0
      port: 80
      protocol: TCP
      targetPort: 80
  selector:
    app: nginx
  type: LoadBalancer
```

**Table 17-71** Key parameters

Parameter	Mandatory	Type	Description
kubernetes.io/elb.class		String	Select a proper load balancer type as required. The value can be: <ul style="list-style-type: none"> <li>• <b>performance</b>: dedicated load balancer, which can be used only in clusters of v1.17 and later.</li> </ul>

Parameter	Mandatory	Type	Description
kubernetes.io/elb.session-affinity-mode	No	String	<p>Listeners ensure session stickiness based on IP addresses. Requests from the same IP address will be forwarded to the same backend server.</p> <ul style="list-style-type: none"> <li>Disabling sticky session: Do not set this parameter.</li> <li>Enabling sticky session: Set this parameter to <b>SOURCE_IP</b>, indicating that the sticky session is based on the source IP address.</li> </ul>
kubernetes.io/elb.session-affinity-option	No	<a href="#">Table 17-72</a> Object	This parameter specifies the sticky session timeout.
kubernetes.io/elb.id	Yes	String	<p>This parameter indicates the ID of a load balancer. The value can contain 1 to 100 characters.</p> <p>Mandatory when an existing load balancer is to be associated.</p> <p><b>Obtaining the load balancer ID:</b></p> <p>On the management console, click <b>Service List</b>, and choose <b>Networking &gt; Elastic Load Balance</b>. Click the name of the target load balancer. On the <b>Summary</b> tab page, find and copy the ID.</p>
kubernetes.io/elb.subnet-id	-	String	<p>This parameter indicates the ID of the subnet where the cluster is located. The value can contain 1 to 100 characters.</p> <ul style="list-style-type: none"> <li>Mandatory when a cluster of v1.11.7-r0 or earlier is to be automatically created.</li> <li>Optional for clusters later than v1.11.7-r0.</li> </ul>

Parameter	Mandatory	Type	Description
kubernetes.io/elb.lb-algorithm	No	String	This parameter indicates the load balancing algorithm of the backend server group. The default value is <b>ROUND_ROBIN</b> . Options: <ul style="list-style-type: none"> <li>• <b>ROUND_ROBIN</b>: weighted round robin algorithm</li> <li>• <b>LEAST_CONNECTIONS</b>: weighted least connections algorithm</li> <li>• <b>SOURCE_IP</b>: source IP hash algorithm</li> </ul> When the value is <b>SOURCE_IP</b> , the weights of backend servers in the server group are invalid.
kubernetes.io/elb.health-check-flag	No	String	Whether to enable the ELB health check. Enabled by default. <ul style="list-style-type: none"> <li>• Enabling health check: Leave blank this parameter or set it to <b>on</b>.</li> <li>• Disabling health check: Set this parameter to <b>off</b>.</li> </ul>
kubernetes.io/elb.health-check-option	No	<a href="#">Table 17-73</a> Object	ELB health check configuration items.
port	Yes	Integer	Access port that is registered on the load balancer and mapped to the cluster-internal IP address.
targetPort	Yes	String	Container port set on the CCE console.

**Table 17-72** Data structure of the elb.session-affinity-option field

Parameter	Mandatory	Type	Description
persistence_timeout	Yes	String	Sticky session timeout, in minutes. This parameter is valid only when <b>elb.session-affinity-mode</b> is set to <b>SOURCE_IP</b> . Value range: 1 to 60. Default value: <b>60</b>



**Table 17-73** Data structure description of the **elb.health-check-option** field

Parameter	Mandatory	Type	Description
delay	No	String	Initial waiting time (in seconds) for starting the health check. Value range: 1 to 50. Default value: <b>5</b>
timeout	No	String	Health check timeout, in seconds. Value range: 1 to 50. Default value: <b>10</b>
max_retries	No	String	Maximum number of health check retries. Value range: 1 to 10. Default value: <b>3</b>
protocol	No	String	Health check protocol. Default value: protocol of the associated Service Value options: TCP, UDP_CONNECT, or HTTP
path	No	String	Health check URL. This parameter needs to be configured when the protocol is HTTP. Default value: / The value contains 1 to 10,000 characters.

**Step 3** Create a workload.

**kubectl create -f nginx-deployment.yaml**

If information similar to the following is displayed, the workload has been created.

```
deployment "nginx" created
```

**kubectl get pod**

If information similar to the following is displayed, the workload is running.

```
NAME                READY   STATUS    RESTARTS   AGE
nginx-2601814895-c1xwh 1/1     Running   0          6s
```

**Step 4** Create a Service.

**kubectl create -f nginx-elb-svc.yaml**

If information similar to the following is displayed, the Service has been created.

```
service "nginx" created
```

**kubectl get svc**

If information similar to the following is displayed, the access type has been set successfully, and the workload is accessible.

NAME	TYPE	CLUSTER-IP	EXTERNAL-IP	PORT(S)	AGE
kubernetes	ClusterIP	10.247.0.1	<none>	443/TCP	3d
nginx	LoadBalancer	10.247.130.196	10.78.42.242	80:31540/TCP	51s

**Step 5** Enter the URL in the address box of the browser, for example, **10.78.42.242:80**. **10.78.42.242** indicates the IP address of the load balancer, and **80** indicates the access port displayed on the CCE console.

The Nginx is accessible.

**Figure 17-124** Accessing Nginx through the LoadBalancer Service

## Welcome to nginx!

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 [nginx.org](http://nginx.org).  
Commercial support is available at [nginx.com](http://nginx.com).

*Thank you for using nginx.*

----End

## Using kubectl to Create a Service (Automatically Creating a Load Balancer)

You can add a Service when creating a workload using kubectl. This section uses an Nginx workload as an example to describe how to add a LoadBalancer Service using kubectl.

**Step 1** Use kubectl to connect to the cluster. For details, see [Connecting to a Cluster Using kubectl](#).

**Step 2** Create and edit the **nginx-deployment.yaml** and **nginx-elb-svc.yaml** files.

The file names are user-defined. **nginx-deployment.yaml** and **nginx-elb-svc.yaml** are merely example file names.

### vi nginx-deployment.yaml

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx
spec:
  replicas: 1
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
        - image: nginx
          name: nginx
          imagePullSecrets:
            - name: default-secret
```

## vi nginx-elb-svc.yaml

### NOTE

Before enabling sticky session, ensure that the following conditions are met:

- The workload protocol is TCP.
- Anti-affinity has been configured between pods of the workload. That is, all pods of the workload are deployed on different nodes. For details, see [Workload-Node Anti-Affinity](#).

Example of a Service using a dedicated, public network load balancer:

```
apiVersion: v1
kind: Service
metadata:
  name: nginx
  labels:
    app: nginx
  namespace: default
  annotations:
    kubernetes.io/elb.class: performance
    kubernetes.io/elb.autocreate:
      {
        "type": "public",
        "bandwidth_name": "cce-bandwidth-1626694478577",
        "bandwidth_chargemode": "traffic",
        "bandwidth_size": 5,
        "bandwidth_sharetype": "PER",
        "eip_type": "5_bgp",
        "available_zone": [
          "eu-west-101a"
        ],
        "l4_flavor_name": "L4_flavor.elb.s1.small"
      }
spec:
  selector:
    app: nginx
  ports:
  - name: cce-service-0
    targetPort: 80
    nodePort: 0
    port: 80
    protocol: TCP
  type: LoadBalancer
```

**Table 17-74** Key parameters

Parameter	Mandatory	Type	Description
kubernetes.io/elb.class		String	Select a proper load balancer type as required. The value can be: <ul style="list-style-type: none"> <li>• <b>performance</b>: dedicated load balancer, which can be used only in clusters of v1.17 and later.</li> </ul>

Parameter	Mandatory	Type	Description
kubernetes.io/elb.subnet-id	-	String	<p>This parameter indicates the ID of the subnet where the cluster is located. The value can contain 1 to 100 characters.</p> <ul style="list-style-type: none"> <li>• Mandatory when a cluster of v1.11.7-r0 or earlier is to be automatically created.</li> <li>• Optional for clusters later than v1.11.7-r0.</li> </ul>
kubernetes.io/elb.enterpriseID	No	String	<p><b>Clusters of v1.15 and later versions support this field. In clusters earlier than v1.15, load balancers are created in the default project by default.</b></p> <p>This parameter indicates the ID of the enterprise project in which the ELB load balancer will be created.</p> <p>If this parameter is not specified or is set to <b>0</b>, resources will be bound to the default enterprise project.</p> <p><b>How to obtain:</b></p> <p>Log in to the management console and choose <b>Enterprise &gt; Project Management</b> on the top menu bar. In the list displayed, click the name of the target enterprise project, and copy the ID on the enterprise project details page.</p>
kubernetes.io/elb.session-affinity-option	No	<a href="#">Table 17-72</a> Object	Sticky session timeout.

Parameter	Mandatory	Type	Description
kubernetes.io/elb.autocreate	Yes	<a href="#">elb.auto create</a> object	<p>Whether to automatically create a load balancer associated with the Service.</p> <p><b>Example:</b></p> <ul style="list-style-type: none"> <li>Automatically created public network load balancer: {"type":"public","bandwidth_name":"cce-bandwidth-1551163379627","bandwidth_chargemode":"traffic","bandwidth_size":5,"bandwidth_sharetype":"PER","eip_type":"5_bgp","name":"james"}</li> <li>Automatically created private network load balancer: {"type":"inner","name":"A-location-d-test"}</li> </ul>
kubernetes.io/elb.lb-algorithm	No	String	<p>This parameter indicates the load balancing algorithm of the backend server group. The default value is <b>ROUND_ROBIN</b>.</p> <p>Options:</p> <ul style="list-style-type: none"> <li><b>ROUND_ROBIN</b>: weighted round robin algorithm</li> <li><b>LEAST_CONNECTIONS</b>: weighted least connections algorithm</li> <li><b>SOURCE_IP</b>: source IP hash algorithm</li> </ul> <p>When the value is <b>SOURCE_IP</b>, the weights of backend servers in the server group are invalid.</p>
kubernetes.io/elb.health-check-flag	No	String	<p>Whether to enable the ELB health check.</p> <p>Disabled by default.</p> <ul style="list-style-type: none"> <li>Enabling health check: Leave blank this parameter or set it to <b>on</b>.</li> <li>Disabling health check: Set this parameter to <b>off</b>.</li> </ul>
kubernetes.io/elb.health-check-option	No	<a href="#">Table 17-73</a> Object	ELB health check configuration items.

Parameter	Mandatory	Type	Description
kubernetes.io/elb.session-affinity-mode	No	String	Listeners ensure session stickiness based on IP addresses. Requests from the same IP address will be forwarded to the same backend server. <ul style="list-style-type: none"> <li>Disabling sticky session: Do not set this parameter.</li> <li>Enabling sticky session: Set this parameter to <b>SOURCE_IP</b>, indicating that the sticky session is based on the source IP address.</li> </ul>
kubernetes.io/elb.session-affinity-option	No	<a href="#">Table 17-72</a> Object	Sticky session timeout.
kubernetes.io/hws-hostNetwork	No	String	This parameter indicates whether the workload Services use the host network. Setting this parameter to <b>true</b> will enable the load balancer to forward requests to the host network. The host network is not used by default. The value can be <b>true</b> or <b>false</b> .
externalTrafficPolicy	No	String	If sticky session is enabled, add this parameter so that requests are transferred to a fixed node. If a LoadBalancer Service with this parameter set to <b>Local</b> is created, a client can access the target backend only if the client is installed on the same node as the backend.

**Table 17-75** Data structure of the elb.autocreate field

Parameter	Mandatory	Type	Description
name	No	String	Name of the automatically created load balancer. Value range: a string of 1 to 64 characters, including lowercase letters, digits, and underscores (_). The value must start with a lowercase letter and end with a lowercase letter or digit. Default name: <b>cce-lb+service.UID</b>

Parameter	Mandatory	Type	Description
type	No	String	Network type of the load balancer. <ul style="list-style-type: none"> <li><b>public</b>: public network load balancer</li> <li><b>inner</b>: private network load balancer</li> </ul> Default value: <b>inner</b>
bandwidth_name	Yes for public network load balancers	String	Bandwidth name. The default value is <b>cce-bandwidth-*****</b> . Value range: a string of 1 to 64 characters, including lowercase letters, digits, and underscores (_). The value must start with a lowercase letter and end with a lowercase letter or digit.
bandwidth_charge_mode	No	String	Bandwidth billing mode. <ul style="list-style-type: none"> <li><b>traffic</b>: billed by traffic</li> </ul>
bandwidth_size	Yes for public network load balancers	Integer	Bandwidth size. The default value is 1 to 2000 Mbit/s. Set this parameter based on the bandwidth range allowed in your region.
bandwidth_sharetype	Yes for public network load balancers	String	Bandwidth sharing mode. <ul style="list-style-type: none"> <li><b>PER</b>: dedicated bandwidth</li> </ul>
eip_type	Yes for public network load balancers	String	EIP type. <ul style="list-style-type: none"> <li><b>5_bgp</b>: dynamic BGP</li> </ul>
available_zone	Yes	Array of strings	AZ where the load balancer is located. This parameter is available only for dedicated load balancers.
l4_flavor_name	Yes	String	Flavor name of the layer-4 load balancer. This parameter is available only for dedicated load balancers.

Parameter	Mandatory	Type	Description
l7_flavor_name	No	String	Flavor name of the layer-7 load balancer. This parameter is available only for dedicated load balancers.
elb_virsubnet_ids	No	Array of strings	Subnet where the backend server of the load balancer is located. If this parameter is left blank, the default cluster subnet is used. Load balancers occupy different number of subnet IP addresses based on their specifications. Therefore, you are not advised to use the subnet CIDR blocks of other resources (such as clusters and nodes) as the load balancer CIDR block. This parameter is available only for dedicated load balancers. Example: "elb_virsubnet_ids": [ "14567f27-8ae4-42b8-ae47-9f847a4690dd" ]

**Step 3** Create a workload.

**kubectl create -f nginx-deployment.yaml**

If information similar to the following is displayed, the workload is being created.

```
deployment "nginx" created
```

**kubectl get po**

If information similar to the following is displayed, the workload is running.

```
NAME                READY   STATUS    RESTARTS   AGE
nginx-2601814895-c1xwh 1/1     Running   0           6s
```

**Step 4** Create a Service.

**kubectl create -f nginx-elb-svc.yaml**

If information similar to the following is displayed, the Service has been created.

```
service "nginx" created
```

**kubectl get svc**

If information similar to the following is displayed, the access type has been set successfully, and the workload is accessible.

```
NAME      TYPE          CLUSTER-IP   EXTERNAL-IP   PORT(S)    AGE
kubernetes ClusterIP   10.247.0.1   <none>        443/TCP    3d
nginx     LoadBalancer 10.247.130.196 10.78.42.242 80:31540/TCP 51s
```



**Step 5** Enter the URL in the address box of the browser, for example, **10.78.42.242:80**. **10.78.42.242** indicates the IP address of the load balancer, and **80** indicates the access port displayed on the CCE console.

The Nginx is accessible.

**Figure 17-125** Accessing Nginx through the LoadBalancer Service

## Welcome to nginx!

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 [nginx.org](https://nginx.org).  
Commercial support is available at [nginx.com](https://nginx.com).

*Thank you for using nginx.*

----End

## Why a Cluster Fails to Access Services by Using the ELB Address

If the service affinity of a LoadBalancer Service is set to the node level, that is, the value of **externalTrafficPolicy** is **Local**, the ELB address may fail to be accessed from the cluster (specifically, nodes or containers).

This is because when the LoadBalancer Service is created, kube-proxy adds the ELB access address (external-ip) to iptables or IPVS. When the ELB address is accessed from the cluster, the ELB load balancer is not used. Instead, kube-proxy directly forwards the access request. The case depends on which container network model and service forwarding mode you use.

The following methods can be used to solve this problem:

- **(Recommended)** In the cluster, use the ClusterIP Service or service domain name for access.
- Set **externalTrafficPolicy** of the Service to **Cluster**, which means cluster-level service affinity. Note that this affects source address persistence.

```
apiVersion: v1
kind: Service
metadata:
  annotations:
    kubernetes.io/elb.class: union
    kubernetes.io/elb.autocreate: '{"type":"public","bandwidth_name":"cce-
bandwidth","bandwidth_chargemode":"traffic","bandwidth_size":5,"bandwidth_sharetype":"PER","eip_t
ype":"5_bgp","name":"james"}'
  labels:
    app: nginx
    name: nginx
spec:
  externalTrafficPolicy: Cluster
  ports:
  - name: service0
    port: 80
    protocol: TCP
    targetPort: 80
  selector:
```

```
app: nginx
type: LoadBalancer
```

### 17.8.3.5 Service Annotations

CCE allows you to add annotations to a YAML file to realize some advanced Service functions. The following table describes the annotations you can add.

The annotations of a Service are the parameters that need to be specified for connecting to a load balancer. For details about how to use the annotations, see [Using kubectl to Create a Service \(Automatically Creating a Load Balancer\)](#).

**Table 17-76** Service annotations

Parameter	Description
kubernetes.io/elb.class	Select a proper load balancer type. Value: <ul style="list-style-type: none"> <li><b>performance</b>: dedicated load balancer, which can be used only in clusters of v1.17 and later.</li> </ul>
kubernetes.io/elb.id	ID of a load balancer. The value can contain 1 to 100 characters. Mandatory when an existing load balancer is to be associated. <b>How to obtain:</b> On the management console, click <b>Service List</b> , and choose <b>Network &gt; Elastic Load Balance</b> . Click the name of the target load balancer. On the <b>Summary</b> tab page, find and copy the ID.
kubernetes.io/elb.subnet-id	ID of the subnet where the cluster is located. The value can contain 1 to 100 characters. <ul style="list-style-type: none"> <li>Mandatory when a cluster of v1.11.7-r0 or earlier is to be automatically created.</li> <li>Optional for clusters later than v1.11.7-r0.</li> </ul>

Parameter	Description
kubernetes.io/elb.enterpriseID	<p><b>Clusters of v1.15 and later versions support this field. In clusters earlier than v1.15, load balancers are created in the default project by default.</b></p> <p>This parameter indicates the ID of the enterprise project in which the ELB load balancer will be created.</p> <p>If this parameter is not specified or is set to <b>0</b>, resources will be bound to the default enterprise project.</p> <p><b>How to obtain:</b></p> <p>Log in to the management console and choose <b>Enterprise &gt; Project Management</b> on the top menu bar. In the list displayed, click the name of the target enterprise project, and copy the ID on the enterprise project details page.</p>
kubernetes.io/elb.autocreate	<p>Whether to automatically create a load balancer associated with the Service.</p> <p><b>Example:</b></p> <ul style="list-style-type: none"> <li>• If a public network load balancer will be automatically created, set this parameter to the following value: {"type":"public","bandwidth_name":"cce-bandwidth-1551163379627","bandwidth_charactermode":"traffic","bandwidth_size":5,"bandwidth_sharetype":"PER","eip_type":"5_bgp","name":"james"}</li> <li>• If a private network load balancer will be automatically created, set this parameter to the following value: {"type":"inner","name":"A-location-d-test"}</li> </ul>
kubernetes.io/elb.lb-algorithm	<p>Load balancing algorithm of the backend server group. The default value is <b>ROUND_ROBIN</b>.</p> <p>Value:</p> <ul style="list-style-type: none"> <li>• <b>ROUND_ROBIN</b>: weighted round robin algorithm</li> <li>• <b>LEAST_CONNECTIONS</b>: weighted least connections algorithm</li> <li>• <b>SOURCE_IP</b>: source IP hash algorithm</li> </ul> <p>When the value is <b>SOURCE_IP</b>, the weights of backend servers in the server group are invalid.</p>

Parameter	Description
kubernetes.io/elb.health-check-flag	Whether to enable the ELB health check. Disabled by default. <ul style="list-style-type: none"> <li>Enabling health check: Leave blank this parameter or set it to <b>on</b>.</li> <li>Disabling health check: Set this parameter to <b>off</b>.</li> </ul>
kubernetes.io/elb.health-check-option	ELB health check configuration items.
kubernetes.io/elb.session-affinity-mode	Listeners ensure session stickiness based on IP addresses. Requests from the same IP address will be routed to the same backend server. <ul style="list-style-type: none"> <li>Disabling sticky session: Do not set this parameter.</li> <li>Enabling sticky session: Set this parameter to <b>SOURCE_IP</b>, indicating that the sticky session is based on the source IP address.</li> </ul>
kubernetes.io/elb.session-affinity-option	Sticky session timeout.
kubernetes.io/hws-hostNetwork	Whether the workload Services use the host network. Setting this parameter to <b>true</b> will enable the load balancer to forward requests to the host network.  The host network is not used by default. The value can be <b>true</b> or <b>false</b> .

## 17.8.4 Ingress

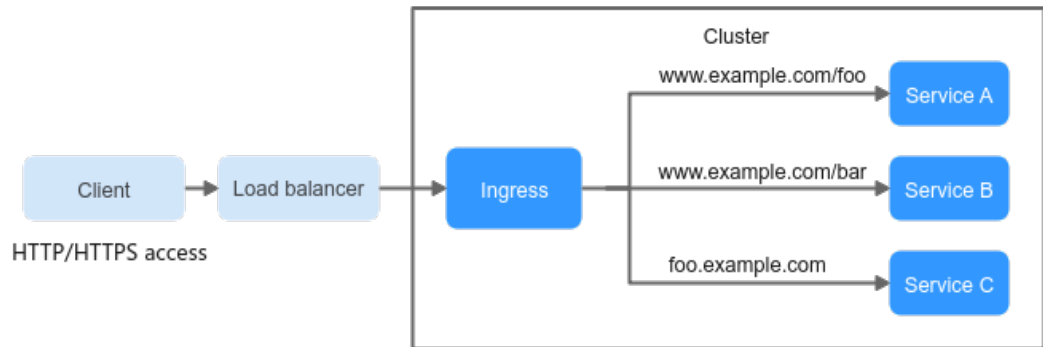
### 17.8.4.1 Overview

#### Why We Need Ingresses

A Service is generally used to forward access requests based on TCP and UDP and provide layer-4 load balancing for clusters. However, in actual scenarios, if there is a large number of HTTP/HTTPS access requests on the application layer, the Service cannot meet the forwarding requirements. Therefore, the Kubernetes cluster provides an HTTP-based access mode, that is, ingress.

An ingress is an independent resource in the Kubernetes cluster and defines rules for forwarding external access traffic. As shown in [Figure 17-126](#), you can customize forwarding rules based on domain names and URLs to implement fine-grained distribution of access traffic.

**Figure 17-126** Ingress diagram



The following describes the ingress-related definitions:

- Ingress object: a set of access rules that forward requests to specified Services based on domain names or URLs. It can be added, deleted, modified, and queried by calling APIs.
- Ingress Controller: an executor for request forwarding. It monitors the changes of resource objects such as ingresses, Services, endpoints, secrets (mainly TLS certificates and keys), nodes, and ConfigMaps in real time, parses rules defined by ingresses, and forwards requests to the corresponding backend Services.

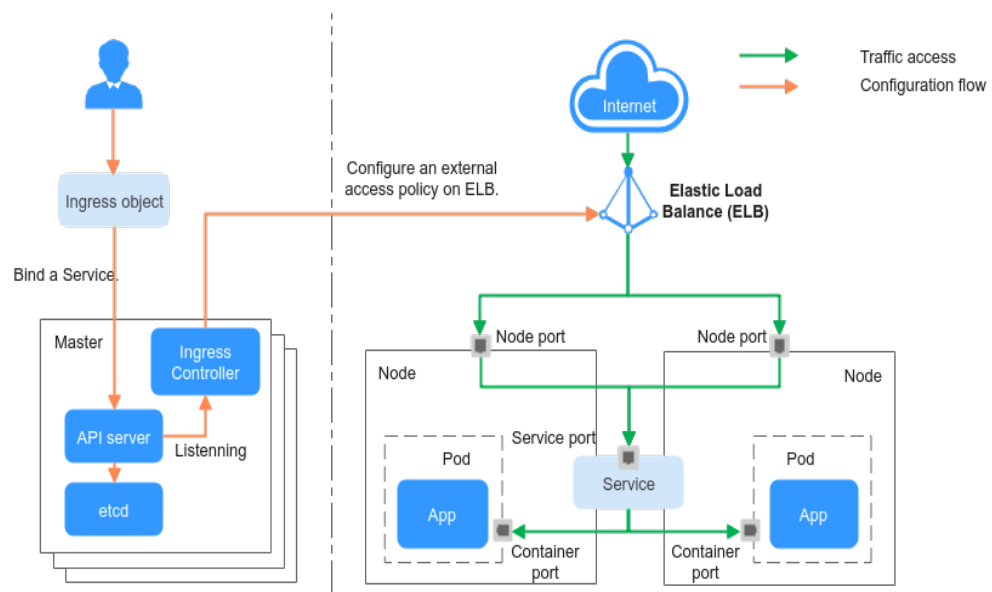
## Working Principle of ELB Ingress Controller

ELB Ingress Controller developed by CCE implements layer-7 network access for the internet and intranet (in the same VPC) based on ELB and distributes access traffic to the corresponding Services using different URLs.

ELB Ingress Controller is deployed on the master node and bound to the load balancer in the VPC where the cluster resides. Different domain names, ports, and forwarding policies can be configured for the same load balancer (with the same IP address). [Figure 17-127](#) shows the working principle of ELB Ingress Controller.

1. A user creates an ingress object and configures a traffic access rule in the ingress, including the load balancer, URL, SSL, and backend service port.
2. When Ingress Controller detects that the ingress object changes, it reconfigures the listener and backend server route on the ELB side according to the traffic access rule.
3. When a user accesses a workload, the traffic is forwarded to the corresponding backend service port based on the forwarding policy configured on ELB, and then forwarded to each associated workload through the Service.

**Figure 17-127** Working principle of ELB Ingress Controller



## 17.8.4.2 Using ELB Ingresses on the Console

### Prerequisites

- An ingress provides network access for backend workloads. Ensure that a workload is available in a cluster. If no workload is available, deploy a workload by referring to [Creating a Deployment](#), [Creating a StatefulSet](#), or [Creating a DaemonSet](#).

### Precautions

- It is recommended that other resources not use the load balancer automatically created by an ingress. Otherwise, the load balancer will be occupied when the ingress is deleted, resulting in residual resources.
- After an ingress is created, upgrade and maintain the configuration of the selected load balancers on the CCE console. Do not modify the configuration on the ELB console. Otherwise, the ingress service may be abnormal.
- The URL registered in an ingress forwarding policy must be the same as the URL exposed by the backend Service. Otherwise, a 404 error will be returned.

### Adding an ELB Ingress

This section uses an Nginx workload as an example to describe how to add an ELB ingress.

**Step 1** Log in to the CCE console.

**Step 2** In the navigation pane, choose **Resource Management > Network**. On the **Ingresses** tab page, select the corresponding cluster and namespace.

**Step 3** Click **Create Ingress** to access the ingress configuration page.

Set the ingress parameters as required. The key parameters are as follows:

- **Access Type:** Use a load balancer to access Services. Requests can be forwarded only to NodePort Services.
- **Ingress Name:** Specify a name of an ingress, for example, **ingress-demo**.
- **Cluster Name:** Select the cluster to which the ingress is to be added.
- **Namespace:** Select the namespace to which the ingress is to be added.
- **ELB Configuration:** Ingress uses the load balancer of the ELB service to provide layer-7 network access. You can select an existing load balancer or have the system automatically create a new one. To manually create a load balancer, click **Create Load Balancer** and then click the refresh button.

---

**NOTICE**

- It is recommended that other resources not use the load balancer automatically created by an ingress. Otherwise, the load balancer will be occupied when the ingress is deleted, resulting in residual resources.
- To interconnect with an existing dedicated load balancer, ensure that HTTP is supported and the network type supports private networks.

---

**Elastic Load Balancer:** The selected or created load balancer must be in the same VPC as the current cluster, and it must match the load balancer type (private or public network).


You can create **public network** or **private network** load balancers. The default value is **Public network**.

- **Public network:** After you attach an EIP to a load balancer, the load balancer can distribute requests from the Internet to backend servers.
  - **Enterprise Project:** Select an enterprise project in which the load balancer is created.
  - **Change Configuration:** When selecting **Public network > Automatically created**, you can click **Change Configuration** to modify the name, specifications, billing mode, and bandwidth of the ELB instance to be created.
- **Private network:** After you attach a private IP address to a load balancer, the load balancer can distribute requests from the clients in the same VPC to backends.
  - **Enterprise Project:** Select an enterprise project in which the load balancer is created.
- **Listener Configuration:** Ingress configures a listener for the load balancer, which listens to requests from the load balancer and distributes traffic. After the configuration is complete, a listener is created on the load balancer. The default listener name is *k8s\_<Protocol type>\_<Port number>*, for example, *k8s\_HTTP\_80*.
  - **Front-End Protocol:** HTTP and HTTPS are available.
  - **External Port:** Port number that is open to the ELB service address. The port number can be specified randomly.
  - **Server Certificate:** When an HTTPS listener is created for a load balancer, you need to bind a certificate to the load balancer to support encrypted

authentication for HTTPS data transmission. For details on how to create a secret, see [Creating a Secret](#).

 NOTE

If there is already an HTTPS ingress for the chosen port on the load balancer, the certificate of the new HTTPS ingress must be the same as the certificate of the existing ingress. This means that a listener has only one certificate. If two certificates, each with a different ingress, are added to the same listener of the same load balancer, only the certificate added earliest takes effect on the load balancer.

- **SNI:** Click  to enable the Server Name Indication (SNI) function. SNI is an extended protocol of TLS. It allows multiple TLS-based access domain names to be provided for external systems using the same IP address and port number. Different domain names can use different security certificates. After SNI is enabled, the client is allowed to submit the requested domain name when initiating a TLS handshake request. After receiving the TLS request, the load balancer searches for the certificate based on the domain name in the request. If the certificate corresponding to the domain name is found, the load balancer returns the certificate for authorization. Otherwise, the default certificate (server certificate) is returned for authorization.

 NOTE

- The **SNI** option is available only when **HTTPS** is selected.
  - This function is supported only for clusters of v1.15.11 and later.
  - Specify the domain name for the SNI certificate. Only one domain name can be specified for each certificate. Wildcard-domain certificates are supported.
- **Security Policy:** combinations of different TLS versions and supported cipher suites available to HTTPS listeners.

For details about security policies, see ELB User Guide.

 NOTE

- **Security Policy** is available only when **HTTPS** is selected.
  - This function is supported only for clusters of v1.17.9 and later.
- **Forwarding Policies:** When the access address of a request matches the forwarding policy (a forwarding policy consists of a domain name and URL, for example, 10.117.117.117:80/helloworld), the request is forwarded to the corresponding target Service for processing. Click **Add Forwarding Policies** to add multiple forwarding policies.
    - **Domain Name:** actual domain name. Ensure that the domain name has been registered and archived. Once a domain name rule is configured, you must use the domain name for access.
    - Rule Matching
      - **Prefix match:** If the URL is set to **/healthz**, the URL that meets the prefix can be accessed. For example, **/healthz/v1** and **/healthz/v2**.



- **Exact match:** The URL can be accessed only when it is fully matched. For example, if the URL is set to **/healthz**, only **/healthz** can be accessed.
  - **Regular expression:** The URL is matched based on the regular expression. For example, if the regular expression is **/[A-Za-z0-9\_-.]+/test**, all URLs that comply with this rule can be accessed, for example, **/abcA9/test** and **/v1-Ab/test**. Two regular expression standards are supported: POSIX and Perl.
  - **URL:** access path to be registered, for example, **/healthz**.
  - **Target Service:** Select an existing Service or create a Service. Services that do not meet search criteria are automatically filtered out.
  - **Service Access Port:** Select the access port of the target Service.
  - **ELB Settings:** If multiple routes use the same Service, they are using the same Service load balancing configuration.
    - **Algorithm Type:** Three algorithms are available: weighted round robin, weighted least connections algorithm, or source IP hash. For details about the allocation policies, see [LoadBalancer](#).
    - **Sticky Session:** This function is disabled by default. After this function is enabled, you need to select a sticky session type and set the sticky session duration.  
**ELB cookie:** The load balancer generates a cookie after receiving a request from the client. All subsequent requests with the cookie are routed to the same backend server for processing.
    - **Health Check:** This function is disabled by default. Set the parameters as instructed.
  - **Operation:** Click **Delete** to delete the configuration.
- Step 4** After the configuration is complete, click **Create**. After the ingress is created, it is displayed in the ingress list.

On the ELB console, you can view the ELB automatically created through CCE. The default name is **cce-lb-ingress.UID**. Click the ELB name to access its details page. On the **Listeners** tab page, view the route settings of the ingress, including the URL, listener port, and backend server group port.

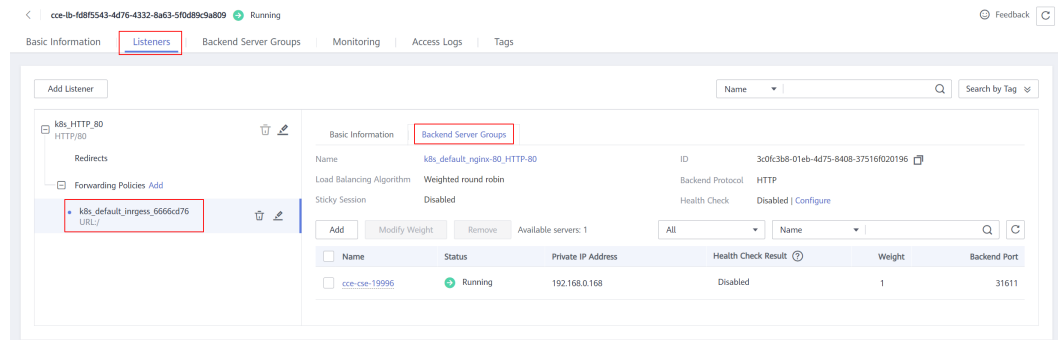
---

**NOTICE**

After the ingress is created, upgrade and maintain the selected load balancer on the CCE console. Do not maintain the load balancer on the ELB console. Otherwise, the ingress service may be abnormal.

---

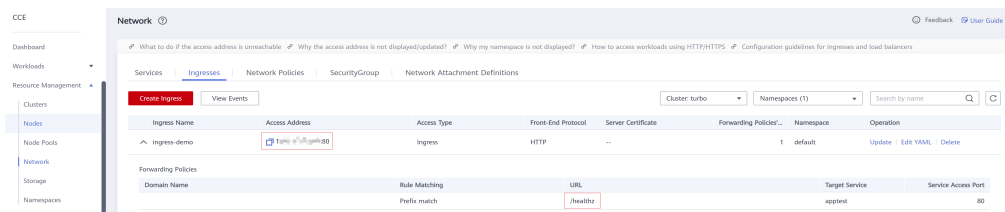
**Figure 17-128** ELB routing configuration



**Step 5** Access the /healthz interface of the workload, for example, workload **defaultbackend**.

1. Obtain the access address of the /healthz interface of the workload. The access address consists of the load balancer IP address, external port, and mapping URL, for example, 10.\*\*.\*\*.\*\*:80/healthz.

**Figure 17-129** Obtaining the access address



2. Enter the URL of the /healthz interface, for example, http://10.\*\*.\*\*.\*\*:80/healthz, in the address box of the browser to access the workload, as shown in [Figure 17-130](#).

**Figure 17-130** Accessing the /healthz interface of defaultbackend



----End

## Updating an Ingress

After adding an ingress, you can update its port, domain name, and route configuration. The procedure is as follows:

### NOTE

You can modify the load balancer settings, including algorithm, sticky session, and health check configurations, after you select a Service in **Forwarding Policies** on the CCE console. Do not modify these configurations on the ELB console.

- Step 1** Log in to the CCE console. In the navigation pane, choose **Resource Management** > **Network**. On the **Ingresses** tab page, filter ingresses by cluster and namespace, and click **Update** for the ingress to be updated.
- Step 2** On the **Update Ingress** page, modify the required parameters.  
The parameters are the same as those set during creation.
- Step 3** Click **Submit**. The ingress will be updated for the workload.
- End

### 17.8.4.3 Using kubectl to Create an ELB Ingress

#### Scenario

This section uses an [Nginx workload](#) as an example to describe how to create an ELB ingress using kubectl.

- If no load balancer is available in the same VPC, CCE can automatically create a load balancer when creating an ingress. For details, see [Creating an Ingress - Automatically Creating a Load Balancer](#).
- If a load balancer is available in the same VPC, perform the operation by referring to [Creating an Ingress - Interconnecting with an Existing Load Balancer](#).

#### Prerequisites

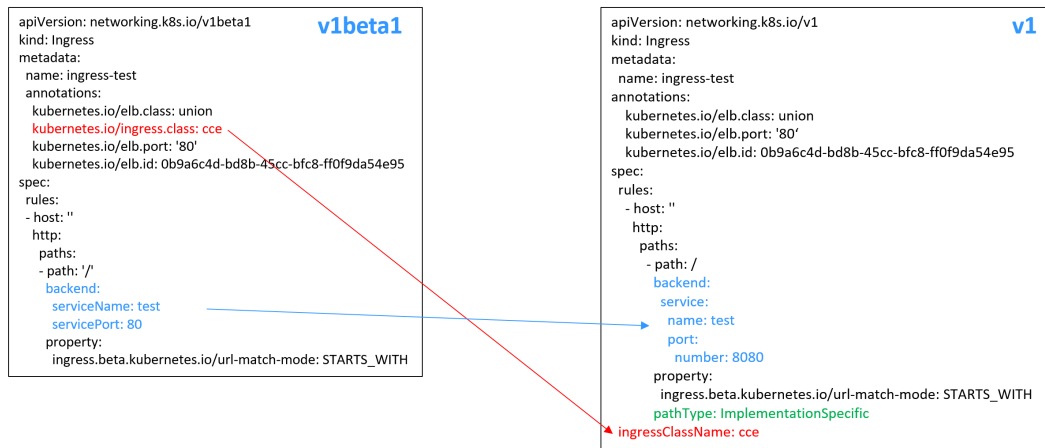
- An ingress provides network access for backend workloads. Ensure that a workload is available in a cluster. If no workload is available, deploy a sample Nginx workload by referring to [Creating a Deployment](#), [Creating a StatefulSet](#), or [Creating a DaemonSet](#).
- A NodePort Service has been configured for the workload. For details about how to configure the Service, see [NodePort](#).

#### networking.k8s.io/v1 Ingress

In CCE clusters of v1.23 or later, ingresses are switched to **networking.k8s.io/v1**.

Compared with v1beta1, v1 has the following differences in parameters:

- The ingress type is changed from **kubernetes.io/ingress.class** in **annotations** to **spec.ingressClassName**.
- The format of **backend** is changed.
- The **pathType** parameter must be specified for each path. The options are as follows:
  - **ImplementationSpecific**: The matching method depends on Ingress Controller. The matching method defined by **ingress.beta.kubernetes.io/url-match-mode** is used in CCE, which is the same as v1beta1.
  - **Exact**: exact matching of the URL, which is case-sensitive.
  - **Prefix**: matching based on the URL prefix separated by a slash (/). The match is case-sensitive, and elements in the path are matched one by one. A path element refers to a list of labels in the path separated by a slash (/).



## Creating an Ingress - Automatically Creating a Load Balancer

The following describes how to run the kubectl command to automatically create a load balancer when creating an ingress.

**Step 1** Use kubectl to connect to the cluster. For details, see [Connecting to a Cluster Using kubectl](#).

**Step 2** Create a YAML file named **ingress-test.yaml**. The file name can be customized.

**vi ingress-test.yaml**

### NOTE

Starting from cluster v1.23, ingresses are switched from **networking.k8s.io/v1beta1** to **networking.k8s.io/v1**. For details about the differences between v1 and v1beta1, see [networking.k8s.io/v1 Ingress](#).

You can create a load balancer as required. The YAML files are as follows:

### Example of a dedicated load balancer (public network access) for clusters of v1.21 or earlier:

```

apiVersion: networking.k8s.io/v1beta1
kind: Ingress
metadata:
  name: ingress-test
  namespace: default
  annotations:
    kubernetes.io/elb.class: performance
    kubernetes.io/ingress.class: cce
    kubernetes.io/elb.port: '80'
    kubernetes.io/elb.autocreate:
      '{
        "type": "public",
        "bandwidth_name": "cce-bandwidth-*****",
        "bandwidth_chargemode": "traffic",
        "bandwidth_size": 5,
        "bandwidth_sharetype": "PER",
        "eip_type": "5_bgp",
        "available_zone": [
          "eu-west-101a"
        ],
        "[7_flavor_name": "L7_flavor.elb.s1.small"
      }'
spec:
  
```

```
rules:
- host: "
  http:
    paths:
    - path: '/'
      backend:
        serviceName: <your_service_name> # Replace it with the name of your target Service.
        servicePort: 80
  property:
    ingress.beta.kubernetes.io/url-match-mode: STARTS_WITH
```

**Table 17-77** Key parameters

Parameter	Mandatory	Type	Description
kubernetes.io/elb.class	Yes	String	Select a proper load balancer type. The value can be: <ul style="list-style-type: none"> <li><b>performance:</b> dedicated load balancer..</li> </ul>
kubernetes.io/ingress.class	Yes (only for clusters of v1.21 or earlier)	String	<b>cce:</b> The self-developed ELB ingress is used. This parameter is mandatory when an ingress is created by calling the API.
ingressClassName	Yes (only for clusters of v1.23 or later)	String	<b>cce:</b> The self-developed ELB ingress is used. Mandatory when an ingress is created by calling the API.
kubernetes.io/elb.port	Yes	Integer	This parameter indicates the external port registered with the address of the LoadBalancer Service. Supported range: 1 to 65535
kubernetes.io/elb.subnet-id	-	String	ID of the subnet where the cluster is located. The value can contain 1 to 100 characters. <ul style="list-style-type: none"> <li>Mandatory when a cluster of v1.11.7-r0 or earlier is to be automatically created.</li> <li>Optional for clusters later than v1.11.7-r0. It is left blank by default.</li> </ul>

Parameter	Mandatory	Type	Description
kubernetes.io/elb.enterpriseID	No	String	<p><b>Kubernetes clusters of v1.15 and later versions support this field. In Kubernetes clusters earlier than v1.15, load balancers are created in the default project by default.</b></p> <p>ID of the enterprise project in which the load balancer will be created.</p> <p>The value contains 1 to 100 characters.</p> <p><b>How to obtain:</b></p> <p>Log in to the management console and choose <b>Enterprise &gt; Project Management</b> on the top menu bar. In the list displayed, click the name of the target enterprise project, and copy the ID on the enterprise project details page.</p>
kubernetes.io/elb.autocreate	Yes	elb.autocreate object	<p>Whether to automatically create a load balancer associated with an ingress. For details about the field description, see <a href="#">Table 17-78</a>.</p> <p><b>Example</b></p> <ul style="list-style-type: none"> <li>If a public network load balancer will be automatically created, set this parameter to the following value:  <pre>{ "type": "public", "bandwidth_name": "cce-bandwidth-*****", "bandwidth_chargemode": "traffic", "bandwidth_size": 5, "bandwidth_sharetype": "PER", "eip_type": "5_bgp", "name": "james" }</pre> </li> <li>If a private network load balancer will be automatically created, set this parameter to the following value:  <pre>{ "type": "inner", "name": "A-location-d-test" }</pre> </li> </ul>

Parameter	Mandatory	Type	Description
host	No	String	Domain name for accessing the Service. By default, this parameter is left blank, and the domain name needs to be fully matched.
path	Yes	String	User-defined route path. All external access requests must match <b>host</b> and <b>path</b> .
serviceName	Yes	String	Name of the target Service bound to the ingress.
servicePort	Yes	Integer	Access port of the target Service.
ingress.beta.kubernetes.io/url-match-mode	No	String	Route matching policy. Default: <b>STARTS_WITH</b> (prefix match) Options: <ul style="list-style-type: none"> <li>• <b>EQUAL_TO</b>: exact match</li> <li>• <b>STARTS_WITH</b>: prefix match</li> <li>• <b>REGEX</b>: regular expression match</li> </ul>
pathType	Yes	String	Path type. This field is supported only by clusters of v1.23 or later. <ul style="list-style-type: none"> <li>• <b>ImplementationSpecific</b>: The matching method depends on Ingress Controller. The matching method defined by <b>ingress.beta.kubernetes.io/url-match-mode</b> is used in CCE.</li> <li>• <b>Exact</b>: exact matching of the URL, which is case-sensitive.</li> <li>• <b>Prefix</b>: matching based on the URL prefix separated by a slash (/). The match is case-sensitive, and elements in the path are matched one by one. A path element refers to a list of labels in the path separated by a slash (/).</li> </ul>

**Table 17-78** Data structure of the elb.autocreate field

Parameter	Mandatory	Type	Description
type	No	String	Network type of the load balancer. <ul style="list-style-type: none"> <li>• <b>public</b>: public network load balancer</li> <li>• <b>inner</b>: private network load balancer</li> </ul> The default value is <b>inner</b> .
bandwidth_name	Yes for public network load balancers	String	Bandwidth name. The default value is <b>cce-bandwidth-*****</b> . Value range: a string of 1 to 64 characters, including lowercase letters, digits, and underscores (_). The value must start with a lowercase letter and end with a lowercase letter or digit.
bandwidth_charge_mode	Yes	String	Bandwidth billing mode. <ul style="list-style-type: none"> <li>• <b>traffic</b>: billed by traffic</li> </ul>
bandwidth_size	Yes for public network load balancers	Integer	Bandwidth size. The value ranges from 1 Mbit/s to 2000 Mbit/s by default. The actual range varies depending on the configuration in each region. <ul style="list-style-type: none"> <li>• The minimum increment for bandwidth adjustment varies depending on the bandwidth range. The details are as follows: <ul style="list-style-type: none"> <li>- The minimum increment is 1 Mbit/s if the allowed bandwidth ranges from 0 Mbit/s to 300 Mbit/s (with 300 Mbit/s included).</li> <li>- The minimum increment is 50 Mbit/s if the allowed bandwidth ranges from 300 Mbit/s to 1000 Mbit/s.</li> <li>- The minimum increment is 500 Mbit/s if the allowed bandwidth is greater than 1000 Mbit/s.</li> </ul> </li> </ul>



Parameter	Mandatory	Type	Description
bandwidth_sharetype	Yes for public network load balancers	String	Bandwidth type. <b>PER</b> : dedicated bandwidth
eip_type	Yes for public network load balancers	String	EIP type <ul style="list-style-type: none"> <li>• <b>5_bgp</b>: dynamic BGP</li> </ul>
name	No	String	Name of the automatically created load balancer. Value range: a string of 1 to 64 characters, including lowercase letters, digits, and underscores (_). The value must start with a lowercase letter and end with a lowercase letter or digit. Default value: <b>cce-lb+ingress.UID</b>
available_zone	Yes	Array of strings	(Mandatory) AZ where the load balancer is located. This parameter is available only for dedicated load balancers.
l4_flavor_name	No	String	Flavor name of the layer-4 load balancer. This parameter is available only for dedicated load balancers.
l7_flavor_name	Yes	String	(Mandatory) Flavor name of the layer-7 load balancer. This parameter is available only for dedicated load balancers.

Parameter	Mandatory	Type	Description
elb_virsubnet_ids	No	Array of strings	Subnet where the backend server of the load balancer is located. If this parameter is left blank, the default cluster subnet is used. Load balancers occupy different number of subnet IP addresses based on their specifications. Therefore, you are not advised to use the subnet CIDR blocks of other resources (such as clusters and nodes) as the load balancer CIDR block.  Default value: subnet where the cluster is located  This parameter is available only for dedicated load balancers.

**Step 3** Create an ingress.

**kubectl create -f ingress-test.yaml**

If information similar to the following is displayed, the ingress has been created.

```
ingress/ingress-test created
```

**kubectl get ingress**

If information similar to the following is displayed, the ingress has been created successfully and the workload is accessible.

```
NAME          HOSTS          ADDRESS          PORTS          AGE
ingress-test  *             121.**.**.**      80            10s
```

**Step 4** Enter **http://121.\*\*.\*\*.\*\*:80** in the address box of the browser to access the workload (for example, [Nginx workload](#)).

**121.\*\*.\*\*.\*\*** indicates the IP address of the unified load balancer.

----End

## Creating an Ingress - Interconnecting with an Existing Load Balancer

CCE allows you to connect to an existing load balancer when creating an ingress.

### NOTE

To interconnect with an existing dedicated load balancer, ensure that HTTP is supported and the network type supports private networks.

**If the cluster version is 1.23 or later, the example YAML file is as follows:**

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: ingress-test
annotations:
```

```
kubernetes.io/elb.id: <your_elb_id> # Replace it with the ID of your existing load balancer.
kubernetes.io/elb.ip: <your_elb_ip> # Replace it with your existing load balancer IP.
kubernetes.io/elb.port: '80'
spec:
  rules:
  - host: ""
    http:
      paths:
      - path: '/'
        backend:
          service:
            name: <your_service_name> # Replace it with the name of your target Service.
            port:
              number: 8080 # Replace 8080 with the port number of your target Service.
          property:
            ingress.beta.kubernetes.io/url-match-mode: STARTS_WITH
            pathType: ImplementationSpecific
  ingressClassName: cce
```

If the cluster version is 1.21 or earlier, the example YAML file is as follows:

```
apiVersion: networking.k8s.io/v1beta1
kind: Ingress
metadata:
  name: ingress-test
  annotations:
    kubernetes.io/elb.id: <your_elb_id> # Replace it with the ID of your existing load balancer.
    kubernetes.io/elb.ip: <your_elb_ip> # Replace it with your existing load balancer IP.
    kubernetes.io/elb.port: '80'
    kubernetes.io/ingress.class: cce
spec:
  rules:
  - host: ""
    http:
      paths:
      - path: '/'
        backend:
          serviceName: <your_service_name> # Replace it with the name of your target Service.
          servicePort: 80
      property:
        ingress.beta.kubernetes.io/url-match-mode: STARTS_WITH
```

**Table 17-79** Key parameters

Parameter	Mandatory	Type	Description
kubernetes.io/elb.class	No	String	Select a proper load balancer type. The value can be: <ul style="list-style-type: none"> <li><b>performance</b>: dedicated load balancer..</li> </ul>
kubernetes.io/elb.id	Yes	String	This parameter indicates the ID of a load balancer. The value can contain 1 to 100 characters. <p><b>How to obtain:</b></p> On the management console, click <b>Service List</b> , and choose <b>Networking &gt; Elastic Load Balance</b> . Click the name of the target load balancer. On the <b>Summary</b> tab page, find and copy the ID.

Parameter	Mandatory	Type	Description
kubernetes.io/elb.ip	Yes	String	This parameter indicates the service address of a load balancer. The value can be the public IP address of a public network load balancer or the private IP address of a private network load balancer.

## Configuring HTTPS Certificates

Ingress supports TLS certificate configuration and provides security services in HTTPS mode.

### NOTE

- If a Service needs to be exposed using HTTPS, you must configure the TLS certificate in the ingress. For details on how to create a secret, see [Creating a Secret](#).
- If HTTPS is used for the same port of the same load balancer of multiple ingresses, you must select the same certificate.

**Step 1** Use `kubectl` to connect to the cluster. For details, see [Connecting to a Cluster Using kubectl](#).

**Step 2** Run the following command to create a YAML file named `ingress-test-secret.yaml` (the file name can be customized):

**vi ingress-test-secret.yaml**

The YAML file is configured as follows:

```
apiVersion: v1
data:
  tls.crt: LS0*****tLS0tCg==
  tls.key: LS0tL*****0tLS0K
kind: Secret
metadata:
  annotations:
    description: test for ingressTLS secrets
    name: ingress-test-secret
    namespace: default
type: IngressTLS
```

### NOTE

In the preceding information, `tls.crt` and `tls.key` are only examples. Replace them with the actual files. The values of `tls.crt` and `tls.key` are the content encrypted using Base64.

**Step 3** Create a secret.

**kubectl create -f ingress-test-secret.yaml**

If information similar to the following is displayed, the secret is being created:

```
secret/ingress-test-secret created
```

View the created secrets.

**kubectl get secrets**

If information similar to the following is displayed, the secret has been created successfully:

NAME	TYPE	DATA	AGE
ingress-test-secret	IngressTLS	2	13s

**Step 4** Create a YAML file named **ingress-test.yaml**. The file name can be customized.

**vi ingress-test.yaml**

 **NOTE**

Security policy (kubernetes.io/elb.tls-ciphers-policy) is supported only in clusters of v1.17.11 or later.

**Example YAML file to associate an existing load balancer:**

```
apiVersion: networking.k8s.io/v1beta1
kind: Ingress
metadata:
  name: ingress-test
  annotations:
    kubernetes.io/elb.class: performance # Load balancer type
    kubernetes.io/elb.id: <your_elb_id> # Replace it with the ID of your existing load balancer.
    kubernetes.io/elb.ip: <your_elb_ip> # Replace it with the IP of your existing load balancer.
    kubernetes.io/ingress.class: cce
    kubernetes.io/elb.port: '443'
    kubernetes.io/elb.tls-ciphers-policy: tls-1-2
spec:
  tls:
  - secretName: ingress-test-secret
  rules:
  - host: ""
    http:
      paths:
      - path: '/'
        backend:
          serviceName: <your_service_name> # Replace it with the name of your target Service.
          servicePort: 80
    property:
      ingress.beta.kubernetes.io/url-match-mode: STARTS_WITH
```

**Table 17-80** Key parameters

Parameter	Mandatory	Type	Description
kubernetes.io/elb.tls-ciphers-policy	No	String	The default value is <b>tls-1-2</b> , which is the security policy used by the listener and takes effect only when the HTTPS protocol is used.  Options: <ul style="list-style-type: none"> <li>• tls-1-0</li> <li>• tls-1-1</li> <li>• tls-1-2</li> <li>• tls-1-2-strict</li> </ul> For details of cipher suites for each security policy, see <a href="#">Table 17-81</a> .

Parameter	Mandatory	Type	Description
tls	No	Array of strings	This parameter is mandatory if HTTPS is used. Multiple independent domain names and certificates can be added to this parameter. For details, see <a href="#">Configuring the Server Name Indication (SNI)</a> .
secretName	No	String	This parameter is mandatory if HTTPS is used. Set this parameter to the name of the created secret.

**Table 17-81** tls\_ciphers\_policy parameter description

Security Policy	TLS Version	Cipher Suite
tls-1-0	TLS 1.2 TLS 1.1 TLS 1.0	ECDHE-RSA-AES256-GCM-SHA384:ECDHE-RSA-AES128-GCM-SHA256:ECDHE-ECDSA-AES256-GCM-SHA384:ECDHE-ECDSA-AES128-GCM-SHA256:AES128-GCM-SHA256:AES256-GCM-SHA384:ECDHE-ECDSA-AES128-SHA256:ECDHE-RSA-AES128-SHA256:AES128-SHA256:AES256-SHA256:ECDHE-ECDSA-AES256-SHA384:ECDHE-RSA-AES256-SHA384:ECDHE-ECDSA-AES128-SHA:ECDHE-RSA-AES128-SHA:ECDHE-RSA-AES256-SHA:ECDHE-ECDSA-AES256-SHA:AES128-SHA:AES256-SHA
tls-1-1	TLS 1.2 TLS 1.1	ECDHE-RSA-AES256-GCM-SHA384:ECDHE-ECDSA-AES128-SHA256:AES128-GCM-SHA256:AES256-GCM-SHA384:ECDHE-ECDSA-AES128-SHA256:ECDHE-RSA-AES128-SHA256:AES128-SHA256:AES256-SHA256:ECDHE-ECDSA-AES256-SHA384:ECDHE-RSA-AES256-SHA384:ECDHE-ECDSA-AES128-SHA:ECDHE-RSA-AES128-SHA:ECDHE-RSA-AES256-SHA:ECDHE-ECDSA-AES256-SHA:AES128-SHA:AES256-SHA
tls-1-2	TLS 1.2	ECDHE-RSA-AES256-GCM-SHA384:ECDHE-RSA-AES128-GCM-SHA256:ECDHE-ECDSA-AES256-GCM-SHA384:ECDHE-ECDSA-AES128-GCM-SHA256:AES128-GCM-SHA256:AES256-GCM-SHA384:ECDHE-ECDSA-AES128-SHA256:ECDHE-RSA-AES128-SHA256:AES128-SHA256:AES256-SHA256:ECDHE-ECDSA-AES256-SHA384:ECDHE-RSA-AES256-SHA384:ECDHE-ECDSA-AES128-SHA:ECDHE-RSA-AES128-SHA:ECDHE-RSA-AES256-SHA:ECDHE-ECDSA-AES256-SHA:AES128-SHA:AES256-SHA
tls-1-2-strict	TLS 1.2	ECDHE-RSA-AES256-GCM-SHA384:ECDHE-RSA-AES128-GCM-SHA256:ECDHE-ECDSA-AES256-GCM-SHA384:ECDHE-ECDSA-AES128-GCM-SHA256:AES128-GCM-SHA256:AES256-GCM-SHA384:ECDHE-ECDSA-AES128-SHA256:ECDHE-RSA-AES128-SHA256:AES128-SHA256:AES256-SHA256:ECDHE-ECDSA-AES256-SHA384:ECDHE-RSA-AES256-SHA384

**Step 5** Create an ingress.

**kubectl create -f ingress-test.yaml**

If information similar to the following is displayed, the ingress has been created.

```
ingress/ingress-test created
```

View the created ingress.

## kubectl get ingress

If information similar to the following is displayed, the ingress has been created successfully and the workload is accessible.

NAME	HOSTS	ADDRESS	PORTS	AGE
ingress-test	*	121.**.**	80	10s

**Step 6** Enter **https://121.\*\*.\*\*.443** in the address box of the browser to access the workload (for example, [Nginx workload](#)).

**121.\*\*.\*\*** indicates the IP address of the unified load balancer.

----End

## Configuring the Server Name Indication (SNI)

SNI allows multiple TLS-based access domain names to be provided for external systems using the same IP address and port number. Different domain names can use different security certificates.

### NOTE

- Only one domain name can be specified for each SNI certificate. Wildcard-domain certificates are supported.
- Security policy (kubernetes.io/elb.tls-ciphers-policy) is supported only in clusters of v1.17.11 or later.

You can enable SNI when the preceding conditions are met. The following uses the automatic creation of a load balancer as an example. In this example, **sni-test-secret-1** and **sni-test-secret-2** are SNI certificates. The domain names specified by the certificates must be the same as those in the certificates.

```
apiVersion: networking.k8s.io/v1beta1
kind: Ingress
metadata:
  name: ingress-test
  annotations:
    kubernetes.io/elb.class: performance # Load balancer type
    kubernetes.io/elb.id: <your_elb_id> # Replace it with the ID of your existing load balancer.
    kubernetes.io/elb.ip: <your_elb_ip> # Replace it with the IP of your existing load balancer.
    kubernetes.io/ingress.class: cce
    kubernetes.io/elb.port: '443'
    kubernetes.io/elb.tls-ciphers-policy: tls-1-2
spec:
  tls:
  - secretName: ingress-test-secret
  - hosts:
    - example.top # Domain name specified when a certificate is issued
      secretName: sni-test-secret-1
    - hosts:
    - example.com # Domain name specified when a certificate is issued
      secretName: sni-test-secret-2
  rules:
  - host: ""
    http:
      paths:
      - path: '/'
        backend:
          serviceName: <your_service_name> # Replace it with the name of your target Service.
          servicePort: 80
    property:
      ingress.beta.kubernetes.io/url-match-mode: STARTS_WITH
```

## Accessing Multiple Services

Ingresses can route requests to multiple backend Services based on different matching policies. The `spec` field in the YAML file is set as below. You can access `www.example.com/foo`, `www.example.com/bar`, and `foo.example.com/` to route to three different backend Services.

### NOTICE

The URL registered in an ingress forwarding policy must be the same as the URL exposed by the backend Service. Otherwise, a 404 error will be returned.

```
spec:
  rules:
  - host: 'www.example.com'
    http:
      paths:
      - path: '/foo'
        backend:
          serviceName: <your_service_name> # Replace it with the name of your target Service.
          servicePort: 80
        property:
          ingress.beta.kubernetes.io/url-match-mode: STARTS_WITH
      - path: '/bar'
        backend:
          serviceName: <your_service_name> # Replace it with the name of your target Service.
          servicePort: 80
        property:
          ingress.beta.kubernetes.io/url-match-mode: STARTS_WITH
  - host: 'foo.example.com'
    http:
      paths:
      - path: '/'
        backend:
          serviceName: <your_service_name> # Replace it with the name of your target Service.
          servicePort: 80
        property:
          ingress.beta.kubernetes.io/url-match-mode: STARTS_WITH
```

### 17.8.4.4 Using Nginx Ingresses on the Console

#### Prerequisites

- An ingress provides network access for backend workloads. Ensure that a workload is available in a cluster. If no workload is available, deploy a workload by referring to [Creating a Deployment](#), [Creating a StatefulSet](#), or [Creating a DaemonSet](#).

#### Precautions

- **Do not modify any configuration of a load balancer on the ELB console. Otherwise, the Service will be abnormal.** If you have modified the configuration, uninstall the nginx-ingress add-on and reinstall it.
- The URL registered in an ingress forwarding policy must be the same as the URL exposed by the backend Service. Otherwise, a 404 error will be returned.
- The selected or created load balancer must be in the same VPC as the current cluster, and it must match the load balancer type (private or public network).



- The load balancer has at least two listeners, and ports 80 and 443 are not occupied by listeners.

## Creating an Nginx Ingress

This section uses an Nginx workload as an example to describe how to create an Nginx ingress.


**Step 1** Log in to the CCE console.

**Step 2** In the navigation pane, choose **Resource Management > Network**. On the **Ingresses** tab page, select the corresponding cluster and namespace.

**Step 3** Click **Create Ingress** to access the ingress configuration page.

Set the ingress parameters as required. The key parameters are as follows:

- **Access Type:** Use a load balancer to access Services.
- **Ingress Name:** Specify a name of an ingress, for example, **nginx-ingress-demo**.
- **Cluster Name:** Select the cluster to which the ingress is to be added.
- **Namespace:** Select the namespace to which the ingress is to be added.
- **nginx-ingress:** This option is displayed only when the **nginx-ingress** add-on has been installed in the cluster.

After you switch on , nginx-ingress is interconnected to provide layer-7 access. You can configure the following parameters:

**Table 17-82** Nginx configuration parameters

Parameter	Description
Front-End Protocol	HTTP and HTTPS are supported.
External Port	Listening port reserved for installing the nginx-ingress add-on. The port number is 80 for HTTP and 443 for HTTPS.
Server Certificate	When creating an HTTPS listener, you need to bind an IngressTLS certificate to support encrypted authentication for HTTPS data transmission. For details on how to create a secret, see <a href="#">Creating a Secret</a> .
Timeout	Timeout period that the client establishes a connection with the proxy server.
Redirected To	Address to which all workload content is to be redirected, for example, <a href="https://www.example.com/">https://www.example.com/</a> .
Custom Settings	The value is in the format of key:value. You can use <a href="#">annotations</a> to query the configurations supported by nginx-ingress.

- **Forwarding Policies:** When the access address of a request matches the forwarding policy (a forwarding policy consists of a domain name and URL),

the request is forwarded to the corresponding target Service for processing. Click **Add Forwarding Policies** to add multiple forwarding policies.

- **Domain Name:** actual domain name. Ensure that the entered domain name has been registered and archived. After the ingress is created, bind the domain name to the IP address of the automatically created load balancer (IP address of the ingress access address). If a domain name rule is configured, the domain name must always be used for access.
- Rule Matching
  - **Prefix match:** If the URL is set to `/healthz`, the URL that meets the prefix can be accessed. For example, `/healthz/v1` and `/healthz/v2`.
  - **Exact match:** The URL can be accessed only when it is fully matched. For example, if the URL is set to `/healthz`, only `/healthz` can be accessed.
  - **Regular expression:** The URL is matched based on the regular expression. For example, if the regular expression is `/[A-Za-z0-9_-.]+/test`, all URLs that comply with this rule can be accessed, for example, `/abcA9/test` and `/v1-Ab/test`. Two regular expression standards are supported: POSIX and Perl.
- **URL:** access path to be registered, for example, `/healthz`.
- **Target Service:** Select an existing Service or create a Service. Services that do not meet search criteria are automatically filtered out.
- **Service Access Port:** Select the access port of the target Service.
- **Operation:** Click **Delete** to delete the configuration.

**Step 4** After the configuration is complete, click **Create**.

After the ingress is created, it is displayed in the ingress list.

----End

## Updating an Ingress

After adding an ingress, you can update its port, domain name, and route configuration.

### NOTE

You can modify the load balancer settings, including algorithm, sticky session, and health check configurations, after you select a Service in **Forwarding Policies** on the CCE console. Do not modify these configurations on the ELB console.

**Step 1** Log in to the CCE console. In the navigation pane, choose **Resource Management** > **Network**. On the **Ingresses** tab page, filter ingresses by cluster and namespace, and click **Update** for the ingress to be updated.

**Step 2** On the **Update Ingress** page, modify the required parameters.

**Step 3** Click **Submit** to update the ingress.

----End

## 17.8.5 DNS

### 17.8.5.1 Overview

#### Introduction to CoreDNS

When you purchase a cluster, the **coredns add-on** is installed to resolve domain names in the cluster.

You can view the pod of the coredns add-on in the kube-system namespace.

```
$ kubectl get po --namespace=kube-system
NAME                                READY STATUS RESTARTS AGE
coredns-7689f8bdf-295rk             1/1   Running 0      9m11s
coredns-7689f8bdf-h7n68            1/1   Running 0      11m
```

After coredns is installed, it becomes a DNS. After the Service is created, coredns records the Service name and IP address. In this way, the pod can obtain the Service IP address by querying the Service name from coredns.

**nginx.<namespace>.svc.cluster.local** is used to access the Service. **nginx** is the Service name, **<namespace>** is the namespace, and **svc.cluster.local** is the domain name suffix. In actual use, you can omit **<namespace>.svc.cluster.local** in the same namespace and use the ServiceName.

An advantage of using ServiceName is that you can write ServiceName into the program when developing the application. In this way, you do not need to know the IP address of a specific Service.

After the coredns add-on is installed, there is also a Service in the kube-system namespace, as shown below.

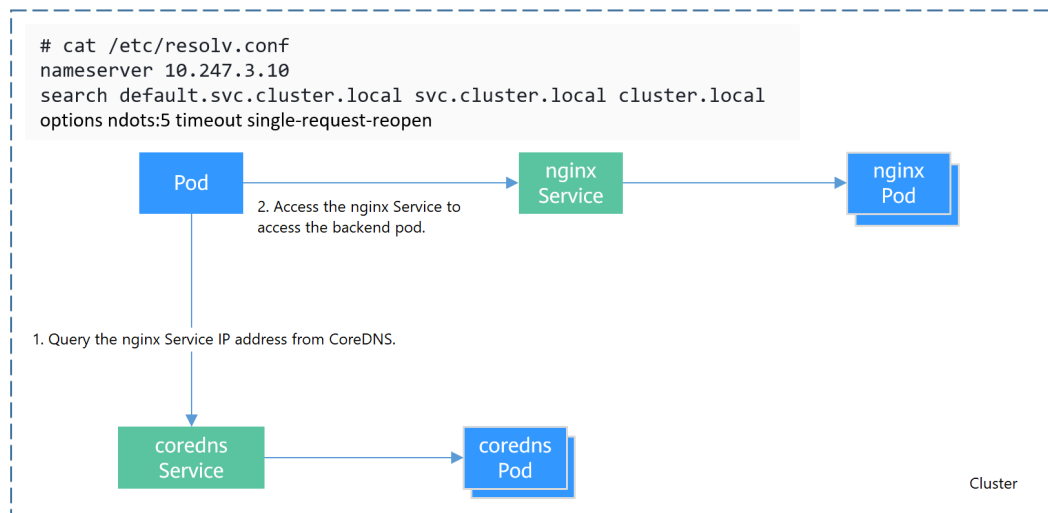
```
$ kubectl get svc -n kube-system
NAME      TYPE      CLUSTER-IP   EXTERNAL-IP   PORT(S)              AGE
coredns   ClusterIP  10.247.3.10  <none>        53/UDP,53/TCP,8080/TCP  13d
```

By default, after other pods are created, the address of the coredns Service is written as the address of the domain name resolution server in the **/etc/resolv.conf** file of the pod. Create a pod and view the **/etc/resolv.conf** file as follows:

```
$ kubectl exec test01-6cbbf97b78-krj6h -it -- /bin/sh
/ # cat /etc/resolv.conf
nameserver 10.247.3.10
search default.svc.cluster.local svc.cluster.local cluster.local
options ndots:5 timeout single-request-reopen
```

When a user accesses the *Service name:Port* of the Nginx pod, the IP address of the Nginx Service is resolved from CoreDNS, and then the IP address of the Nginx Service is accessed. In this way, the user can access the backend Nginx pod.

**Figure 17-131** Example of domain name resolution in a cluster



## Related Operations

You can also configure DNS in a workload. For details, see [DNS Configuration](#).

You can also use DNSCache to improve the DNS resolution performance. For details, see [Using NodeLocal DNSCache to Improve DNS Performance](#).

### 17.8.5.2 DNS Configuration

Every Kubernetes cluster has a built-in DNS add-on (CoreDNS) to provide domain name resolution for workloads in the cluster. When handling a high concurrency of DNS queries, CoreDNS may encounter a performance bottleneck, that is, it may fail occasionally to fulfill DNS queries. There are cases when Kubernetes workloads initiate unnecessary DNS queries. This makes DNS overloaded if there are many concurrent DNS queries. Tuning DNS configuration for workloads will reduce the risks of DNS query failures to some extent.

For more information about DNS, see [coredns \(System Resource Add-on, Mandatory\)](#).

## DNS Configuration Items

Run the `cat /etc/resolv.conf` command on a Linux node or container to view the DNS resolver configuration file. The following is an example DNS resolver configuration of a container in a Kubernetes cluster:

```

nameserver 10.247.x.x
search default.svc.cluster.local svc.cluster.local cluster.local
options ndots:5
    
```

### Configuration Options

- **nameserver:** an IP address list of a name server that the resolver will query. If this parameter is set to 10.247.x.x, the resolver will query the kube-dns/CoreDNS. If this parameter is set to another IP address, the resolver will query a cloud or on-premises DNS server.
- **search:** a search list for host-name lookup. When a domain name cannot be resolved, DNS queries will be attempted combining the domain name with

each domain in the search list in turn until a match is found or all domains in the search list are tried. For CCE clusters, the search list is currently limited to three domains per container. When a nonexistent domain name is being resolved, eight DNS queries will be initiated because each domain name (including those in the search list) will be queried twice, one for IPv4 and the other for IPv6.

- **options:** options that allow certain internal resolver variables to be modified. Common options include timeout and ndots.

The value **ndots:5** means that if a domain name has fewer than 5 dots (.), DNS queries will be attempted by combining the domain name with each domain in the search list in turn. If no match is found after all the domains in the search list are tried, the domain name is then used for DNS query. If the domain name has 5 or more than 5 dots, it will be tried first for DNS query. In case that the domain name cannot be resolved, DNS queries will be attempted by combining the domain name with each domain in the search list in turn.

For example, the domain name **www.\*\*\*.com** has only two dots (smaller than the value of **ndots**), and therefore the sequence of DNS queries is as follows: **www.\*\*\*.com.default.svc.cluster.local**, **www.\*\*\*.com.svc.cluster.local**, **www.\*\*\*.com.cluster.local**, and **www.\*\*\*.com**. This means that at least seven DNS queries will be initiated before the domain name is resolved into an IP address. It is clear that when many unnecessary DNS queries will be initiated to access an external domain name. There is room for improvement in workload's DNS configuration.

#### NOTE

For more information about configuration options in the resolver configuration file used by Linux operating systems, visit <http://man7.org/linux/man-pages/man5/resolv.conf.5.html>.

## Configuring DNS Using the Workload YAML

When creating a workload using a YAML file, you can configure the DNS settings in the YAML. The following is an example for an Nginx application:

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx
  namespace: default
spec:
  replicas: 1
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
        - name: container-1
          image: nginx:latest
          imagePullPolicy: IfNotPresent
      imagePullSecrets:
        - name: default-secret
      dnsPolicy: None
      dnsConfig:
        options:
```

```
- name: ndots
  value: '5'
- name: timeout
  value: '3'
nameservers:
- 10.2.3.4
searches:
- my.dns.search.suffix
```

- **dnsPolicy**

The **dnsPolicy** field is used to configure a DNS policy for an application. The default value is **ClusterFirst**. The following table lists **dnsPolicy** configurations.

**Table 17-83** dnsPolicy

Parameter	Description
ClusterFirst (default value)	Custom DNS configuration added to the default DNS configuration. By default, the application connects to CoreDNS (CoreDNS of the CCE cluster connects to the DNS on the cloud by default). The custom dnsConfig will be added to the default DNS parameters. Containers can resolve both the cluster-internal domain names registered by a Service and the external domain names exposed to public networks. The search list ( <b>search</b> option) and <b>ndots: 5</b> are present in the DNS configuration file. Therefore, when accessing an external domain name and a long cluster-internal domain name (for example, <code>kubernetes.default.svc.cluster.local</code> ), the search list will usually be traversed first, resulting in at least six invalid DNS queries. The issue of invalid DNS queries disappears only when a short cluster-internal domain name (for example, <code>kubernetes</code> ) is being accessed.
ClusterFirstWithHostNet	By default, the applications configured with the host network are interconnected with the DNS configuration of the node where the pod is located. The DNS configuration is specified in the DNS file that the kubelet <b>--resolv-conf</b> parameter points to. In this case, the CCE cluster uses the DNS on the cloud. If workloads need to use Kube-DNS/ CoreDNS of the cluster, set <b>dnsPolicy</b> to <b>ClusterFirstWithHostNet</b> and container's DNS configuration file is the same as ClusterFirst, in which invalid DNS queries still exist. ... spec: containers: - image: nginx:latest imagePullPolicy: IfNotPresent name: container-1 restartPolicy: Always <b>hostNetwork: true</b> <b>dnsPolicy: ClusterFirstWithHostNet</b>

Parameter	Description
Default	The DNS configuration of the node where the pod is located is inherited, and the custom DNS configuration is added to the inherited configuration. Container's DNS configuration file is the DNS configuration file that the kubelet's <b>--resolv-conf</b> flag points to. In this case, a cloud DNS is used for CCE clusters. Both <b>search</b> and <b>options</b> fields are left unspecified. This configuration can only resolve the external domain names registered with the Internet, and not cluster-internal domain names. This configuration is free from the issue of invalid DNS queries.
None	The default DNS configuration is replaced by the custom DNS configuration, and only the custom DNS configuration is used. If <b>dnsPolicy</b> is set to <b>None</b> , the <b>dnsConfig</b> field must be specified because all DNS settings are supposed to be provided using the <b>dnsConfig</b> field.

 NOTE

If the **dnsPolicy** field is not specified, the default value is **ClusterFirst** instead of **Default**.

- **dnsConfig**

The **dnsConfig** field is used to configure DNS parameters for workloads. The configured parameters are merged to the DNS configuration file generated according to **dnsPolicy**. If **dnsPolicy** is set to **None**, the workload's DNS configuration file is specified by the **dnsConfig** field. If **dnsPolicy** is not set to **None**, the DNS parameters configured in **dnsConfig** are added to the DNS configuration file generated according to **dnsPolicy**.

**Table 17-84** dnsConfig

Parameter	Description
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 in <b>dnsPolicy</b> . Duplicate entries are removed.

Parameter	Description
nameservers	<p>A list of IP addresses that will be used as DNS servers. If workload's <b>dnsPolicy</b> is set to <b>None</b>, the list must contain at least one IP address, otherwise this property is optional. The servers listed will be combined to the nameservers generated from the specified DNS policy in <b>dnsPolicy</b> with duplicate addresses removed.</p> <p><b>NOTE</b> A maximum of three DNS addresses can be configured for a nameserver in the container DNS configuration file.</p> <ul style="list-style-type: none"> <li>• If <b>dnsPolicy</b> is set to <b>ClusterFirst</b> and the cluster uses <b>CoreDNS</b>, you can add two custom DNS addresses in addition to the CoreDNS address. Excess DNS addresses are invalid.</li> <li>• If <b>dnsPolicy</b> is set to <b>ClusterFirst</b> and the cluster uses <b>CoreDNS</b> and <b>NodeLocal DNSCache</b>, you can add one custom DNS address in addition to the CoreDNS and NodeLocal DNSCache addresses. Excess DNS addresses are invalid.</li> </ul>
searches	<p>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 search domain names generated from the chosen DNS policy in <b>dnsPolicy</b>. Duplicate domain names are removed. Kubernetes allows for at most 6 search domains.</p>

## Configuration Examples

The following example describes how to configure DNS for workloads.

- **Use Case 1: Using Kube-DNS/CoreDNS Built in Kubernetes Clusters**

### Scenario

Kubernetes in-cluster Kube-DNS/CoreDNS applies to resolving only cluster-internal domain names or cluster-internal domain names + external domain names. This is the default DNS for workloads.

### Example:

```
apiVersion: v1
kind: Pod
metadata:
  namespace: default
  name: dns-example
spec:
  containers:
  - name: test
    image: nginx:alpine
    dnsPolicy: ClusterFirst
  imagePullSecrets:
  - name: default-secret
```

Container's DNS configuration file:

```
nameserver 10.247.3.10
search default.svc.cluster.local svc.cluster.local cluster.local
options ndots:5
```

- **Use Case 2: Using a Cloud DNS**

### Scenario



A DNS cannot resolve cluster-internal domain names and therefore applies to the scenario where workloads access only external domain names registered with the Internet.

**Example:**

```
apiVersion: v1
kind: Pod
metadata:
  namespace: default
  name: dns-example
spec:
  containers:
  - name: test
    image: nginx:alpine
    dnsPolicy: Default # The DNS configuration file that the kubelet --resolv-conf parameter points to
    is used. In this case, the CCE cluster uses the DNS on the cloud.
  imagePullSecrets:
  - name: default-secret
```

Container's DNS configuration file:

```
nameserver 100.125.x.x
```

- **Use Case 3: Using Kube-DNS/CoreDNS for Workloads Running with hostNetwork**

**Scenario**

By default, a DNS is used for workloads running with hostNetwork. If workloads need to use Kube-DNS/CoreDNS, set **dnsPolicy** to **ClusterFirstWithHostNet**.

**Example:**

```
apiVersion: v1
kind: Pod
metadata:
  name: nginx
spec:
  hostNetwork: true
  dnsPolicy: ClusterFirstWithHostNet
  containers:
  - name: nginx
    image: nginx:alpine
    ports:
    - containerPort: 80
  imagePullSecrets:
  - name: default-secret
```

Container's DNS configuration file:

```
nameserver 10.247.3.10
search default.svc.cluster.local svc.cluster.local cluster.local
options ndots:5
```

- **Use Case 4: Customizing Application's DNS Configuration**

**Scenario**

You can flexibly customize the DNS configuration file for applications. Using **dnsPolicy** and **dnsConfig** together can address almost all scenarios, including the scenarios in which an on-premises DNS will be used, multiple DNSs will be cascaded, and DNS configuration options will be modified.

**Example 1: Using Your On-Premises DNS**

*Set **dnsPolicy** to **None** so application's DNS configuration file is generated based on **dnsConfig**.*

```
apiVersion: v1
kind: Pod
metadata:
```

```
namespace: default
name: dns-example
spec:
  containers:
  - name: test
    image: nginx:alpine
  dnsPolicy: "None"
  dnsConfig:
    nameservers:
    - 10.2.3.4 # IP address of your on-premises DNS
    searches:
    - ns1.svc.cluster.local
    - my.dns.search.suffix
    options:
    - name: ndots
      value: "2"
    - name: timeout
      value: "3"
  imagePullSecrets:
  - name: default-secret
```

Container's DNS configuration file:

```
nameserver 10.2.3.4
search ns1.svc.cluster.local my.dns.search.suffix
options timeout:3 ndots:2
```

### Example 2: Modifying the ndots Option in the DNS Configuration File to Reduce Invalid DNS Queries

Set **dnsPolicy** to a value other than **None** so the DNS parameters configured in **dnsConfig** are added to the DNS configuration file generated based on **dnsPolicy**.

```
apiVersion: v1
kind: Pod
metadata:
  namespace: default
  name: dns-example
spec:
  containers:
  - name: test
    image: nginx:alpine
  dnsPolicy: "ClusterFirst"
  dnsConfig:
    options:
    - name: ndots
      value: "2" # The ndots:5 option in the DNS configuration file generated based on the
ClusterFirst policy is changed to ndots:2.
  imagePullSecrets:
  - name: default-secret
```

Container's DNS configuration file:

```
nameserver 10.247.3.10
search default.svc.cluster.local svc.cluster.local cluster.local
options ndots:2
```

### Example 3: Using Multiple DNSs in Serial Sequence

```
apiVersion: v1
kind: Pod
metadata:
  namespace: default
  name: dns-example
spec:
  containers:
  - name: test
    image: nginx:alpine
  dnsPolicy: ClusterFirst # Added DNS configuration. The cluster connects to CoreDNS by default.
  dnsConfig:
    nameservers:
    - 10.2.3.4 # IP address of your on-premises DNS
```

```
imagePullSecrets:  
- name: default-secret
```

#### NOTE

A maximum of three DNS addresses can be configured for a nameserver in the container DNS configuration file.

- If **dnsPolicy** is set to **ClusterFirst** and the cluster uses **CoreDNS**, you can add two custom DNS addresses in addition to the CoreDNS address. Excess DNS addresses are invalid.
- If **dnsPolicy** is set to **ClusterFirst** and the cluster uses **CoreDNS** and **NodeLocal DNSCache**, you can add one custom DNS address in addition to the CoreDNS and NodeLocal DNSCache addresses. Excess DNS addresses are invalid.

Container's DNS configuration file:

```
nameserver 10.247.3.10 10.2.3.4  
search default.svc.cluster.local svc.cluster.local cluster.local  
options ndots:5
```

### 17.8.5.3 Using CoreDNS for Custom Domain Name Resolution

#### Challenges

When using CCE, you may need to resolve custom internal domain names in the following scenarios:

- In the legacy code, a fixed domain name is configured for calling other internal services. If the system decides to use Kubernetes Services, the code refactoring workload could be heavy.
- A service is created outside the cluster. Data in the cluster needs to be sent to the service through a fixed domain name.

#### Solutions

There are several CoreDNS-based solutions for custom domain name resolution:

- **Configuring the Stub Domain for CoreDNS:** You can add it on the console, which is easy to operate.
- **Using the CoreDNS Hosts plug-in to configure resolution for any domain name:** You can add any record set, which is similar to adding a record set in the local **/etc/hosts** file.
- **Using the CoreDNS Rewrite plug-in to point a domain name to a service in the cluster:** A nickname is assigned to the Kubernetes Service. You do not need to know the IP address of the resolution record in advance.
- **Using the CoreDNS Forward plug-in to set the self-built DNS as the upstream DNS:** The self-built DNS can manage a large number of resolution records. You do not need to modify the CoreDNS configuration when adding or deleting records.

#### Precautions

Improper modification on CoreDNS configuration may cause domain name resolution failures in the cluster. Perform tests before and after the modification.

## Configuring the Stub Domain for CoreDNS

Cluster administrators can modify the ConfigMap for the CoreDNS Corefile to change how service discovery works. They can configure stub domains for CoreDNS using the proxy plug-in.

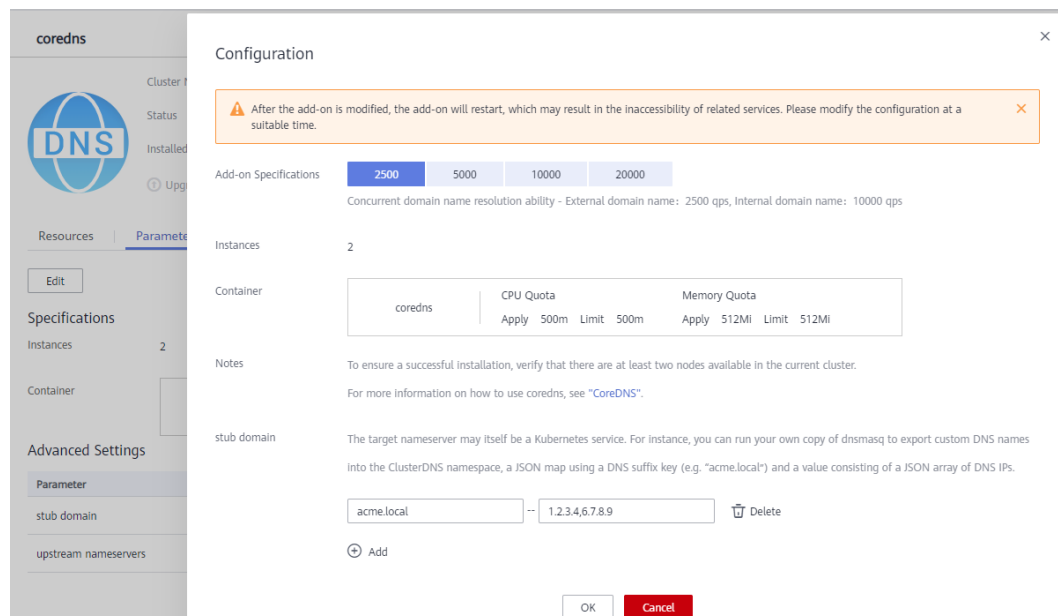
Assume that a cluster administrator has a Consul DNS server located at 10.150.0.1 and all Consul domain names have the suffix **.consul.local**.

**Step 1** Log in to the CCE console.

**Step 2** In the navigation pane on the left, choose **Add-ons**. On the **Add-on Instance** tab page, click **coredns**.

**Step 3** Choose the **Parameters** tab page and click **Edit** to add a stub domain.

**Figure 17-132** Configuring the stub domain



**Step 4** Click **OK**.

-----End

## Modifying the CoreDNS Hosts Configuration File

**Step 1** Use kubectl to connect to the cluster.

**Step 2** Modify the CoreDNS configuration file and add the custom domain name to the hosts file.

Point **www.example.com** to **192.168.1.1**. When CoreDNS resolves **www.example.com**, **192.168.1.1** is returned.

**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 CoreDNS plug-in. 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/>.

```
$ kubectl edit configmap coredns -n kube-system
apiVersion: v1
data:
  Corefile: |-
    .: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
      }
      hosts {
        192.168.1.1 www.example.com
        fallthrough
      }
      loadbalance round_robin
      prometheus {$POD_IP}:9153
      forward . /etc/resolv.conf
      reload
    }
kind: ConfigMap
metadata:
  creationTimestamp: "2021-08-23T13:27:28Z"
  labels:
    app: coredns
    k8s-app: coredns
    kubernetes.io/cluster-service: "true"
    kubernetes.io/name: CoreDNS
  release: cceaddon-coredns
  name: coredns
  namespace: kube-system
  resourceVersion: "460"
  selfLink: /api/v1/namespaces/kube-system/configmaps/coredns
  uid: be64aaad-1629-441f-8a40-a3efc0db9fa9
```

After modifying the hosts file in CoreDNS, you do not need to configure the hosts file in each pod.

----End

## Adding the CoreDNS Rewrite Configuration to Point the Domain Name to Services in the Cluster

Use the Rewrite plug-in of CoreDNS to resolve a specified domain name to the domain name of a Service.

- Step 1** Use `kubectl` to connect to the cluster.
- Step 2** Modify the CoreDNS configuration file to point **example.com** to the **example** service in the **default** namespace.

```
$ kubectl edit configmap coredns -n kube-system
apiVersion: v1
data:
  Corefile: |-
    .: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
      }
      rewrite name example.com example.default.svc.cluster.local
      loadbalance round_robin
      prometheus {$POD_IP}:9153
      forward . /etc/resolv.conf
      reload
    }
kind: ConfigMap
metadata:
  creationTimestamp: "2021-08-23T13:27:28Z"
  labels:
    app: coredns
    k8s-app: coredns
    kubernetes.io/cluster-service: "true"
    kubernetes.io/name: CoreDNS
  release: cceaddon-coredns
  name: coredns
  namespace: kube-system
  resourceVersion: "460"
  selfLink: /api/v1/namespaces/kube-system/configmaps/coredns
  uid: be64aaad-1629-441f-8a40-a3efc0db9fa9
```

----End

## Using CoreDNS to Cascade Self-Built DNS

**Step 1** Use kubectl to connect to the cluster.

**Step 2** Modify the CoreDNS configuration file and change **/etc/resolv.conf** following **forward** to the IP address of the external DNS server.

```
$ kubectl edit configmap coredns -n kube-system
apiVersion: v1
data:
  Corefile: |-
    .: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 . 192.168.1.1
      reload
    }
kind: ConfigMap
metadata:
  creationTimestamp: "2021-08-23T13:27:28Z"
  labels:
    app: coredns
    k8s-app: coredns
    kubernetes.io/cluster-service: "true"
    kubernetes.io/name: CoreDNS
```

```
release: cceaddon-coredns
name: coredns
namespace: kube-system
resourceVersion: "460"
selfLink: /api/v1/namespaces/kube-system/configmaps/coredns
uid: be64aaad-1629-441f-8a40-a3efc0db9fa9
```

----End

### 17.8.5.4 Using NodeLocal DNSCache to Improve DNS Performance

#### Challenges

During DNS resolution, if there are a large number of requests, CoreDNS will be under pressure, which has the following impacts:

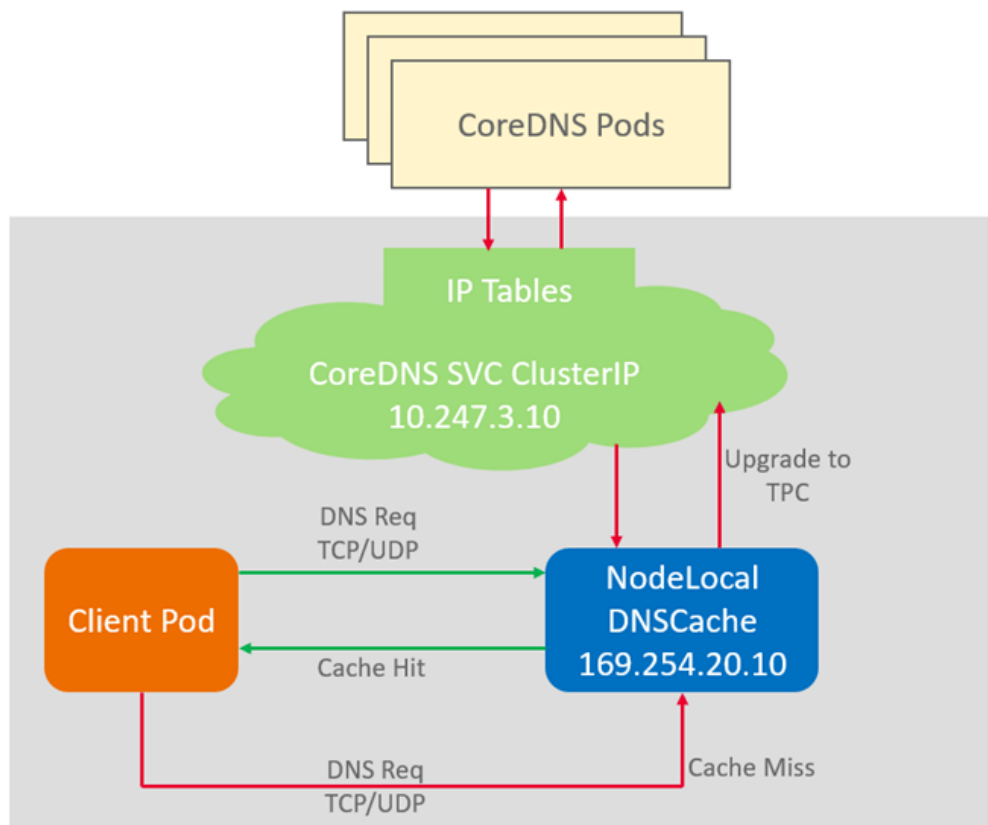
- The query becomes slow, affecting service performance.
- CoreDNS requires higher specifications.

#### Solutions

NodeLocal DNSCache improves cluster DNS performance by running DNS cache proxies on cluster nodes.

After NodeLocal DNSCache is enabled, a DNS query goes through the path as shown below.

Figure 17-133 NodeLocal DNSCache query path



## Installing the Add-on

CCE provides add-on **node-local-dns** for you to install NodeLocal DNSCache.

### NOTE

- The node-local-dns add-on supports only clusters of v1.19 and later.
- NodeLocal DNSCache serves as a transparent caching proxy for CoreDNS and does not provide plug-ins such as hosts or rewrite. If you want to enable these plug-ins, modify the CoreDNS configurations.
- Pods cannot be automatically injected into the kube-system namespace.

**Step 1** (Optional) Modify the CoreDNS configuration so that the CoreDNS preferentially uses UDP to communicate with the upstream DNS server.

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. However, the cloud server does not support TCP. To use NodeLocal DNSCache, you need to modify the CoreDNS configuration so that UDP is preferentially used to communicate with the upstream DNS server, preventing resolution exceptions.

Run the following command:

```
kubectl edit configmap coredns -nkube-system
```

In the forward plug-in, specify **prefer\_udp** as the protocol used by requests. After the modification, CoreDNS preferentially uses UDP to communicate with the upstream system.

```
forward . /etc/resolv.conf { prefer_udp }
```

**Step 2** Log in to the CCE console and access the cluster details page. Choose **Add-ons** in the navigation pane, locate **node-local-dns** on the right, and click **Install**.

**Step 3** On the **Install Add-on** page, select the add-on specifications and set related parameters.

**enable\_dnsconfig\_admission**: whether to automatically inject DNSConfig to the newly created pods. Defaults to **false**. The value **true** enables auto injection, preventing the injection of manually configured pod YAML files.

**Step 4** After the preceding configurations are complete, click **Install**.

----End

## Using NodeLocal DNSCache

You can use NodeLocal DNSCache in either of the following ways:

- Auto injection: Automatically configure the **dnsConfig** field of the pod when creating the pod. (Pods cannot be automatically injected into the kube-system namespace.)
- Manual configuration: Manually configure the **dnsConfig** field of the pod.

### Auto injection

The following conditions must be met:



- **enable\_dnsconfig\_admission** has been set to **true** for the add-on.
- The **node-local-dns-injection=enabled** label has been added to the namespace.  
**kubectl label namespace default node-local-dns-injection=enabled**
- The new pod does not run in the kube-system or kube-public namespace.
- The **node-local-dns-injection=disabled** label for disabling DNS injection is not added to the new pod.
- The new pod uses the host network and **DNSPolicy** is **ClusterFirstWithHostNet**. Alternatively, the pod does not use the host network and **DNSPolicy** is **ClusterFirst**.

After auto injection is enabled, the following **dnsConfig** settings are automatically added to the created pod. In addition to the NodeLocal DNSCache address 169.254.20.10, the CoreDNS address 10.247.3.10 is added to **nameservers**, ensuring high availability of the service DNS server.

```
dnsConfig:
  nameservers:
    - 169.254.20.10
    - 10.247.3.10
  searches:
    - default.svc.cluster.local
    - svc.cluster.local
    - cluster.local
  options:
    - name: timeout
      value: ""
    - name: ndots
      value: '5'
    - name: single-request-reopen
```

### Manual configuration

Manually add the **dnsConfig** settings to the pod.

Create a pod and set **dnsConfig** to **169.254.20.10**.

```
apiVersion: v1
kind: Pod
metadata:
  name: nginx
spec:
  containers:
    - image: nginx:alpine
      name: container-0
  dnsConfig:
    nameservers:
      - 169.254.20.10
    searches:
      - default.svc.cluster.local
      - svc.cluster.local
      - cluster.local
    options:
      - name: ndots
        value: '2'
  imagePullSecrets:
    - name: default-secret
```

## 17.8.6 How Does a Container Access an Intranet in a VPC?

This section describes how to access an intranet from a container (outside the cluster in a VPC), including intra-VPC access and cross-VPC access.

## Intra-VPC Access

The performance of accessing an intranet from a container varies depending on the container network models of a cluster.

- **Container tunnel network**

The container tunnel network encapsulates network data packets through tunnels based on the node network. A container can access other resources in the same VPC as long as the node can access the resources. If the access fails, check whether the security group of the peer resource allows access from the node where the container is located.

- **VPC network**

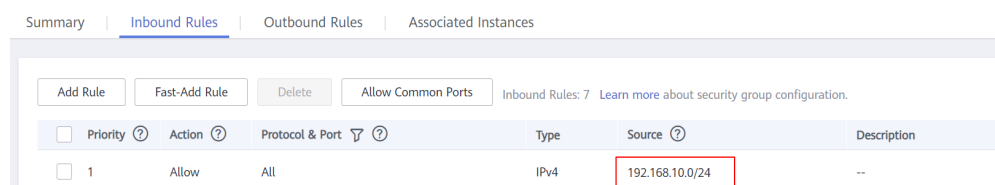
The VPC network model uses VPC routes to forward container traffic. The container CIDR block and the node VPC are not in the same CIDR block. When a container accesses other resources in the same VPC, **the security group of the peer resource must allow access of the container CIDR block.**

For example, the CIDR block where the cluster node resides is 192.168.10.0/24, and the container CIDR block is 172.16.0.0/16.

### Network

Network Model	VPC network
VPC	vpc-asm
Subnet	subnet-asm
Service Forwarding Mode	iptables
Service Network Segment	10.247.0.0/16
Container Network Segment	10.0.0.0/16
Internal API Server Address	https://10.10.0.245:5443
Public API Server Address	Bind EIP

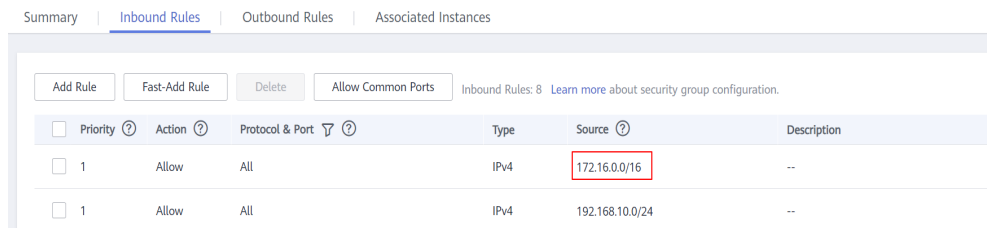
There is an ECS whose IP address is 192.168.10.52 in the VPC (outside the cluster). The security group of the ECS allows access of only the CIDR block of the cluster node.



In this case, if you ping 192.168.10.52 from the container, the ping operation fails.

```
kubectl exec test01-6cbbf97b78-krj6h -it -- /bin/sh
/# ping 192.168.10.25
PING 192.168.10.25 (192.168.10.25): 56 data bytes
^C
--- 192.168.10.25 ping statistics ---
104 packets transmitted, 0 packets received, 100% packet loss
```

Configure the security group to allow access from the container CIDR block 172.16.0.0/16.



In this case, 192.168.10.52 can be pinged from the container.

```
$ kubectl exec test01-6cbbf97b78-krj6h -it -- /bin/sh
/# ping 192.168.10.25
PING 192.168.10.25 (192.168.10.25): 56 data bytes
64 bytes from 192.168.10.25: seq=0 ttl=64 time=1.412 ms
64 bytes from 192.168.10.25: seq=1 ttl=64 time=1.400 ms
64 bytes from 192.168.10.25: seq=2 ttl=64 time=1.299 ms
64 bytes from 192.168.10.25: seq=3 ttl=64 time=1.283 ms
^C
--- 192.168.10.25 ping statistics ---
4 packets transmitted, 4 packets received, 0% packet loss
```

## Cross-VPC Access

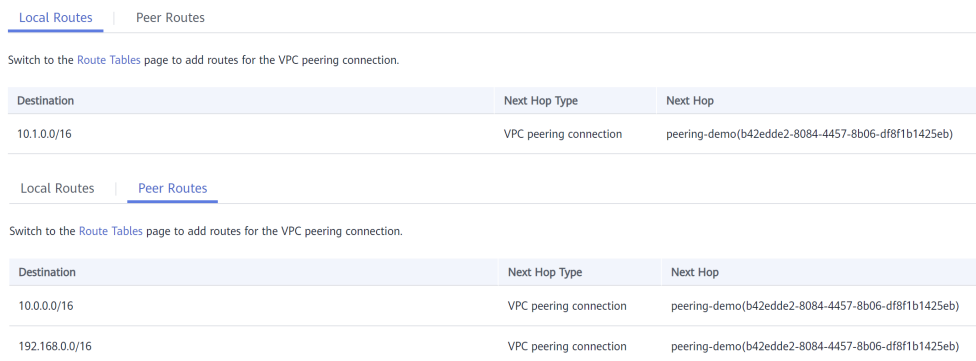
Cross-VPC access is implemented by establishing a peering connection between VPCs.

- In the container tunnel network model, a container can access the peer VPC only when the communication is enabled between the node network and the peer VPC.
- Cloud Native Network 2.0 is similar to the container tunnel network. You only need to enable the communication between the subnet where the container is located and the peer VPC.
- Each VPC network has an independent container CIDR block. In addition to the VPC CIDR block, the container CIDR block also needs to be connected.

Assume that there are two VPCs.

- vpc-demo: Its CIDR block is 192.168.0.0/16, the cluster is in vpc-demo, and the container CIDR block is 10.0.0.0/16.
- vpc-demo2: Its CIDR block is 10.1.0.0/16.

Create a peering connection named **peering-demo** (the local VPC is vpc-demo and the peer VPC is vpc-demo2). Add the container CIDR block to the route of the peer VPC.



After this configuration, you can access the container CIDR block 10.0.0.0/16 in vpc-demo2. During the access, pay attention to the security group configuration and enable the port configuration.

## Accessing Other Cloud Services

Common services that communicate with CCE through an intranet include RDS, DCS, Kafka, RabbitMQ, and ModelArts.

In addition to the network configurations described in [Intra-VPC Access](#) and [Cross-VPC Access](#), you also need to check **whether these cloud services allow external access**. For example, the DCS Redis instance can be accessed only by the IP addresses in its whitelist. Generally, these cloud services can be accessed by IP addresses in the same VPC. However, the container CIDR block in the VPC network model is different from the CIDR block of the VPC. Therefore, you must add the container CIDR block to the whitelist.

## What If a Container Fails to Access an Intranet?

If an intranet cannot be accessed from a container, perform the following operations:

1. View the security group rule of the peer server to check whether the container is allowed to access the peer server.
  - The container tunnel network model needs to allow the IP address of the node where the container is located.
  - The VPC network model needs to allow the container CIDR block.
  - The Cloud Native Network 2.0 model needs to allow the subnet where the container is located.
2. Check whether a whitelist is configured for the peer server. For example, the DCS Redis instance can be accessed only by the IP addresses in its whitelist. Add the container and node CIDR blocks to the whitelist.
3. Check whether the container engine is installed on the peer server and whether it conflicts with the container CIDR block in CCE. If a network conflict occurs, the access fails.

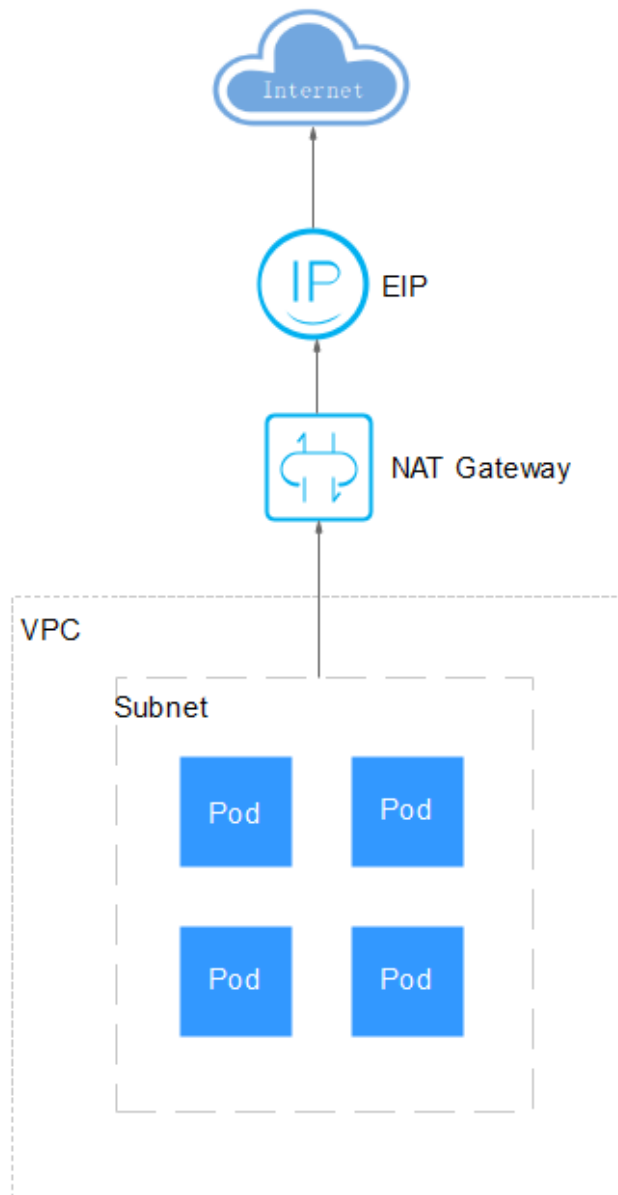
## 17.8.7 Accessing Public Networks from a Container

Containers can access public networks in either of the following ways:

- Bind a public IP address to the node where the container is located.
- Configure SNAT rules through NAT Gateway.



You can use NAT Gateway to enable container pods in a VPC to access public networks. NAT Gateway provides source network address translation (SNAT), which translates private IP addresses to a public IP address by binding an elastic IP address (EIP) to the gateway, providing secure and efficient access to the Internet. [Figure 17-134](#) shows the SNAT architecture. The SNAT function allows the container pods in a VPC to access the Internet without being bound to an EIP. SNAT supports a large number of concurrent connections, which makes it suitable for applications involving a large number of requests and connections.

Figure 17-134 SNAT



To enable a container pod to access the Internet, perform the following steps:

**Step 1** Assign an EIP.



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. Click  at the upper left corner and choose **Networking** > **Elastic IP** in the expanded list.

4. On the **EIPs** page, click **Buy EIP**.
5. Set parameters as required.

 **NOTE**

Set **Region** to the region where container pods are located.



**Step 2** Create a NAT gateway.

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. Click  at the upper left corner and choose **Networking** > **NAT Gateway** in the expanded list.
4. On the displayed page, click **Buy Public NAT Gateway** in the upper right corner.
5. Set parameters as required.

 **NOTE**

Select the same VPC.

**Step 3** Configure an SNAT rule and bind the EIP to the subnet.

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. Click  at the upper left corner and choose **Networking** > **NAT Gateway** in the expanded list.
4. On the page displayed, click the name of the NAT gateway for which you want to add the SNAT rule.
5. On the **SNAT Rules** tab page, click **Add SNAT Rule**.
6. Set parameters as required.

 **NOTE**

SNAT rules take effect by CIDR block. As different container network models use different communication modes, the subnet needs to be selected according to the following rules:

- Tunnel network and VPC network: Select the subnet where the node is located, that is, the subnet selected during node creation.

If there are multiple CIDR blocks, you can create multiple SNAT rules or customize a CIDR block as long as the CIDR block contains the node subnet.

After the SNAT rule is configured, workloads can access public networks from the container. Public networks can be pinged from the container.

----End

## 17.8.8 Configuring Network Policies to Restrict Pod Access

Network policies are designed by Kubernetes to restrict pod access. It is equivalent to a firewall at the application layer to enhance network security. The capabilities

supported by network policies depend on the capabilities of the network add-ons of the cluster.

By default, if a namespace does not have any policy, pods in the namespace accept traffic from any source and send traffic to any destination.

Network policies are classified into the following types:

- **namespaceSelector**: selects particular namespaces for which all pods should be allowed as ingress sources or egress destinations.
- **podSelector**: selects particular pods in the same namespace as the network policy which should be allowed as ingress sources or egress destinations.
- **IPBlock**: selects particular IP blocks to allow as ingress sources or egress destinations.

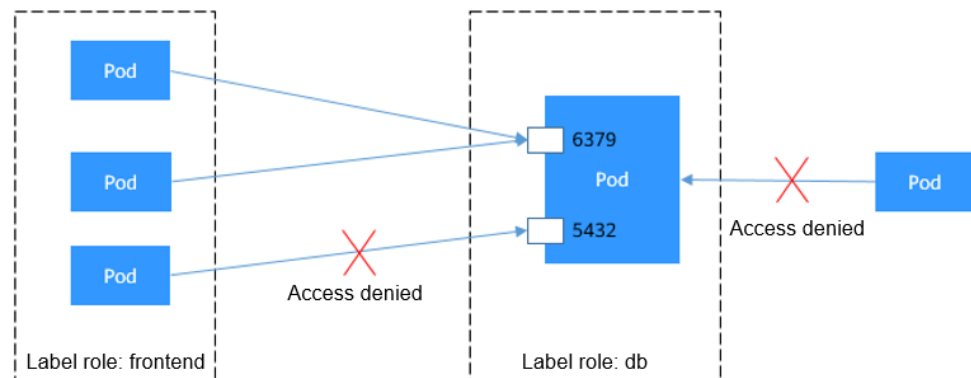
## Notes and Constraints

- Only clusters that use the tunnel network model support network policies. Network policies are classified into the following types:
  - Ingress: All versions support this type.
  - Egress: This rule type cannot be set currently.
- Network isolation is not supported for IPv6 addresses.

## Using Ingress Rules Through YAML

- **Scenario 1: Use a network policy to limit access to a pod to only pods with specific labels.**

Figure 17-135 podSelector



The pod labeled with **role=db** only permits access to its port 6379 from pods labeled with **role=frontend**. To do so, perform the following operations:

- Create the **access-demo1.yaml** file.

```
vim access-demo1.yaml
```

File content:

```
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: access-demo1
  namespace: default
spec:
```

```
podSelector:          # The rule takes effect for pods with the role=db label.
  matchLabels:
    role: db
ingress:              # This is an ingress rule.
- from:
  - podSelector:      # Only allow the access of the pods labeled with role=frontend.
    matchLabels:
      role: frontend
ports:                # Only TCP can be used to access port 6379.
- protocol: TCP
  port: 6379
```

- b. Run the following command to create the network policy based on the **access-demo1.yaml** file:

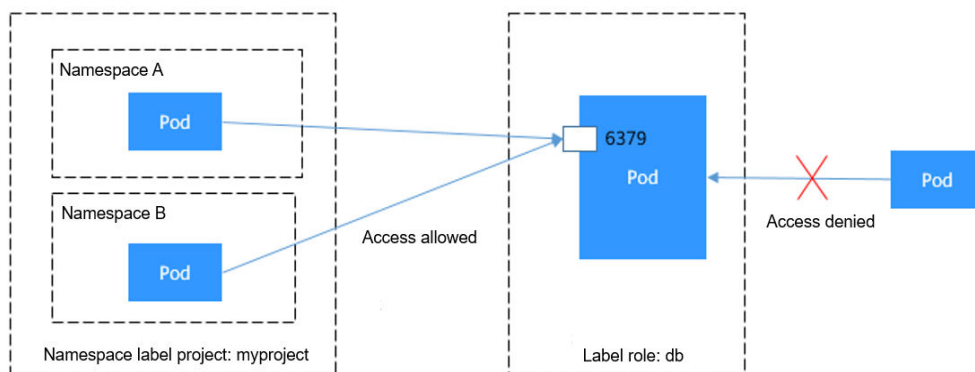
```
kubectl apply -f access-demo1.yaml
```

Expected output:

```
networkpolicy.networking.k8s.io/access-demo1 created
```

- **Scenario 2: Use a network policy to limit access to a pod to only pods in a specific namespace.**

Figure 17-136 namespaceSelector



The pod labeled with **role=db** only permits access to its port 6379 from pods in the namespace labeled with **project=myproject**. To do so, perform the following operations:

- a. Create the **access-demo2.yaml** file.

```
vim access-demo2.yaml
```

File content:

```
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: access-demo2
spec:
  podSelector:          # The rule takes effect for pods with the role=db label.
    matchLabels:
      role: db
  ingress:              # This is an ingress rule.
  - from:
    - namespaceSelector: # Only allow the access of the pods in the namespace labeled
      # with project=myproject.
      matchLabels:
        project: myproject
    ports:                # Only TCP can be used to access port 6379.
    - protocol: TCP
      port: 6379
```

- b. Run the following command to create the network policy based on the **access-demo2.yaml** file:



```
kubectl apply -f access-demo2.yaml
```

Expected output:

```
networkpolicy.networking.k8s.io/access-demo2 created
```

## Creating a Network Policy on the Console

**Step 1** Log in to the CCE console and click the cluster name to access the cluster console.

**Step 2** Choose **Policies** in the navigation pane, click the **Network Policies** tab, and click **Create Network Policy** in the upper right corner.

- **Policy Name:** Specify a network policy name.
- **Namespace:** Select a namespace in which the network policy is applied.
- **Selector:** Enter a label, select the pod to be associated, and click **Add**. You can also click **Reference Workload Label** to use the label of an existing workload.
- **Inbound Rule:** Click **+** to add an inbound rule. For details about parameter settings, see [Table 17-85](#).

**Table 17-85** Adding an inbound rule

Parameter	Description
Protocol & Port	Select the protocol type and port. Currently, TCP and UDP are supported.
Source Namespace	Select a namespace whose objects can be accessed. If this parameter is not specified, the object belongs to the same namespace as the current policy.
Source Pod Label	Allow accessing the pods with this label. If this parameter is not specified, all pods in the namespace can be accessed.

**Step 3** Click **OK**.

----End

## 17.9 Storage (CSI)

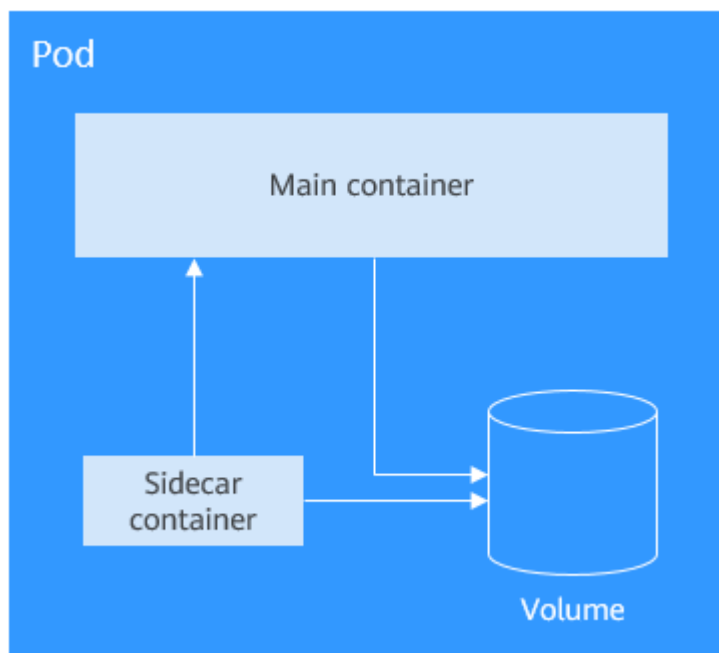
## 17.9.1 Overview

### Volume

On-disk files in a container are ephemeral, which will be lost when the container crashes and are difficult to be shared between containers running together in a pod. The Kubernetes volume abstraction solves both of these problems. Volumes cannot be independently created, but defined in the pod spec.

All containers in a pod can access its volumes, but the volumes must have been mounted. Volumes can be mounted to any directory in a container.

The following figure shows how a storage volume is used between containers in a pod.



A volume will no longer exist if the pod to which it is mounted does not exist. However, files in the volume may outlive the volume, depending on the volume type.

### Volume Types

Volumes can be classified into local volumes and cloud volumes.

- Local volumes

CCE supports the following five types of local volumes. For details about how to use them, see [Using Local Disks as Storage Volumes](#).

- emptyDir: an empty volume used for temporary storage
- hostPath: mounts a directory on a host (node) to your container for reading data from the host.
- ConfigMap: references the data stored in a ConfigMap for use by containers.

- Secret: references the data stored in a secret for use by containers.
- Cloud volumes  
CCE supports the following types of cloud volumes:
  - EVS
  - SFS Turbo
  - OBS

## CSI

You can use Kubernetes Container Storage Interface (CSI) to develop plug-ins to support specific storage volumes.

CCE developed the storage add-on **everest** for you to use cloud storage services, such as EVS and OBS. You can install this add-on when creating a cluster.

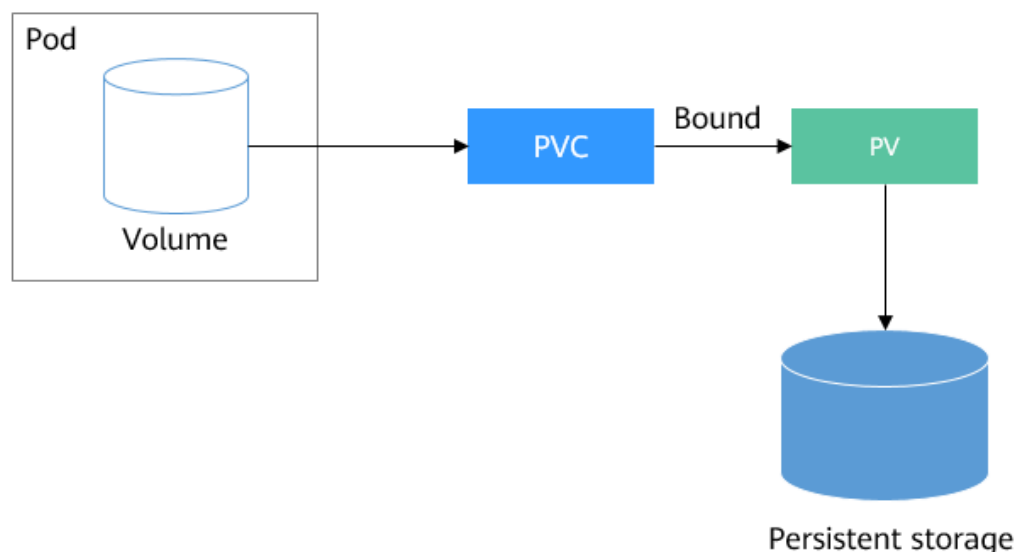
## PV and PVC

Kubernetes provides PersistentVolumes (PVs) and PersistentVolumeClaims (PVCs) to abstract details of how storage is provided from how it is consumed. You can request specific size of storage when needed, just like pods can request specific levels of resources (CPU and memory).

- PV: A PV is a persistent storage volume in a cluster. Same as a node, a PV is a cluster-level resource.
- PVC: A PVC describes a workload's request for storage resources. This request consumes existing PVs in the cluster. If there is no PV available, underlying storage and PVs are dynamically created. When creating a PVC, you need to describe the attributes of the requested persistent storage, such as the size of the volume and the read/write permissions.

You can bind PVCs to PVs in a pod so that the pod can use storage resources. The following figure shows the relationship between PVs and PVCs.

**Figure 17-137** PVC-to-PV binding



PVs describes storage resources in the cluster. PVCs are requests for those resources. The following sections will describe how to use kubectl to connect to storage resources.

If you do not want to create storage resources or PVs manually, you can use [StorageClasses](#).

## StorageClass

StorageClass describes the storage class used in the cluster. You need to specify StorageClass when creating a PVC or PV. As of now, CCE provides storage classes such as csi-disk, csi-nas, and csi-obs by default. When defining a PVC, you can use a StorageClassName to create a PV of the corresponding type and automatically create underlying storage resources.

You can run the following command to query the storage classes that CCE supports. You can use the CSI plug-in provided by CCE to customize a storage class, which functions similarly as the default storage classes in CCE.

```
# kubectl get sc
NAME          PROVISIONER          AGE          # Storage class for EVS disks
csi-disk      everest-csi-provisioner  17d         # Storage class for EVS disks with delayed binding
csi-disk-topology  everest-csi-provisioner  17d         # Storage class for SFS file systems
csi-nas       everest-csi-provisioner  17d         # Storage class for OBS buckets
csi-obs       everest-csi-provisioner  17d
```

After a StorageClass is set, PVs can be automatically created and maintained. You only need to specify the StorageClass when creating a PVC, which greatly reduces the workload.

## Cloud Services for Container Storage

CCE allows you to mount local and cloud storage volumes listed in [Volume Types](#) to your pods. Their features are described below.

**Table 17-86** Detailed description of cloud storage services

Dimension	EVS	OBS	SFS Turbo
Definition	EVS offers scalable block storage for cloud servers. With high reliability, high performance, and rich specifications, EVS disks can be used for distributed file systems, dev/test environments, data warehouses, and high-performance computing (HPC) applications.	OBS is a stable, secure, and easy-to-use object storage service that lets you inexpensively store data of any format and size. You can use it in enterprise backup/archiving, video on demand (VoD), video surveillance, and many other scenarios.	Expandable to 320 TB, SFS Turbo provides a fully hosted shared file storage, highly available and stable to support small files and applications requiring low latency and high IOPS. You can use SFS Turbo in high-traffic websites, log storage, compression/decompression, DevOps, enterprise OA, and containerized applications.
Data storage logic	Stores binary data and cannot directly store files. To store files, you need to format the file system first.	Stores objects. Files directly stored automatically generate the system metadata, which can also be customized by users.	Stores files and sorts and displays data in the hierarchy of files and folders.
Services	Accessible only after being mounted to ECSs or BMSs and initialized.	Accessible through the Internet or Direct Connect (DC). You need to specify the bucket address and use transmission protocols such as HTTP and HTTPS.	Supports the Network File System (NFS) protocol (NFSv3 only). You can seamlessly integrate existing applications and tools with SFS Turbo.
Static provisioning	Supported	Supported	Supported
Dynamic provisioning	Supported	Supported	Not supported
Features	Non-shared storage. Each volume can be mounted to only one node.	Shared, user-mode file system	Shared storage featuring high performance and bandwidth

Dimension	EVS	OBS	SFS Turbo
Usage	HPC, enterprise core cluster applications, enterprise application systems, and dev/test  <b>NOTE</b> HPC apps here require high-speed and high-IOPS storage, such as industrial design and energy exploration.	Big data analysis, static website hosting, online video on demand (VoD), gene sequencing, intelligent video surveillance, backup and archiving, and enterprise cloud boxes (web disks)	High-traffic websites, log storage, DevOps, and enterprise OA
Capacity	TB	EB	TB
Latency	1-2 ms	10ms	1-2 ms
IOPS/TPS	33,000 for a single disk	Tens of millions	100K
Bandwidth	MB/s	TB/s	GB/s

## Notes and Constraints

- A single user can create a maximum of 100 OBS buckets on the console. If you have a large number of CCE workloads and you want to mount an OBS bucket to every workload, you may easily run out of buckets. In this scenario, you are advised to use OBS through the OBS API or SDK and do not mount OBS buckets to the workload on the console.
- For clusters earlier than v1.19.10, if an HPA policy is used to scale out a workload with EVS volumes mounted, the existing pods cannot be read or written when a new pod is scheduled to another node.  
  
For clusters of v1.19.10 and later, if an HPA policy is used to scale out a workload with EVS volume mounted, a new pod cannot be started because EVS disks cannot be attached.
- When you uninstall a subpath in a cluster of v1.19 or earlier, all folders in the subpath are traversed. If there are a large number of folders, the traversal takes a long time, so does the volume unmount. You are advised not to create too many folders in the subpath.
- The maximum size of a single file in OBS mounted to a CCE cluster is far smaller than that defined by obsfs.

## Notice on Using Add-ons

- In version 1.2.0 of the everest add-on, **key authentication** is optimized when OBS is used. After the everest add-on is upgraded from a version earlier than 1.2.0, you need to restart all workloads that use OBS in the cluster. Otherwise, workloads may not be able to use OBS.

## 17.9.2 Using Local Disks as Storage Volumes

You can mount a file directory of the host where a container is located to a specified container path (the `hostPath` mode in Kubernetes) for persistent data storage. Alternatively, you can leave the source path empty (the `emptyDir` mode in Kubernetes), and a temporary directory of the host will be mounted to the mount point of the container for temporary storage.

### Using Local Volumes

CCE supports four types of local volumes.

- **hostPath**: mounts a file directory of the host where the container is located to the specified mount point of the container. For example, if the container needs to access `/etc/hosts`, you can use a `hostPath` volume to map `/etc/hosts`.
- **emptyDir**: stores data temporarily. An `emptyDir` volume is first created when a pod is assigned to a node, and exists as long as that pod is running on that node. When a container pod is terminated, **EmptyDir** will be deleted and the data is permanently lost.
- **ConfigMap**: A `ConfigMap` can be mounted as a volume, and all contents stored in its key are mounted onto the specified container directory. A `ConfigMap` is a type of resource that stores configuration information required by a workload. Its content is user-defined. For details about how to create a `ConfigMap`, see [Creating a ConfigMap](#). For details about how to use a `ConfigMap`, see [Using a ConfigMap](#).
- **Secret**: You can store sensitive information such as passwords, in secrets and mount them as files for use by pods. A secret is a type of resource that holds sensitive data, such as authentication and key information. All content is user-defined. For details about how to create a secret, see [Creating a Secret](#). For details about how to use a secret, see [Using a Secret](#).

The following describes how to mount these four types of volumes.

### hostPath

You can mount a path on the host to a specified container path. A `hostPath` volume is usually used to **store workload logs permanently** or used by workloads that need to **access internal data structure of the Docker engine on the host**.

- Step 1** Log in to the CCE console.
- Step 2** When creating a workload, click **Data Storage** in the **Container Settings**. Click the **Local Volumes** tab and click **+**.
- Step 3** Set parameters for adding a local volume, as listed in [Table 17-87](#).

**Table 17-87** Setting parameters for mounting a `hostPath` volume

Parameter	Description
Storage Type	Select <b>HostPath</b> .


Parameter	Description
Host Path	<p>Path of the host to which the local volume is to be mounted, for example, <b>/etc/hosts</b>.</p> <p><b>NOTE</b>  <b>Host Path</b> cannot be set to the root directory <b>/</b>. Otherwise, the mounting fails. Mount paths can be as follows:</p> <ul style="list-style-type: none"> <li>• <b>/opt/xxxx</b> (excluding <b>/opt/cloud</b>)</li> <li>• <b>/mnt/xxxx</b> (excluding <b>/mnt/paas</b>)</li> <li>• <b>/tmp/xxx</b></li> <li>• <b>/var/xxx</b> (excluding key directories such as <b>/var/lib</b>, <b>/var/script</b>, and <b>/var/paas</b>)</li> <li>• <b>/xxxx</b> (It cannot conflict with the system directory, such as <b>bin</b>, <b>lib</b>, <b>home</b>, <b>root</b>, <b>boot</b>, <b>dev</b>, <b>etc</b>, <b>lost+found</b>, <b>mnt</b>, <b>proc</b>, <b>sbin</b>, <b>srv</b>, <b>tmp</b>, <b>var</b>, <b>media</b>, <b>opt</b>, <b>selinux</b>, <b>sys</b>, and <b>usr</b>.)</li> </ul> <p>Do not set this parameter to <b>/home/paas</b>, <b>/var/paas</b>, <b>/var/lib</b>, <b>/var/script</b>, <b>/mnt/paas</b>, or <b>/opt/cloud</b>. Otherwise, the system or node installation will fail.</p>
Add Container Path	<p>Configure the following parameters:</p> <ol style="list-style-type: none"> <li>1. <b>subPath</b>: Enter a subpath, for example, <b>tmp</b>.  A subpath is used to mount a local disk so that the same data volume is used in a single pod. If this parameter is left blank, the root path is used by default.</li> <li>2. <b>Container Path</b>: Enter the path of the container, for example, <b>/tmp</b>.  This parameter indicates the container path to which a data volume will be mounted. Do not mount the volume to a system directory such as <b>/</b> or <b>/var/run</b>; this action may cause container errors. You are advised to mount the container to an empty directory. If the directory is not empty, ensure that there are no files affecting container startup in the directory. Otherwise, such files will be replaced, resulting in failures to start the container and create the workload.</li> </ol> <p><b>NOTICE</b>  When the container is mounted to a high-risk directory, you are advised to use an account with minimum permissions to start the container; otherwise, high-risk files on the host machine may be damaged.</p> <ol style="list-style-type: none"> <li>3. <b>Permission</b> <ul style="list-style-type: none"> <li>– <b>Read-only</b>: You can only read the data volumes mounted to the path.</li> <li>– <b>Read/Write</b>: You can modify the data volumes mounted to the path. Newly written data is not migrated if the container is migrated, which may cause a data loss.</li> </ul> </li> </ol> <p>Click <b>Add Container Path</b> to add multiple settings. Then, click <b>OK</b>.</p>



----End

## emptyDir

emptyDir applies to temporary data storage, disaster recovery, and runtime data sharing. It will be deleted upon deletion or transfer of workload pods.

- Step 1** Log in to the CCE console.
- Step 2** When creating a workload, click **Data Storage** in the **Container Settings**. Click the **Local Volumes** tab and click .
- Step 3** Set the local volume type to **emptyDir** and set parameters for adding a local volume, as described in [Table 17-88](#).

**Table 17-88** Setting parameters for mounting an emptyDir volume

Parameter	Description
Storage Type	Select <b>emptyDir</b> .
Medium	<ul style="list-style-type: none"> <li>• <b>Default:</b> Data is stored in hard disks, which is applicable to a large amount of data with low requirements on reading and writing efficiency.</li> <li>• <b>Memory:</b> Selecting this option can improve the running speed, but the storage capacity is subject to the memory size. This mode applies to scenarios where the data volume is small and the read and write efficiency is high.</li> </ul> <p><b>NOTE</b></p> <ul style="list-style-type: none"> <li>• If you select <b>Memory</b>, any files you write will count against your container's memory limit. Pay attention to the memory quota. If the memory usage exceeds the threshold, OOM may occur.</li> <li>• If <b>Memory</b> is selected, the size of an emptyDir volume is 50% of the pod specifications and cannot be changed.</li> <li>• If <b>Memory</b> is not selected, emptyDir volumes will not occupy the system memory.</li> </ul>

Parameter	Description
Add Container Path	<p>Configure the following parameters:</p> <ol style="list-style-type: none"> <li>1. <b>subPath</b>: Enter a subpath, for example, <b>tmp</b>. A subpath is used to mount a local disk so that the same data volume is used in a single pod. If this parameter is left blank, the root path is used by default.</li> <li>2. <b>Container Path</b>: Enter the path of the container, for example, <b>/tmp</b>. This parameter indicates the container path to which a data volume will be mounted. Do not mount the volume to a system directory such as <b>/</b> or <b>/var/run</b>; this action may cause container errors. You are advised to mount the container to an empty directory. If the directory is not empty, ensure that there are no files affecting container startup in the directory. Otherwise, such files will be replaced, resulting in failures to start the container and create the workload. <b>NOTICE</b> When the container is mounted to a high-risk directory, you are advised to use an account with minimum permissions to start the container; otherwise, high-risk files on the host machine may be damaged.</li> <li>3. Permission <ul style="list-style-type: none"> <li>- <b>Read-only</b>: You can only read the data volumes mounted to the path.</li> <li>- <b>Read/Write</b>: You can modify the data volumes mounted to the path. Newly written data is not migrated if the container is migrated, which may cause a data loss.</li> </ul> </li> </ol> <p>Click <b>Add Container Path</b> to add multiple settings. Then, click <b>OK</b>.</p>

----End

## ConfigMap

The data stored in a ConfigMap can be referenced in a volume of type ConfigMap. You can mount such a volume to a specified container path. The platform supports the separation of workload codes and configuration files. ConfigMap volumes are used to store workload configuration parameters. Before that, you need to create ConfigMaps in advance. For details, see [Creating a ConfigMap](#).

**Step 1** Log in to the CCE console.

**Step 2** When creating a workload, click **Data Storage** in the **Container Settings**. Click the **Local Volumes** tab and click **+**.

**Step 3** Set the local volume type to **ConfigMap** and set parameters for adding a local volume, as shown in [Table 17-89](#).

**Table 17-89** Setting parameters for mounting a ConfigMap volume

Parameter	Description
Storage Type	Select <b>ConfigMap</b> .
Option	Select the desired ConfigMap name. A ConfigMap must be created in advance. For details, see <a href="#">Creating a ConfigMap</a> .
Add Container Path	<p>Configure the following parameters:</p> <ol style="list-style-type: none"> <li><b>subPath</b>: Enter a subpath, for example, <b>tmp</b>. <ul style="list-style-type: none"> <li>A subpath is used to mount a local volume so that the same data volume is used in a single pod.</li> <li>The subpath can be the key and value of a ConfigMap or secret. If the subpath is a key-value pair that does not exist, the data import does not take effect.</li> <li>The data imported by specifying a subpath will not be updated along with the ConfigMap/secret updates.</li> </ul> </li> <li><b>Container Path</b>: Enter the path of the container, for example, <b>/tmp</b>. This parameter indicates the container path to which a data volume will be mounted. Do not mount the volume to a system directory such as <b>/</b> or <b>/var/run</b>; this action may cause container errors. You are advised to mount the container to an empty directory. If the directory is not empty, ensure that there are no files affecting container startup in the directory. Otherwise, such files will be replaced, resulting in failures to start the container and create the workload. <b>NOTICE</b> When the container is mounted to a high-risk directory, you are advised to use an account with minimum permissions to start the container; otherwise, high-risk files on the host machine may be damaged.</li> <li>Set the permission to <b>Read-only</b>. Data volumes in the path are read-only.</li> </ol> <p>Click <b>Add Container Path</b> to add multiple settings. Then, click <b>OK</b>.</p>

----End

## Secret

You can mount a secret as a volume to the specified container path. Contents in a secret are user-defined. Before that, you need to create a secret. For details, see [Creating a Secret](#).

**Step 1** Log in to the CCE console.

**Step 2** When creating a workload, click **Data Storage** in the **Container Settings**. Click the **Local Volumes** tab and click **+**.

**Step 3** Set the local volume type to **Secret** and set parameters for adding a local volume, as shown in [Table 17-90](#).

**Table 17-90** Setting parameters for mounting a secret volume

Parameter	Description
Storage Type	Select <b>Secret</b> .
Secret	Select the desired secret name. A secret must be created in advance. For details, see <a href="#">Creating a Secret</a> .
Add Container Path	<p>Configure the following parameters:</p> <ol style="list-style-type: none"> <li><b>subPath</b>: Enter a subpath, for example, <b>tmp</b>. <ul style="list-style-type: none"> <li>A subpath is used to mount a local volume so that the same data volume is used in a single pod.</li> <li>The subpath can be the key and value of a ConfigMap or secret. If the subpath is a key-value pair that does not exist, the data import does not take effect.</li> <li>The data imported by specifying a subpath will not be updated along with the ConfigMap/secret updates.</li> </ul> </li> <li><b>Container Path</b>: Enter the path of the container, for example, <b>/tmp</b>. This parameter indicates the container path to which a data volume will be mounted. Do not mount the volume to a system directory such as <b>/</b> or <b>/var/run</b>; this action may cause container errors. You are advised to mount the container to an empty directory. If the directory is not empty, ensure that there are no files affecting container startup in the directory. Otherwise, such files will be replaced, resulting in failures to start the container and create the workload. <b>NOTICE</b> When the container is mounted to a high-risk directory, you are advised to use an account with minimum permissions to start the container; otherwise, high-risk files on the host machine may be damaged.</li> <li>Set the permission to <b>Read-only</b>. Data volumes in the path are read-only.</li> </ol> <p>Click <b>Add Container Path</b> to add multiple settings. Then, click <b>OK</b>.</p>

----End

## Mounting a hostPath Volume Using kubectl

You can use kubectl to mount a file directory of the host where the container is located to a specified mount path of the container.

**Step 1** Use kubectl to connect to the cluster. For details, see [Connecting to a Cluster Using kubectl](#).

**Step 2** Run the following commands to configure the `hostPath-pod-example.yaml` file, which is used to create a pod.

**touch hostPath-pod-example.yaml**

**vi hostPath-pod-example.yaml**

Mount the hostPath volume for the Deployment. The following is an example:

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: hostpath-pod-example
  namespace: default
spec:
  replicas: 1
  selector:
    matchLabels:
      app: hostpath-pod-example
  template:
    metadata:
      labels:
        app: hostpath-pod-example
    spec:
      containers:
        - image: nginx
          name: container-0
          volumeMounts:
            - mountPath: /tmp
              name: hostpath-example
      imagePullSecrets:
        - name: default-secret
      restartPolicy: Always
      volumes:
        - name: hostpath-example
          hostPath:
            path: /tmp/test
```

**Table 17-91** Local disk storage dependency parameters

Parameter	Description
mountPath	Mount path of the container. In this example, the volume is mounted to the <code>/tmp</code> directory.
hostPath	Host path. In this example, the host path is <code>/tmp/test</code> .

 **NOTE**

`spec.template.spec.containers.volumeMounts.name` and `spec.template.spec.volumes.name` must be consistent because they have a mapping relationship.

**Step 3** Run the following command to create the pod:

**kubectl create -f hostPath-pod-example.yaml**

**Step 4** Verify the mounting.

1. Query the pod name of the workload (`hostpath-pod-example` is used as an example).

```
kubectl get pod | grep hostpath-pod-example
```

Expected outputs:

```
hostpath-pod-example-55c8d4dc59-md5d9 1/1 Running 0 35s
```

2. Create the **test1** file in the container mount path **/tmp**.  
kubect exec hostpath-pod-example-55c8d4dc59-md5d9 -- touch /tmp/test1
3. Verify that the file is created in the host path **/tmp/test/**.  
ll /tmp/test/

Expected outputs:

```
-rw-r--r-- 1 root root 0 Jun 1 16:12 test1
```

4. Create the **test2** file in the host path **/tmp/test/**.  
touch /tmp/test/test2
5. Verify that the file is created in the container mount path.  
kubect exec hostpath-pod-example-55c8d4dc59-md5d9 -- ls -l /tmp

Expected outputs:

```
-rw-r--r-- 1 root root 0 Jun 1 08:12 test1  
-rw-r--r-- 1 root root 0 Jun 1 08:14 test2
```

----End

### 17.9.3 PersistentVolumes (PVs)

A PV is a persistent storage volume in a cluster. Same as a node, a PV is a cluster-level resource.

#### Notes and Constraints

- On the new CCE console (the cluster needs to be **upgraded to v1.19.10 or later** and **the everest add-on needs to be upgraded to v1.2.10 or later**), PVs are open to you for management. On the old CCE console, PVs can only be imported or dynamically created. You cannot manage the PV lifecycle on the console.
- Multiple PVs can use the same SFS file system with the following restrictions:
  - An error may occur if multiple PVCs/PVs that use the same underlying SFS file system are mounted to the same pod.
  - The **persistentVolumeReclaimPolicy** parameter in the PVs must be set to **Retain**. Otherwise, when a PV is deleted, the associated underlying volume may be deleted. In this case, other PVs associated with the underlying volume may be abnormal.
  - When the underlying volume is repeatedly used, it is recommended that ReadWriteMany be implemented at the application layer to prevent data overwriting and loss.

#### Volume Access Modes

PVs can be mounted to the host system only in the mode supported by underlying storage resources. For example, a file storage system can be read and written by multiple nodes, but an EVS disk can be read and written by only one node.

- ReadWriteOnce: A volume can be mounted as read-write by a single node. This access mode is supported by EVS.
- ReadWriteMany: A volume can be mounted as read-write by multiple nodes. This access mode is supported by SFS and OBS.

**Table 17-92** Access modes supported by cloud storage

Storage Type	ReadWriteOnce	ReadWriteMany
EVS	√	×
SFS	×	√
OBS	×	√

## PV Reclaim Policy

A PV reclaim policy is used to delete or reclaim underlying volumes when a PVC is deleted. The value can be **Delete** or **Retain**.

- **Delete:** When a PVC is deleted, the PV and underlying storage resources 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 a PVC is deleted, the PV resource is in the Released state and cannot be bound to the PVC again.

Everest also allows you to delete a PVC without deleting underlying storage resources. This function can be achieved only by using a YAML file. Set the PV reclaim policy to **Delete** and add **annotations"everest.io/reclaim-policy: retain-volume-only"**. In this way, when the PVC is deleted, the PV resource is deleted, but the underlying storage resources are retained.

## Creating an EVS Volume

### NOTE

The requirements for creating an EVS volume are as follows:

- System disks, DSS disks, and shared disks cannot be used.
- The EVS disk is one of the supported types (common I/O, high I/O, and ultra-high I/O), and the EVS disk device type is SCSI.
- The EVS disk is not frozen or used, and the status is available.

### Using the CCE Console

**Step 1** Log in to the CCE console.

**Step 2** Access the cluster details page, choose **Storage** from the navigation pane, and click the **Volumes** tab.

**Step 3** Click **Create Volume** in the upper right corner. In the dialog box displayed, set the volume parameters.

- **Volume Type:** Select **EVS**.
- **EVS:**
- **PV Name:** Enter a PV name.
- **Access Mode:** ReadWriteOnce
- **Reclaim Policy:** Select **Delete** or **Retain** as required. For details, see [PV Reclaim Policy](#).

**Step 4 Click Create.**

----End

**Using YAML**

```
apiVersion: v1
kind: PersistentVolume
metadata:
  annotations:
    pv.kubernetes.io/provisioned-by: everest-csi-provisioner
    everest.io/reclaim-policy: retain-volume-only # (Optional) The PV is deleted while the underlying
volume is retained.
  name: cce-evs-test
  labels:
    failure-domain.beta.kubernetes.io/region: eu-west-101
    failure-domain.beta.kubernetes.io/zone: eu-west-101a
spec:
  accessModes:
    - ReadWriteOnce # Access mode. The value is fixed to ReadWriteOnce for EVS.
  capacity:
    storage: 10Gi # EVS disk capacity, in the unit of Gi. The value ranges from 1 to 32768.
  csi:
    driver: disk.csi.everest.io # Dependent storage driver for the mounting.
    fsType: ext4
    volumeHandle: 459581af-e78c-4356-9e78-eaf9cd8525eb # Volume ID of the EVS disk.
  volumeAttributes:
    everest.io/disk-mode: SCSI # Device type of the EVS disk. Only SCSI is supported.
    everest.io/disk-volume-type: SAS # EVS disk type.
    storage.kubernetes.io/csiProvisionerIdentity: everest-csi-provisioner

persistentVolumeReclaimPolicy: Delete # Reclaim policy.
storageClassName: csi-disk # Storage class name. The value must be csi-disk.
```

**Table 17-93** Key parameters

Parameter	Description
everest.io/reclaim-policy: retain-volume-only	This field is optional. Currently, only <b>retain-volume-only</b> is supported. This field is valid only when the everest version is 1.2.9 or later and the reclaim policy is Delete. If the reclaim policy is Delete and the current value is <b>retain-volume-only</b> , the associated PV is deleted while the underlying storage volume is retained, when a PVC is deleted.
failure-domain.beta.kubernetes.io/region	Region where the cluster is located.
failure-domain.beta.kubernetes.io/zone	AZ where the EVS volume is created. It must be the same as the AZ planned for the workload.



Parameter	Description
volumeHandle	<p>Volume ID of the EVS disk.</p> <p>To obtain the volume ID, log in to the <b>Cloud Server Console</b>. In the navigation pane, choose <b>Elastic Volume Service &gt; Disks</b>. Click the name of the target EVS disk to go to its details page. On the <b>Summary</b> tab page, click the copy button after <b>ID</b>.</p>
everest.io/disk-volume-type	<p>EVS disk type. All letters are in uppercase.</p> <ul style="list-style-type: none"> <li>• <b>SAS</b>: high I/O</li> <li>• <b>SSD</b>: ultra-high I/O</li> </ul>
everest.io/enterprise-project-id	<p>This field is optional.</p> <p>Enterprise project ID of the EVS disk. If an enterprise project is specified, you need to specify the same enterprise project when creating a PVC. Otherwise, the PVC cannot be bound to a PV.</p> <p>On the ECS console, choose <b>Elastic Volume Service &gt; Disks</b> in the navigation pane. Click the name of the EVS disk to access its details page. On the <b>Summary</b> tab page, click the enterprise project in <b>Management Information</b> to access the enterprise project console. Copy the corresponding ID to obtain the ID of the enterprise project to which the EVS disk belongs.</p>

Parameter	Description
persistentVolumeReclaimPolicy	<p>A reclaim policy is supported when the cluster version is equal to or later than 1.19.10 and the everest version is equal to or later than 1.2.9.</p> <p>The Delete and Retain policies are supported.</p> <p><b>Delete:</b></p> <ul style="list-style-type: none"> <li>If <b>everest.io/reclaim-policy</b> is not specified, both the PV and EVS disk are deleted when a PVC is deleted.</li> <li>If <b>everest.io/reclaim-policy</b> is set to <b>retain-volume-only set</b>, when a PVC is deleted, the PV is deleted but the EVS resources are retained.</li> </ul> <p><b>Retain:</b> 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.</p> <p>If high data security is required, you are advised to select <b>Retain</b> to prevent data from being deleted by mistake.</p>

## Creating an SFS Volume

### NOTE

- The SFS file system and the cluster must be in the same VPC.

### Using YAML

```

apiVersion: v1
kind: PersistentVolume
metadata:
  annotations:
    pv.kubernetes.io/provisioned-by: everest-csi-provisioner
    everest.io/reclaim-policy: retain-volume-only # (Optional) The PV is deleted while the underlying
volume is retained.
  name: cce-sfs-test
spec:
  accessModes:
    - ReadWriteMany # Access mode. The value must be ReadWriteMany for SFS.
  capacity:
    storage: 1Gi # File storage capacity.
  csi:
    driver: disk.csi.everest.io # Mount the dependent storage driver.
    fsType: nfs
    volumeHandle: 30b3d92a-0bc7-4610-b484-534660db81be # SFS file system ID.
    volumeAttributes:
      everest.io/share-export-location:
      storage.kubernetes.io/csiProvisionerIdentity: everest-csi-provisioner
    persistentVolumeReclaimPolicy: Retain # Reclaim policy.

```

```
storageClassName: csi-nas      # Storage class name. The value must be csi-nas for SFS.
mountOptions: []             # Mount options
```

**Table 17-94** Key parameters

Parameter	Description
everest.io/reclaim-policy: retain-volume-only	This field is optional. Currently, only <b>retain-volume-only</b> is supported. This field is valid only when the everest version is 1.2.9 or later and the reclaim policy is Delete. If the reclaim policy is Delete and the current value is <b>retain-volume-only</b> , the associated PV is deleted while the underlying storage volume is retained, when a PVC is deleted.
volumeHandle	File system ID. On the management console, choose <b>Service List &gt; Storage &gt; Scalable File Service</b> . In the SFS file system list, click the name of the target file system and copy the content following <b>ID</b> on the page displayed.
everest.io/share-export-location	Shared path of the file system. On the management console, choose <b>Service List &gt; Storage &gt; Scalable File Service</b> . You can obtain the shared path of the file system from the <b>Mount Address</b> column.
mountOptions	Mount options. If not specified, the following configurations are used by default. For details, see <a href="#">SFS Volume Mount Options</a> . mountOptions: - vers=3 - timeo=600 - nolock - hard

Parameter	Description
persistentVolumeReclaimPolicy	<p>A reclaim policy is supported when the cluster version is equal to or later than 1.19.10 and the everest version is equal to or later than 1.2.9.</p> <p>The options are as follows:</p> <p><b>Delete:</b></p> <ul style="list-style-type: none"> <li>• If <b>everest.io/reclaim-policy</b> is not specified, both the PV and SFS volume are deleted when a PVC is deleted.</li> <li>• If <b>everest.io/reclaim-policy</b> is set to <b>retain-volume-only set</b>, when a PVC is deleted, the PV is deleted but the file storage resources are retained.</li> </ul> <p><b>Retain:</b> 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.</p> <p>If high data security is required, you are advised to select <b>Retain</b> to prevent data from being deleted by mistake.</p>

## Creating an OBS Volume

### NOTE

A single user can create a maximum of 100 OBS buckets on the console. If you have a large number of CCE workloads and you want to mount an OBS bucket to every workload, you may easily run out of buckets. In this scenario, you are advised to use OBS through the OBS API or SDK and do not mount OBS buckets to the workload on the console.

### Using the CCE Console

**Step 1** Log in to the CCE console.

**Step 2** Access the cluster details page, choose **Storage** from the navigation pane, and click the **Volumes** tab.

**Step 3** Click **Create Volume** in the upper right corner. In the dialog box displayed, set the volume parameters.

- **Volume Type:** Select **OBS**.
- Select OBS resources.
- **PV Name:** Enter a PV name.
- **Access Mode:** ReadWriteMany
- **Reclaim Policy:** Select **Delete** or **Retain** as required. For details, see [PV Reclaim Policy](#).
- **Key:** You can customize the access key (AK/SK) for mounting an OBS volume. You can use the AK/SK to create a secret and mount the secret to the PV. For details, see [Using a Custom AK/SK to Mount an OBS Volume](#).

- **Mount Options:** mount options. For details about the options, see [Setting Mount Options](#).

**Step 4** Click **Create**.

----End

**Using YAML**

```
apiVersion: v1
kind: PersistentVolume
metadata:
  annotations:
    pv.kubernetes.io/provisioned-by: everest-csi-provisioner
    everest.io/reclaim-policy: retain-volume-only # (Optional) The PV is deleted while the underlying
volume is retained.
  name: cce-obs-test
spec:
  accessModes:
  - ReadWriteMany # Access mode. The value must be ReadWriteMany for OBS.
  capacity:
    storage: 1Gi # Storage capacity. This parameter is set only to meet the PV format requirements. It can
be set to any value. The actual OBS space size is not limited by this value.
  csi:
    driver: obs.csi.everest.io # Dependent storage driver for the mounting.
    fsType: obsfs # OBS file type.
    volumeHandle: cce-obs-bucket # OBS bucket name.
  volumeAttributes:
    everest.io/obs-volume-type: STANDARD
    everest.io/region:

    storage.kubernetes.io/csiProvisionerIdentity: everest-csi-provisioner
  nodePublishSecretRef:
    name: test-user
    namespace: default
  persistentVolumeReclaimPolicy: Retain # Reclaim policy.
  storageClassName: csi-obs # Storage class name. The value must be csi-obs for OBS.
  mountOptions: [] # Mount options.
```

**Table 17-95** Key parameters

Parameter	Description
everest.io/reclaim-policy: retain-volume-only	This field is optional. Currently, only <b>retain-volume-only</b> is supported. This field is valid only when the everest version is 1.2.9 or later and the reclaim policy is Delete. If the reclaim policy is Delete and the current value is <b>retain-volume-only</b> , the associated PV is deleted while the underlying storage volume is retained, when a PVC is deleted.
fsType	File type. The value can be <b>obsfs</b> or <b>s3fs</b> . If the value is <b>s3fs</b> , an OBS bucket is created and mounted using s3fs. If the value is <b>obsfs</b> , an OBS parallel file system is created and mounted using obsfs. You are advised to set this field to <b>obsfs</b> .
volumeHandle	OBS bucket name.
everest.io/obs-volume-type	Storage class, including <b>STANDARD</b> (standard bucket) and <b>WARM</b> (infrequent access bucket).

Parameter	Description
everest.io/region	Region where the OBS bucket is deployed.
nodePublishSecretRef	Access key (AK/SK) used for mounting the object storage volume. You can use the AK/SK to create a secret and mount it to the PV. For details, see <a href="#">Using a Custom AK/SK to Mount an OBS Volume</a> .
mountOptions	Mount options. For details, see <a href="#">OBS Volume Mount Options</a> .
persistentVolumeReclaimPolicy	<p>A reclaim policy is supported when the cluster version is equal to or later than 1.19.10 and the everest version is equal to or later than 1.2.9.</p> <p>The Delete and Retain policies are supported.</p> <p><b>Delete:</b></p> <ul style="list-style-type: none"> <li>• If <b>everest.io/reclaim-policy</b> is not specified, both the PV and OBS volume are deleted when a PVC is deleted.</li> <li>• If <b>everest.io/reclaim-policy</b> is set to <b>retain-volume-only set</b>, when a PVC is deleted, the PV is deleted but the object storage resources are retained.</li> </ul> <p><b>Retain:</b> 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.</p> <p>If high data security is required, you are advised to select <b>Retain</b> to prevent data from being deleted by mistake.</p>

## Creating an SFS Turbo Volume

### NOTE

SFS Turbo and the cluster must be in the same VPC.

### Using the CCE Console

**Step 1** Log in to the CCE console.

**Step 2** Access the cluster details page, choose **Storage** from the navigation pane, and click the **Volumes** tab.

**Step 3** Click **Create Volume** in the upper right corner. In the dialog box displayed, set the volume parameters.

- **Volume Type:** Select **SFS Turbo**.
- **SFS Turbo:** Select SFS Turbo resources.
- **PV Name:** Enter a PV name.
- **Access Mode:** ReadWriteMany
- **Reclaim Policy:** Select **Retain**. For details, see [PV Reclaim Policy](#).

- **Mount Options:** mount options. For details about the options, see [Setting Mount Options](#).

**Step 4** Click **Create**.

----End

**Using YAML**

```
apiVersion: v1
kind: PersistentVolume
metadata:
  annotations:
    pv.kubernetes.io/provisioned-by: everest-csi-provisioner
  name: cce-sfsturbo-test
spec:
  accessModes:
    - ReadWriteMany # Access mode. The value must be ReadWriteMany for SFS Turbo.
  capacity:
    storage: 100.00Gi # SFS Turbo volume capacity.
  csi:
    driver: sfsturbo.csi.everest.io # Dependent storage driver for the mounting.
    fsType: nfs
    volumeHandle: 6674bd0a-d760-49de-bb9e-805c7883f047 # SFS Turbo volume ID.
    volumeAttributes:
      everest.io/share-export-location: 192.168.0.85:/ # Shared path of the SFS Turbo volume.
      storage.kubernetes.io/csiProvisionerIdentity: everest-csi-provisioner
    persistentVolumeReclaimPolicy: Retain # Reclaim policy.
    storageClassName: csi-sfsturbo # Storage class name. The value must be csi-sfsturbo for SFS Turbo.
  mountOptions: [] # Mount options.
```

**Table 17-96** Key parameters

Parameter	Description
volumeHandle	SFS Turbo volume ID. You can obtain the ID on the SFS Turbo storage instance details page on the SFS console.
everest.io/share-export-location	Shared path of the SFS Turbo volume.
mountOptions	Mount options. If not specified, the following configurations are used by default. For details, see <a href="#">SFS Volume Mount Options</a> . mountOptions: - vers=3 - timeo=600 - nolock - hard

Parameter	Description
persistentVolumeReclaim-Policy	<p>A reclaim policy is supported when the cluster version is equal to or later than 1.19.10 and the everest version is equal to or later than 1.2.9.</p> <p>The Delete and Retain policies are supported.</p> <p><b>Delete:</b></p> <ul style="list-style-type: none"> <li>If <b>everest.io/reclaim-policy</b> is not specified, both the PV and SFS Turbo volume are deleted when a PVC is deleted.</li> <li>If <b>everest.io/reclaim-policy</b> is set to <b>retain-volume-only set</b>, when a PVC is deleted, the PV is deleted but the SFF Turbo resources are retained.</li> </ul> <p><b>Retain:</b> 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.</p> <p>If high data security is required, you are advised to select <b>Retain</b> to prevent data from being deleted by mistake.</p>

## 17.9.4 PersistentVolumeClaims (PVCs)

A PVC describes a workload's request for storage resources. This request consumes existing PVs in the cluster. If there is no PV available, underlying storage and PVs are dynamically created. When creating a PVC, you need to describe the attributes of the requested persistent storage, such as the size of the volume and the read/write permissions.

### Notes and Constraints

When a PVC is created, the system checks whether there is an available PV with the same configuration in the cluster. If yes, the PVC binds the available PV to the cluster. If no PV meets the matching conditions, the system dynamically creates a storage volume.

Description	PVC Field	PV Field	Matching Logic
region	pvc.metadata.labels (failure-domain.beta.kubernetes.io/region or topology.kubernetes.io/region)	pv.metadata.labels (failure-domain.beta.kubernetes.io/region or topology.kubernetes.io/region)	Defined or not defined at the same time. If defined, the settings must be consistent.



Descrip tion	PVC Field	PV Field	Matching Logic
zone	pvc.metadata.labels (failure-domain.beta.kubernetes.io/zone or topology.kubernetes.io/zone)	pv.metadata.labels (failure-domain.beta.kubernetes.io/zone or topology.kubernetes.io/zone)	Defined or not defined at the same time. If defined, the settings must be consistent.
EVS disk type	pvc.metadata.annotations (everest.io/disk-volume-type)	pv.spec.csi.volumeAttributes (everest.io/disk-volume-type)	Defined or not defined at the same time. If defined, the settings must be consistent.
Enterprise project ID	pvc.metadata.annotations (everest.io/enterprise-project-id)	pv.spec.csi.volumeAttributes (everest.io/enterprise-project-id)	Defined or not defined at the same time. If defined, the settings must be consistent.
access Mode	accessMode	accessMode	The settings must be consistent.
Storage class	storageclass	storageclass	The settings must be consistent.

## Volume Access Modes

PVs can be mounted to the host system only in the mode supported by underlying storage resources. For example, a file storage system can be read and written by multiple nodes, but an EVS disk can be read and written by only one node.

- **ReadWriteOnce:** A volume can be mounted as read-write by a single node. This access mode is supported by EVS.
- **ReadWriteMany:** A volume can be mounted as read-write by multiple nodes. This access mode is supported by SFS and OBS.

**Table 17-97** Supported access modes

Storage Type	ReadWriteOnce	ReadWriteMany
EVS	√	×
SFS	×	√
OBS	×	√

## Enterprise Project Support

### NOTE

To use this function, the everest add-on must be upgraded to 1.2.33 or later.

- **Using a Storage Class to Create a PVC:**

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 no enterprise project is specified, the enterprise project specified in StorageClass will be used by default for creating storage resources.

  - For a custom StorageClass, you can specify an enterprise project in StorageClass. For details, see [Specifying an Enterprise Project for Storage Classes](#). If no enterprise project is specified in StorageClass, the default enterprise project is used.
  - For the csi-disk and csi-obs storage classes provided by CCE, the created storage resources belong to the default enterprise project.
- **Using a PV to Create a PVC:**

When you create a PVC using a PV, ensure that **everest.io/enterprise-project-id** specified in the PVC and PV are the same because an enterprise project has been specified during storage resource creation. Otherwise, the PVC and PV cannot be bound.

## Using a Storage Class to Create a PVC

StorageClass describes the storage class used in the cluster. You need to specify StorageClass to dynamically create PVs and underlying storage resources when creating a PVC.

### Using the CCE Console

- Step 1** Log in to the CCE console.
- Step 2** Go to the cluster details page, choose **Storage** from the navigation pane, and click the **PersistentVolumeClaims (PVCs)** tab.
- Step 3** Click **Create PVC** in the upper right corner. In the dialog box displayed, set the PVC parameters.
  - **Creation Method:** Select **Storage class**.
  - **PVC Name:** Enter a PVC name.
  - **Storage Class:** Select the required storage class. The following storage resources can be dynamically provisioned:
    - **csi-disk:** EVS disk.
    - **csi-obs:** OBS bucket.
  - **AZ** (supported only by EVS): Select the AZ where the EVS disk is located.
  - **Disk Type:** Select an EVS disk type. EVS disk types vary in different regions.
    - High I/O
    - Ultra-high I/O

- **Access Mode:** `ReadWriteOnce` and `ReadWriteMany` are supported. For details, see [Volume Access Modes](#).
- **Capacity (GiB)** (supported only by EVS and SFS): storage capacity. This parameter is not available for OBS.
- **Secret** (supported only for OBS): Select an access key for OBS. For details, see [Using a Custom AK/SK to Mount an OBS Volume](#).
- **Enterprise Project** (supported only for EVS and OBS): Select an enterprise project to which the cluster belongs or use the default enterprise project.

#### Step 4 Click Create.

----End

#### Using YAML

Example YAML for EVS

- **failure-domain.beta.kubernetes.io/region:** region where the cluster is located.
- **failure-domain.beta.kubernetes.io/zone:** AZ where the EVS volume is created. It must be the same as the AZ planned for the workload.

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: pvc-evs-auto-example
  namespace: default
  annotations:
    everest.io/disk-volume-type: SSD # EVS disk type.

    everest.io/enterprise-project-id: 86bfc701-9d9e-4871-a318-6385aa368183 # (Optional) Enterprise
project ID. Only the enterprise project to which the cluster belongs and the default enterprise project are
supported. The value 0 indicates the default enterprise project. Everest must be upgraded to 1.2.33 or later.
  labels:
    failure-domain.beta.kubernetes.io/region: eu-west-101
    failure-domain.beta.kubernetes.io/zone: eu-west-101a
spec:
  accessModes:
    - ReadWriteOnce # The value must be ReadWriteOnce for EVS.
  resources:
    requests:
      storage: 10Gi # EVS disk capacity, ranging from 1 to 32768.
      storageClassName: csi-disk # The storage class type is EVS.
```

#### Example YAML for OBS:

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: obs-warm-provision-pvc
  namespace: default
  annotations:
    everest.io/obs-volume-type: STANDARD # OBS bucket type. Currently, standard (STANDARD) and
infrequent access (WARM) are supported.
    csi.storage.k8s.io/fstype: obsfs # File type. obsfs indicates to create a parallel file system
(recommended), and s3fs indicates to create an OBS bucket.
    everest.io/enterprise-project-id: 86bfc701-9d9e-4871-a318-6385aa368183 # (Optional) Enterprise
project ID. Only the enterprise project to which the cluster belongs and the default enterprise project are
supported. The value 0 indicates the default enterprise project. Everest must be upgraded to 1.2.33 or later.
  spec:
  accessModes:
    - ReadWriteMany # The value must be ReadWriteMany for OBS.
  resources:
    requests:
      storage: 1Gi # This field 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 # The storage class type is OBS.

## Using a PV to Create a PVC

If a PV has been created, you can create a PVC to apply for PV resources.

### Using the CCE Console

**Step 1** Log in to the CCE console.

**Step 2** Go to the cluster details page, choose **Storage** from the navigation pane, and click the **PersistentVolumeClaims (PVCs)** tab.

**Step 3** Click **Create PVC** in the upper right corner. In the dialog box displayed, set the PVC parameters.

- **Creation Method:** Select **Existing volume**.
- **PVC Name:** Enter a PVC name.
- **Volume Type:** Select your required volume type.
  - EVS
  - SFS
  - OBS
- **Associate Volume:** Select the volume to be associated, that is, the PV.

**Step 4** Click **Create**.

----End

### Using YAML

Example YAML for EVS

- **failure-domain.beta.kubernetes.io/region:** region where the cluster is located.
- **failure-domain.beta.kubernetes.io/zone:** AZ where the EVS volume is created. It must be the same as the AZ planned for the workload.

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: pvc-test
  namespace: default
  annotations:
    everest.io/disk-volume-type: SAS # EVS disk type.

  volume.beta.kubernetes.io/storage-provisioner: everest-csi-provisioner
  everest.io/enterprise-project-id: 86bfc701-9d9e-4871-a318-6385aa368183 # (Optional) If an enterprise
  project is specified, ensure that the value is the same as that of everest.io/enterprise-project-id specified
  in the PV. Otherwise, the PVC cannot be bound.
  labels:
    failure-domain.beta.kubernetes.io/region: eu-west-101
    failure-domain.beta.kubernetes.io/zone: eu-west-101a
spec:
  accessModes:
    - ReadWriteOnce # The value must be ReadWriteOnce for EVS.
  resources:
    requests:
      storage: 10Gi
  storageClassName: csi-disk # Storage class name. The value is csi-disk for EVS.
  volumeName: cce-evs-test # PV name.
```

### Example YAML for SFS:

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: pvc-sfs-test
  namespace: default
  annotations:
    volume.beta.kubernetes.io/storage-provisioner: everest-csi-provisioner
spec:
  accessModes:
    - ReadWriteMany          # The value must be ReadWriteMany for SFS.
  resources:
    requests:
      storage: 100Gi        # Requested PVC capacity.
  storageClassName: csi-nas # Storage class name. The value is csi-nas for SFS.
  volumeName: cce-sfs-test # PV name.
```

### Example YAML for OBS:

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: pvc-obs-test
  namespace: default
  annotations:
    everest.io/obs-volume-type: STANDARD          # OBS bucket type. Currently, standard
(STANDARD) and infrequent access (WARM) are supported.
    csi.storage.k8s.io/fstype: s3fs              # File type. obsfs indicates to create a parallel file
system (recommended), and s3fs indicates to create an OBS bucket.
    csi.storage.k8s.io/node-publish-secret-name: test-user
    csi.storage.k8s.io/node-publish-secret-namespace: default
    volume.beta.kubernetes.io/storage-provisioner: everest-csi-provisioner
    everest.io/enterprise-project-id: 86bfc701-9d9e-4871-a318-6385aa368183 # (Optional) If an enterprise
project is specified, ensure that the value is the same as that of everest.io/enterprise-project-id specified
in the PV. Otherwise, the PVC cannot be bound.
spec:
  accessModes:
    - ReadWriteMany          # The value must be ReadWriteMany for OBS.
  resources:
    requests:
      storage: 1Gi          # Requested PVC capacity. This field 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 # Storage class name. The value is csi-obs for OBS.
  volumeName: cce-obs-test # PV name.
```

## Using a Snapshot to Creating a PVC

The disk type and disk mode of the created EVS PVC are consistent with those of the snapshot's source EVS disk.

### Using the CCE Console

- Step 1** Log in to the CCE console.
- Step 2** Click the cluster name and go to the cluster console. Choose **Storage** from the navigation pane, and click the **Snapshots and Backups** tab.
- Step 3** Locate the snapshot for which you want to create a PVC, click **Create PVC**, and specify the PVC name in the displayed dialog box.
- Step 4** Click **Create**.

----End

### Creating from YAML

```

apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: pvc-test
  namespace: default
  annotations:
    everest.io/disk-volume-type: SSD # EVS disk type, which must be the same as that of the source EVS
    disk of the snapshot.
  labels:
    failure-domain.beta.kubernetes.io/region: eu-west-101
    failure-domain.beta.kubernetes.io/zone:
spec:
  accessModes:
  - ReadWriteOnce
  resources:
    requests:
      storage: '10'
  storageClassName: csi-disk
  dataSource:
    name: cce-disksnap-test # Snapshot name
    kind: VolumeSnapshot
    apiGroup: snapshot.storage.k8s.io

```

## 17.9.5 StorageClass

StorageClass describes the storage class used in the cluster. You need to specify StorageClass when creating a PVC or PV. As of now, CCE provides storage classes 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.

You can run the following command to query the storage classes that CCE supports. You can use the CSI plug-in provided by CCE to customize a storage class, which functions similarly as the default storage classes in CCE.

```

# kubectl get sc
NAME          PROVISIONER          AGE      # Storage class for EVS disks
csi-disk      everest-csi-provisioner  17d
csi-nas       everest-csi-provisioner  17d      # Storage class for SFS file systems
csi-obs       everest-csi-provisioner  17d      # Storage class for OBS buckets

```

After a StorageClass is set, PVs can be automatically created and maintained. You only need to specify the StorageClass when creating a PVC, which greatly reduces the workload.

In addition to the predefined storage classes provided by CCE, you can also customize storage classes. The following sections describe the application status, solutions, and methods of customizing 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).

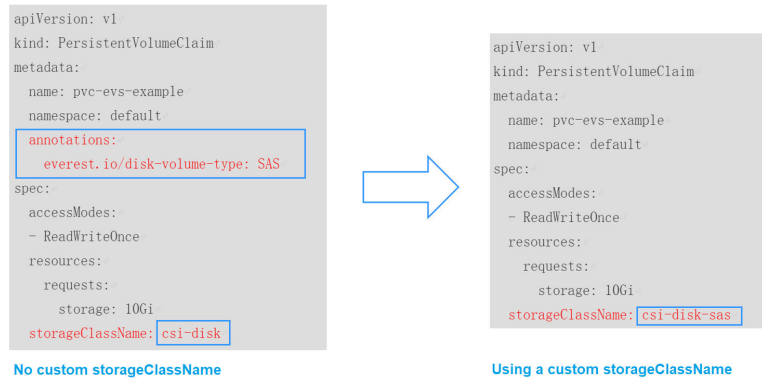
This configuration method may not work if you want to:

- 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
```

## 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: disk.csi.everest.io
  csi.storage.k8s.io/fstype: ext4
  everest.io/disk-volume-type: SAS # High I/O EVS disk type, which cannot be customized.
  everest.io/passthrough: "true"
provisioner: everest-csi-provisioner
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.

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.

- **File storage**

```
# kubectl get sc csi-nas -oyaml
kind: StorageClass
apiVersion: storage.k8s.io/v1
metadata:
  name: csi-nas
provisioner: 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-4d00-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**

```
# 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
  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.storage.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 9s
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

- Use **csi-disk-sas** to create a PVC.

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: sas-disk
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 10Gi
  storageClassName: csi-disk-sas
```

Create a storage class and view its details. As shown below, the object can be created and the value of **STORAGECLASS** is **csi-disk-sas**.

```
# kubectl create -f sas-disk.yaml
persistentvolumeclaim/sas-disk created
# kubectl get pvc
NAME      STATUS  VOLUME                                     CAPACITY  ACCESS MODES  STORAGECLASS  AGE
sas-disk  Bound   pvc-6e2f37f9-7346-4419-82f7-b42e79f7964c  10Gi      RWO           csi-disk-sas  24s
# kubectl get pv
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
```

View the PVC details on the CCE console. On the PV details page, you can see that the disk type is high I/O.

- If **storageClassName** is not specified, the default configuration is used, as shown below.

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: ssd-disk
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 10Gi
```

Create and view the storage resource. You can see that the storage class of PVC **ssd-disk** is **csi-disk-ssd**, indicating that **csi-disk-ssd** is used by default.

```
# kubectl create -f ssd-disk.yaml
persistentvolumeclaim/ssd-disk created
# kubectl get pvc
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
# kubectl get pv
NAME      CAPACITY  ACCESS MODES  RECLAIM POLICY  STATUS
CLAIM     STORAGECLASS  REASON  AGE
pvc-4d2b059c-0d6c-44af-9994-f74d01c78731  10Gi  RWO           Delete          Bound
default/ssd-disk   csi-disk-ssd    15s
pvc-6e2f37f9-7346-4419-82f7-b42e79f7964c  10Gi  RWO           Delete          Bound          default/
sas-disk   csi-disk-sas    17m
```

View the PVC details on the CCE console. On the PV details page, you can see that the disk type is ultra-high I/O.

## 17.9.6 Snapshots and Backups

CCE works with EVS to support snapshots. A snapshot is a complete copy or image of EVS disk data at a certain point of time, which can be used for data DR.

You can create snapshots to rapidly save the disk data at specified time points. In addition, you can use snapshots to create new disks so that the created disks will contain the snapshot data in the beginning.

### Precautions

- The snapshot function is available **only for clusters of v1.15 or later** and requires the CSI-based everest add-on.
- The subtype (common I/O, high I/O, or ultra-high I/O), disk mode (SCSI or VBD), sharing status, and capacity of an EVS disk created from a snapshot must be the same as those of the disk associated with the snapshot. These attributes cannot be modified after being queried or set.
- Snapshots can be created only for available or in-use CSI disks. During the free trial, you can create up to 7 snapshots per disk.

### Application Scenario

The snapshot feature helps address your following needs:

- **Routine data backup**

You can create snapshots for EVS disks regularly and use snapshots to recover your data in case that data loss or data inconsistency occurred due to misoperations, viruses, or attacks.

- **Rapid data restoration**

You can create a snapshot or multiple snapshots before an OS change, application software upgrade, or a service data migration. If an exception occurs during the upgrade or migration, service data can be rapidly restored to the time point when the snapshot was created.

For example, a fault occurred on system disk A of ECS A, and therefore ECS A cannot be started. Because system disk A is already faulty, the data on system disk A cannot be restored by rolling back snapshots. In this case, you can use an existing snapshot of system disk A to create EVS disk B and attach it to ECS B that is running properly. Then, ECS B can read data from system disk A using EVS disk B.

 **NOTE**

The snapshot capability provided by CCE is the same as the CSI snapshot function provided by the Kubernetes community. EVS disks can be created only based on snapshots, and snapshots cannot be rolled back to source EVS disks.

- **Rapid deployment of multiple services**

You can use a snapshot to create multiple EVS disks containing the same initial data, and these disks can be used as data resources for various services, for example, data mining, report query, and development and testing. This method protects the initial data and creates disks rapidly, meeting the diversified service data requirements.

## Creating a Snapshot

### Using the CCE Console

- Step 1** Log in to the CCE console.
- Step 2** Go to the cluster details page, choose **Storage** from the navigation pane, and click the **Snapshots and Backups** tab.
- Step 3** Click **Create Snapshot** in the upper right corner. In the dialog box displayed, set related parameters.
- **Snapshot Name:** Enter a snapshot name.
  - **Storage:** Select the PVC for which you want to create a snapshot.
- Step 4** Click **Create**.

----End

### Using YAML

```
kind: VolumeSnapshot
apiVersion: snapshot.storage.k8s.io/v1beta1
metadata:
  finalizers:
    - snapshot.storage.kubernetes.io/volumesnapshot-as-source-protection
    - snapshot.storage.kubernetes.io/volumesnapshot-bound-protection
  name: cce-disksnap-test
  namespace: default
spec:
  source:
    persistentVolumeClaimName: pvc-evs-test # PVC name. Only an EVS PVC can be created.
    volumeSnapshotClassName: csi-disk-snapclass
```

## Using a Snapshot to Creating a PVC

The disk type and disk mode of the created EVS PVC are consistent with those of the snapshot's source EVS disk.

### Using the CCE Console

- Step 1** Log in to the CCE console.
- Step 2** Go to the cluster details page, choose **Storage** from the navigation pane, and click the **PersistentVolumeClaims (PVCs)** tab.
- Step 3** Click **Create PVC** in the upper right corner. In the dialog box displayed, set the PVC parameters.
- **Creation Mode:** Select **Snapshot**.
  - **PVC Name:** name of a PVC.
  - **Snapshot:** Select the snapshot to be used.
- Step 4** Click **Create**.

----End

### Using YAML

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: pvc-test
```

```
namespace: default
annotations:
  everest.io/disk-volume-type: SSD # EVS disk type, which must be the same as that of the source EVS
  disk of the snapshot.
labels:
  failure-domain.beta.kubernetes.io/region: eu-west-101
  failure-domain.beta.kubernetes.io/zone: eu-west-101a
spec:
  accessModes:
  - ReadWriteOnce
  resources:
    requests:
      storage: '10'
  storageClassName: csi-disk
  dataSource:
    name: cce-disksnap-test # Snapshot name
    kind: VolumeSnapshot
    apiGroup: snapshot.storage.k8s.io
```

## 17.9.7 Using a Custom AK/SK to Mount an OBS Volume

### Scenario

You can solve this issue by using Everest 1.2.8 and later versions to use custom access keys for different IAM users.

### Prerequisites

- The everest add-on version must be 1.2.8 or later.
- The cluster version must be 1.15.11 or later.

### Disabling Auto Key Mounting

The key you uploaded is used by default when mounting an OBS volume. That is, all IAM users under your account will use the same key to mount OBS buckets, and they have the same permissions on buckets. This setting does not allow you to configure differentiated permissions for different IAM users.

If you have uploaded the AK/SK, you are advised to disable the automatic mounting of access keys by enabling the **disable\_auto\_mount\_secret** parameter in the everest add-on to prevent IAM users from performing unauthorized operations. In this way, the access keys uploaded on the console will not be used when creating OBS volumes.

#### NOTE

- When enabling **disable-auto-mount-secret**, ensure that no OBS volume exists in the cluster. A workload mounted with an OBS volume, when scaled or restarted, will fail to remount the OBS volume because it needs to specify the access key but is prohibited by **disable-auto-mount-secret**.
- If **disable-auto-mount-secret** is set to **true**, an access key must be specified when a PV or PVC is created. Otherwise, the OBS volume fails to be mounted.

#### **kubectl edit ds everest-csi-driver -nkube-system**

Search for **disable-auto-mount-secret** and set it to **true**.

```

- /bin/sh
- c
- /var/paas/everest-csi-driver/everest-csi-driver --call-mode=kubelet --drivers=*.local.csi.everest.io
--aksk-secret-name=paas.aksk --iam-endpoint=https://iam.cn-north-7.ulangab.huawei.com:443 --evs-endpoint=https://evs.cn-north-7.ulangab.huawei.com:443
--ecs-endpoint=https://ecs.cn-north-7.ulangab.huawei.com:443 --sfs-endpoint=https://sfs.cn-north-7.ulangab.huawei.com:443
--obs-endpoint=https://obs.cn-north-7.ulangab.huawei.com:443 --sfsturbo-endpoint=https://sfs-turbo.cn-north-7.myhuaweicloud.com:443
--bms-endpoint=https://bms.cn-north-7.ulangab.huawei.com:443 --ims-endpoint=https://ims.cn-north-7.ulangab.huawei.com:443
--feature-gates=supporthis=fake --project-id=09315d9d0ff40c531a9c3256794989
--cluster-id=827dced9-c2ad-11ea-bf3e-0255ac1036e6 --default-vpc-id=0f090290-2b77-48ae-a601-0e746f350265
--disable-auto-mount-secret=true --cluster-version=v1.19.10-r0 --v=2 1>/var/paas/sys/log/everest-csi-driver/everest-csi-driver-standalone.log
2>&1
ENV:

```

Run `:wq` to save the settings and exit. Wait until the pod is restarted.

## Creating a Secret Using an Access Key

**Step 1** Obtain an access key.

For details, see [Obtaining Access Keys \(AK and SK\)](#).

**Step 2** Encode the keys using Base64. (Assume that the AK is xxx and the SK is yyy.)

```
echo -n xxx|base64
```

```
echo -n yyy|base64
```

Record the encoded AK and SK.

**Step 3** Create a YAML file for the secret, for example, `test-user.yaml`.

```

apiVersion: v1
data:
  access.key: WE5WWVhVNU*****
  secret.key: Nnk4emJyZ0*****
kind: Secret
metadata:
  name: test-user
  namespace: default
  labels:
    secret.kubernetes.io/used-by: csi
type: cfe/secure-opaque

```

Specifically:

Parameter	Description
access.key	Base64-encoded AK.
secret.key	Base64-encoded SK.
name	Secret name.
namespace	Namespace of the secret.
secret.kubernetes.io/used-by: csi	You need to add this label in the YAML file if you want to make it available on the CCE console when you create an OBS PV/PVC.
type	Secret type. The value must be <b>cfe/secure-opaque</b> . When this type is used, the data entered by users is automatically encrypted.

**Step 4** Create the secret.

```
kubectl create -f test-user.yaml
```

----End

## Mounting a Secret When Statically Creating an OBS Volume

After a secret is created using the AK/SK, you can associate the secret with the PV to be created and then use the AK/SK in the secret to mount an OBS volume.

- Step 1** Log in to the OBS console, create an OBS bucket, and record the bucket name and storage class. The parallel file system is used as an example.
- Step 2** Create a YAML file for the PV, for example, **pv-example.yaml**.

```
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:
    nodePublishSecretRef:
      name: test-user
      namespace: default
    driver: obs.csi.everest.io
    fsType: obsfs
    volumeAttributes:
      everest.io/obs-volume-type: STANDARD
      everest.io/region: eu-west-101
      storage.kubernetes.io/csiProvisionerIdentity: everest-csi-provisioner
    volumeHandle: obs-normal-static-pv
    persistentVolumeReclaimPolicy: Delete
    storageClassName: csi-obs
```

Parameter	Description
nodePublishSecretRef	Secret specified during the mounting. <ul style="list-style-type: none"> <li>• <b>name</b>: name of the secret</li> <li>• <b>namespace</b>: namespace of the secret</li> </ul>
fsType	File type. The value can be <b>obsfs</b> or <b>s3fs</b> . If the value is <b>s3fs</b> , an OBS bucket is created and mounted using s3fs. If the value is <b>obsfs</b> , an OBS parallel file system is created and mounted using obsfs. You are advised to set this field to <b>obsfs</b> .
volumeHandle	OBS bucket name.

- Step 3** Create the PV.

**kubectl create -f pv-example.yaml**

After a PV is created, you can create a PVC and associate it with the PV.

- Step 4** Create a YAML file for the PVC, for example, **pvc-example.yaml**.

**Example YAML file for the PVC:**

```
apiVersion: v1
kind: PersistentVolumeClaim
```



```

metadata:
  annotations:
    csi.storage.k8s.io/node-publish-secret-name: test-user
    csi.storage.k8s.io/node-publish-secret-namespace: default
    volume.beta.kubernetes.io/storage-provisioner: everest-csi-provisioner
    everest.io/obs-volume-type: STANDARD
    csi.storage.k8s.io/fstype: obsfs
  name: obs-secret
  namespace: default
spec:
  accessModes:
  - ReadWriteMany
  resources:
    requests:
      storage: 1Gi
  storageClassName: csi-obs
  volumeName: pv-obs-example

```

Parameter	Description
csi.storage.k8s.io/node-publish-secret-name	Name of the secret
csi.storage.k8s.io/node-publish-secret-namespace	Namespace of the secret

**Step 5** Create the PVC.

**kubectl create -f pvc-example.yaml**

After the PVC is created, you can create a workload and associate it with the PVC to create volumes.

----End

## Mounting a Secret When Dynamically Creating an OBS Volume

When dynamically creating an OBS volume, you can use the following method to specify a secret:

**Step 1** Create a YAML file for the PVC, for example, **pvc-example.yaml**.

```

apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  annotations:
    csi.storage.k8s.io/node-publish-secret-name: test-user
    csi.storage.k8s.io/node-publish-secret-namespace: default
    everest.io/obs-volume-type: STANDARD
    csi.storage.k8s.io/fstype: obsfs
  name: obs-secret
  namespace: default
spec:
  accessModes:
  - ReadWriteMany
  resources:
    requests:
      storage: 1Gi
  storageClassName: csi-obs

```

Parameter	Description
csi.storage.k8s.io/node-publish-secret-name	Name of the secret
csi.storage.k8s.io/node-publish-secret-namespace	Namespace of the secret

**Step 2** Create the PVC.

**kubectl create -f pvc-example.yaml**

After the PVC is created, you can create a workload and associate it with the PVC to create volumes.

----End

**Verification**

You can use a secret of an IAM user to mount an OBS volume. Assume that a workload named **obs-secret** is created, the mount path in the container is **/temp**, and the IAM user has the CCE **ReadOnlyAccess** and **Tenant Guest** permissions.

1. Query the name of the workload pod.

**kubectl get po | grep obs-secret**

Expected outputs:

```
obs-secret-5cd558f76f-vxslv    1/1    Running    0    3m22s
```

2. Query the objects in the mount path. In this example, the query is successful.

**kubectl exec obs-secret-5cd558f76f-vxslv -- ls -l /temp/**

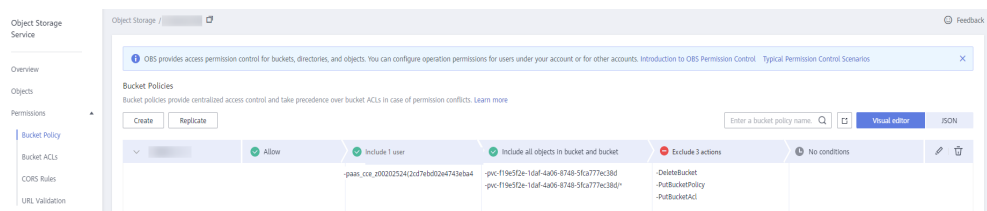
3. Write data into the mount path. In this example, the write operation fails.

**kubectl exec obs-secret-5cd558f76f-vxslv -- touch /temp/test**

Expected outputs:

```
touch: setting times of '/temp/test': No such file or directory
command terminated with exit code 1
```

4. Set the read/write permissions for the IAM user who mounted the OBS volume by referring to the bucket policy configuration.



5. Write data into the mouth path again. In this example, the write operation succeeded.

**kubectl exec obs-secret-5cd558f76f-vxslv -- touch /temp/test**

6. Check the mount path in the container to see whether the data is successfully written.

**kubectl exec obs-secret-5cd558f76f-vxslv -- ls -l /temp/**

Expected outputs:

-rwxrwxrwx 1 root root 0 Jun 7 01:52 test

## 17.9.8 Setting Mount Options

### Scenario

You can mount cloud storage volumes to your containers and use these volumes as local directories.

This section describes how to set mount options when mounting SFS and OBS volumes. You can set mount options in a PV and bind the PV to a PVC.

Alternatively, set mount options in a StorageClass and use the StorageClass to create a PVC. In this way, PVs can be dynamically created and inherit mount options configured in the StorageClass by default.

### SFS Volume Mount Options

The everest add-on in CCE presets the options described in [Table 17-98](#) for mounting SFS volumes.

**Table 17-98** Preset mount options for SFS volumes

Option	Description
vers=3	File system version. Currently, only NFSv3 is supported, Value: <b>3</b>
nolock	Whether to lock files on the server using the NLM protocol. If <b>no</b> lock is selected, the lock is valid for applications on one host. For applications on another host, the lock is invalid.
timeo=600	Waiting time before the NFS client retransmits a request. The unit is 0.1 seconds. Recommended value: <b>600</b>
hard/soft	Mounting mode. <ul style="list-style-type: none"> <li>• <b>hard</b>: If the NFS request times out, the client keeps resending the request until the request is successful.</li> <li>• <b>soft</b>: If the NFS request times out, the client returns an error to the invoking program.</li> </ul> The default value is <b>hard</b> .

### OBS Volume Mount Options

When mounting file storage, the everest add-on presets the options described in [Table 17-99](#) and [Table 17-100](#) by default. The options in [Table 17-99](#) are mandatory.

**Table 17-99** Mandatory mount options configured by default

Option	Description
use_ino	If enabled, obsfs allocates the <b>ino</b> number. Enabled by default in read/write mode.
big_writes	If configured, the maximum size of the cache can be modified.
nonempty	Allows non-empty mount paths.
allow_other	Allows other users to access the parallel file system.
no_check_certificate	Disables server certificate verification.
enable_noobj_cache	Enables cache entries for objects that do not exist, which can improve performance. Enabled by default in object bucket read/write mode. <b>This option is no longer set by default since everest 1.2.40.</b>
sigv2	Specifies the signature version. Used by default in object buckets.

**Table 17-100** Optional mount options configured by default

Option	Description
max_write=131072	If specified, obsfs allocates the <b>ino</b> number. Enabled by default in read/write mode.
ssl_verify_hostname=0	Disables verifying the SSL certificate based on the host name.
max_background=100	Allows setting the maximum number of waiting requests in the background. Used by default in parallel file systems.
public_bucket=1	If set to <b>1</b> , public buckets are mounted anonymously. Enabled by default in object bucket read/write mode.

You can log in to the node to which the pod is scheduled and view all mount options used for mounting the OBS volume in the process details.

- Object bucket: `ps -ef | grep s3fs`  

```
root 22142 1 0 Jun03 ? 00:00:00 /usr/bin/s3fs pvc-82fe2cbe-3838-43a2-8afb-f994e402fb9d /mnt/paas/kubernetes/kubelet/pods/0b13ff68-4c8e-4a1c-b15c-724fd4d64389/volumes/kubernetes.io~csi/pvc-82fe2cbe-3838-43a2-8afb-f994e402fb9d/mount -o url=https://{{endpoint}}:443 -o endpoint=xxxxxx -o passwd_file=/opt/everest-host-connector/1622707954357702943_obstmpcred/pvc-82fe2cbe-3838-43a2-8afb-f994e402fb9d -o nonempty -o big_writes -o enable_noobj_cache -o sigv2 -o allow_other -o no_check_certificate -o ssl_verify_hostname=0 -o max_write=131072 -o multipart_size=20 -o umask=0
```
- Parallel file system: `ps -ef | grep obsfs`  

```
root 1355 1 0 Jun03 ? 00:03:16 /usr/bin/obsfs pvc-86720bb9-5aa8-4cde-9231-5253994f8468 /mnt/paas/kubernetes/kubelet/pods/c959a91d-
```

```
eced-4b41-91c6-96cbd65324f9/volumes/kubernetes.io~csi/  
pvc-86720bb9-5aa8-4cde-9231-5253994f8468/mount -o url=https://{{endpoint}}:443 -o  
endpoint=xxxxxx -o passwd_file=/opt/everest-host-connector/1622714415305160399_obstmpcred/  
pvc-86720bb9-5aa8-4cde-9231-5253994f8468 -o allow_other -o nonempty -o big_writes -o use_ino -  
o no_check_certificate -o ssl_verify_hostname=0 -o umask=0027 -o max_write=131072 -o  
max_background=100 -o uid=10000 -o gid=10000
```

## Prerequisites

- The everest add-on version must be **1.2.8 or later**.
- The add-on identifies the mount options and transfers them to the underlying storage resources, which determine whether the specified options are valid.

## Notes and Constraints

Mount options cannot be configured for secure containers.

## Setting Mount Options in a PV

You can use the **mountOptions** field to set mount options in a PV. The options you can configure in **mountOptions** are listed in [SFS Volume Mount Options](#) and [OBS Volume Mount Options](#).

```
apiVersion: v1  
kind: PersistentVolume  
metadata:  
  name: pv-obs-example  
  annotations:  
    pv.kubernetes.io/provisioned-by: everest-csi-provisioner  
spec:  
  mountOptions:  
    - umask=0027  
    - uid=10000  
    - gid=10000  
  accessModes:  
    - ReadWriteMany  
  capacity:  
    storage: 1Gi  
  claimRef:  
    apiVersion: v1  
    kind: PersistentVolumeClaim  
    name: pvc-obs-example  
    namespace: default  
  csi:  
    driver: obs.csi.everest.io  
    fsType: obsfs  
    volumeAttributes:  
      everest.io/obs-volume-type: STANDARD  
      everest.io/region: eu-west-101  
      storage.kubernetes.io/csiProvisionerIdentity: everest-csi-provisioner  
    volumeHandle: obs-normal-static-pv  
    persistentVolumeReclaimPolicy: Delete  
    storageClassName: csi-obs
```

After a PV is created, you can create a PVC and bind it to the PV, and then mount the PV to the container in the workload.

## Setting Mount Options in a StorageClass

You can use the **mountOptions** field to set mount options in a StorageClass. The options you can configure in **mountOptions** are listed in [SFS Volume Mount Options](#) and [OBS Volume Mount Options](#).

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: csi-obs-mount-option
mountOptions:
- umask=0027
- uid=10000
- gid=10000
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
```

After the StorageClass is configured, you can use it to create a PVC. By default, the dynamically created PVs inherit the mount options set in the StorageClass.

## 17.9.9 Deployment Examples

### 17.9.9.1 EVS Volumes

#### 17.9.9.1.1 Using EVS Volumes

#### Prerequisites

You have created a CCE cluster and installed the CSI plug-in ([everest](#)) in the cluster.

#### Notes and Constraints

- EVS disks cannot be attached across AZs and cannot be used by multiple workloads, multiple pods of the same workload, or multiple jobs.
- Data in a shared disk cannot be shared between nodes in a CCE cluster. If the same EVS disk is attached to multiple nodes, read and write conflicts and data cache conflicts may occur. When creating a Deployment, you are advised to create only one pod if you want to use EVS disks.
- For clusters earlier than v1.19.10, if an HPA policy is used to scale out a workload with EVS volumes mounted, the existing pods cannot be read or written when a new pod is scheduled to another node.  
For clusters of v1.19.10 and later, if an HPA policy is used to scale out a workload with EVS volume mounted, a new pod cannot be started because EVS disks cannot be attached.
- When you create a StatefulSet and add a cloud storage volume, existing EVS volumes cannot be used.
- EVS disks that have partitions or have non-ext4 file systems cannot be imported.
- Container storage in CCE clusters of Kubernetes 1.13 or later version supports encryption. Currently, E2E encryption is supported only in certain regions.
- EVS volumes cannot be created in specified enterprise projects. Only the default enterprise project is supported.

## Creating an EVS Disk

**Step 1** Log in to the CCE console. In the navigation pane, choose **Resource Management** > **Storage**. Click **Buy EVS Disk**.

**Step 2** Configure basic disk information. [Table 17-101](#) describes the parameters.

**Table 17-101** Configuring basic disk information

Parameter	Description
* PVC Name	<b>New PVC Name:</b> name of the PVC to be created. A storage volume is automatically created when a PVC is created. One PVC corresponds to one storage volume. The storage volume name is automatically generated when the PVC is created.
Cluster Name	Cluster where the EVS disk is deployed.
Namespace	Select the namespace where the EVS disk is deployed. If you do not need to select a namespace, retain the default value.
Volume Capacity (GB)	Size of the storage to be created.
Access Mode	Access permissions of user applications on storage resources (PVs). <ul style="list-style-type: none"> <li>• <b>ReadWriteOnce (RWO):</b> The volume can be mounted as read-write by a single node, and data reading and writing are supported based on a non-shared EVS volume. EVS volumes in RWO mode are supported since v1.13.10-r1.</li> </ul>
Primary AZ	AZ to which the volume belongs.
Type	Type of the new EVS disk. <ul style="list-style-type: none"> <li>• <b>High I/O:</b> uses serial attached SCSI (SAS) drives to store data.</li> <li>• <b>Ultra-high I/O:</b> uses solid state disk (SSD) drives to store data.</li> </ul>
Storage Format	The default value is <b>CSI</b> and cannot be changed.

**Step 3** Click **Buy Now**. Review your order, click **Submit**, and wait until the creation is successful.

The file system is displayed in the list. When its status becomes **Normal**, the file system is created successfully.

**Step 4** Click the volume name to view detailed information about the volume.

----End

## Adding an EVS Volume

**Step 1** Create a workload or job by referring to [Creating a Deployment](#), [Creating a StatefulSet](#), or [Creating a Job](#). During creation, expand **Data Storage** after adding a container. On the **Cloud Volume** tab page, click **Add Cloud Volume**.

**Step 2** Set the storage volume type to **EVS**.

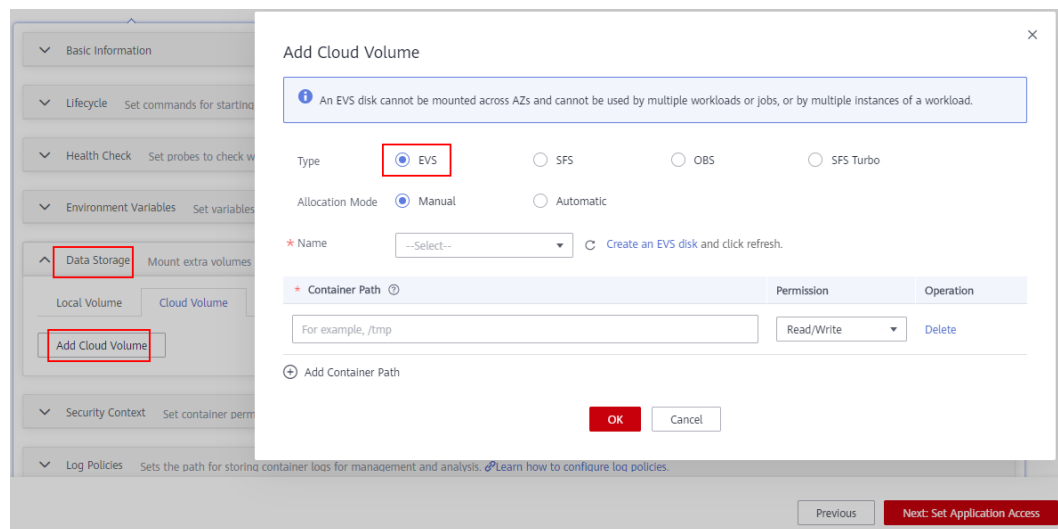
**Table 17-102** Parameters required for mounting an EVS volume

Parameter	Description
<b>Type</b>	<p><b>EVS:</b> You can use EVS disks the same way you use traditional hard disks on servers. EVS disks deliver higher data reliability and I/O throughput and are easy to use. They can be used for file systems, databases, or other system software and applications that require block storage resources.</p> <p><b>CAUTION</b></p> <ul style="list-style-type: none"> <li>To attach an EVS disk to a workload, you must set the number of pods to <b>1</b> when creating the workload. If multiple pods are configured, you cannot attach EVS disks.</li> <li>When you create a StatefulSet and add a cloud storage volume, existing EVS volumes cannot be used.</li> <li>EVS disks cannot be attached across AZs and cannot be used by multiple workloads, multiple pods of the same workload, or multiple jobs.</li> </ul>
<b>Allocation Mode</b>	
Manual	<p>Select a created disk. If no disk is available, follow the prompts to create one.</p> <p>For the same cluster and namespace, you can use an existing storage volume when creating a Deployment (with <b>Allocation Mode</b> set to <b>Manual</b>).</p> <p>When creating a StatefulSet, you can only use a volume automatically allocated by the system (only <b>Automatic</b> is available for <b>Allocation Mode</b>).</p>



Parameter	Description
Automatic	<p>If you select <b>Automatic</b>, you need to configure the following items:</p> <ol style="list-style-type: none"> <li>1. <b>Access Mode:</b> permissions of user applications on storage resources (PVs). <ul style="list-style-type: none"> <li>- <b>ReadWriteOnce (RWO):</b> The volume can be mounted as read-write by a single node, and data reading and writing are supported based on a non-shared EVS volume. EVS volumes in RWO mode are supported since v1.13.10-r1.</li> </ul> </li> <li>2. <b>Availability Zone:</b> AZ where the storage volume is located. Only the AZ where the node is located can be selected.</li> <li>3. <b>Sub-Type:</b> Select a storage subtype. <ul style="list-style-type: none"> <li>- <b>High I/O:</b> uses serial attached SCSI (SAS) drives to store data.</li> <li>- <b>Ultra-high I/O:</b> uses solid state disk (SSD) drives to store data.</li> </ul> </li> <li>4. <b>Storage Capacity:</b> Enter the storage capacity in the unit of GB. Ensure that the storage capacity quota is not exceeded; otherwise, creation will fail.</li> <li>5. <b>Storage Format:</b> The default value is <b>CSI</b>. The container storage interface (CSI) is used to establish a set of standard storage management interfaces between Kubernetes and external storage systems to provide storage services for containers.</li> </ol>
Add Container Path	<ol style="list-style-type: none"> <li>1. Click <b>Add Container Path</b>.</li> <li>2. <b>Container Path:</b> Enter the container path to which the data volume is mounted. <p><b>NOTICE</b></p> <ul style="list-style-type: none"> <li>- Do not mount a data volume to a system directory such as <code>/</code> or <code>/var/run</code>; this action may cause a container error to occur. You are advised to mount the volume to an empty directory. If the directory is not empty, ensure that there are no files affecting container startup in the directory. Otherwise, such files will be replaced, resulting in failures to start the container and create the workload.</li> <li>- If the volume is mounted to a high-risk directory, you are advised to use an account with minimum permissions to start the container; otherwise, high-risk files on the host machine may be damaged.</li> </ul> </li> <li>3. Set permissions. <ul style="list-style-type: none"> <li>- <b>Read-only:</b> You can only read the data volumes mounted to the path.</li> <li>- <b>Read/Write:</b> You can modify the data volumes mounted to the path. Newly written data is not migrated if the container is migrated, which may cause a data loss.</li> </ul> </li> </ol>

**Figure 17-138** Adding a cloud volume



**Step 3** Click **OK**.

----End

## Importing an EVS Disk

CCE allows you to import existing EVS disks.

### NOTE

An EVS disk can be imported into only one namespace. If an EVS disk has been imported into a namespace, it is invisible in other namespaces and cannot be imported again. **If you want to import an EVS disk that has file system (ext4) formatted, ensure that no partition has been created for the disk. Otherwise, data may be lost.**

**Step 1** Log in to the CCE console. In the navigation pane, choose **Resource Management** > **Storage**. On the **EVS** tab page, click **Import**.

**Step 2** Select one or more EVS disks that you want to import. Then, click **OK**.

----End

## Yearly/Monthly-Billed EVS Disks

1. Create a yearly/monthly-billed EVS disk with the required capacity on the cloud server console.
2. Back on the **EVS** tab page of the CCE console, click **Import**. Select the EVS disk you just created, and click **OK**.

## Unbinding an EVS Disk

After an EVS volume is successfully created or imported, the EVS volume is automatically bound to the current cluster and cannot be used by other clusters. When the volume is unbound from the cluster, other clusters can still use the volume.

If the EVS volume has been mounted to a workload, it cannot be unbound from the cluster.

**Step 1** Log in to the CCE console. In the navigation pane, choose **Resource Management** > **Storage**. In the EVS disk list, click **Unbind** next to the target EVS disk.

**Step 2** Confirm the unbinding, and click **OK**.

----End

## Related Operations

After an EVS volume is created, you can perform operations described in [Table 17-103](#).

**Table 17-103** Other operations

Operation	Description
Deleting an EVS volume	<ol style="list-style-type: none"> <li>1. Select the EVS volume to be deleted and click <b>Delete</b> in the <b>Operation</b> column.</li> <li>2. Follow the prompts to delete the EVS volume.</li> </ol>

### 17.9.9.1.2 Creating a Pod Mounted with an EVS Volume

#### Scenario

After an EVS volume is created or imported to CCE, you can mount it to a workload.

#### NOTICE

EVS disks cannot be attached across AZs. Before mounting a volume, you can run the **kubectl get pvc** command to query the available PVCs in the AZ where the current cluster is located.

#### Prerequisites

You have created a CCE cluster and installed the CSI plug-in ([everest](#)) in the cluster.

#### Notes and Constraints

The following configuration example applies to clusters of Kubernetes 1.15 or later.

#### Using EVS Volumes for Deployments

**Step 1** Use **kubectl** to connect to the cluster. For details, see [Connecting to a Cluster Using kubectl](#).

**Step 2** Run the following commands to configure the **evs-deployment-example.yaml** file, which is used to create a Deployment.

### touch evs-deployment-example.yaml

#### vi evs-deployment-example.yaml

Example of mounting an EVS volume to a Deployment (PVC-based, shared volume):

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: evs-deployment-example
  namespace: default
spec:
  replicas: 1
  selector:
    matchLabels:
      app: evs-deployment-example
  template:
    metadata:
      labels:
        app: evs-deployment-example
    spec:
      containers:
        - image: nginx
          name: container-0
          volumeMounts:
            - mountPath: /tmp
              name: pvc-evs-example
      imagePullSecrets:
        - name: default-secret
      restartPolicy: Always
      volumes:
        - name: pvc-evs-example
          persistentVolumeClaim:
            claimName: pvc-evs-auto-example
```

**Table 17-104** Key parameters

Parent Parameter	Parameter	Description
spec.template.spec.containers.volumeMounts	name	Name of the volume mounted to the container.
spec.template.spec.containers.volumeMounts	mountPath	Mount path of the container. In this example, the volume is mounted to the <b>/tmp</b> directory.
spec.template.spec.volumes	name	Name of the volume.
spec.template.spec.volumes.persistentVolumeClaim	claimName	Name of an existing PVC.

#### NOTE

**spec.template.spec.containers.volumeMounts.name** and **spec.template.spec.volumes.name** must be consistent because they have a mapping relationship.

**Step 3** Run the following command to create the workload:

```
kubectl create -f evs-deployment-example.yaml
```

```
----End
```

## Using EVS Volumes for StatefulSets

**Step 1** Use kubectl to connect to the cluster. For details, see [Connecting to a Cluster Using kubectl](#).

**Step 2** Run the following commands to configure the `evs-statefulset-example.yaml` file, which is used to create a Deployment.

```
touch evs-statefulset-example.yaml
```

```
vi evs-statefulset-example.yaml
```

Mounting an EVS volume to a StatefulSet (PVC template-based, non-shared volume):

### Example YAML:

```
apiVersion: apps/v1
kind: StatefulSet
metadata:
  name: evs-statefulset-example
  namespace: default
spec:
  replicas: 1
  selector:
    matchLabels:
      app: evs-statefulset-example
  template:
    metadata:
      labels:
        app: evs-statefulset-example
    spec:
      containers:
        - name: container-0
          image: 'nginx:latest'
          volumeMounts:
            - name: pvc-evs-auto-example
              mountPath: /tmp
          restartPolicy: Always
          imagePullSecrets:
            - name: default-secret
      volumeClaimTemplates:
        - metadata:
            name: pvc-evs-auto-example
            namespace: default
            labels:
              failure-domain.beta.kubernetes.io/region: eu-west-101
              failure-domain.beta.kubernetes.io/zone:
            annotations:
              everest.io/disk-volume-type: SAS
          spec:
            accessModes:
              - ReadWriteOnce
            resources:
              requests:
                storage: 10Gi
            storageClassName: csi-disk
      serviceName: evs-statefulset-example-headless
      updateStrategy:
        type: RollingUpdate
```

**Table 17-105** Key parameters

Parent Parameter	Parameter	Description
metadata	name	Name of the created workload.
spec.template.spec.containers	image	Image of the workload.
spec.template.spec.containers.volumeMount	mountPath	Mount path of the container. In this example, the volume is mounted to the <b>/tmp</b> directory.
spec	serviceName	Service corresponding to the workload. For details about how to create a Service, see <a href="#">Creating a StatefulSet</a> .

 **NOTE**

`spec.template.spec.containers.volumeMounts.name` and `spec.volumeClaimTemplates.metadata.name` must be consistent because they have a mapping relationship.

**Step 3** Run the following command to create the workload:

```
kubectl create -f evs-statefulset-example.yaml
```

----End

## Verifying Persistent Storage of an EVS Volume

**Step 1** Query the pod and EVS files of the deployed workload (for example, `evs-statefulset-example`).

1. Run the following command to query the pod name of the workload:  

```
kubectl get po | grep evs-statefulset-example
```

Expected outputs:

```
evs-statefulset-example-0 1/1 Running 0 22h
```

2. Run the following command to check whether an EVS volume is mounted to the **/tmp** directory:

```
kubectl exec evs-statefulset-example-0 -- df tmp
```

Expected outputs:

```
/dev/sda 10255636 36888 10202364 1% /tmp
```

**Step 2** Run the following command to create a file named **test** in the **/tmp** directory:

```
kubectl exec evs-statefulset-example-0 -- touch /tmp/test
```

**Step 3** Run the following command to view the file in the **/tmp** directory:

```
kubectl exec evs-statefulset-example-0 -- ls -l /tmp
```

Expected outputs:

```
-rw-r--r-- 1 root root 0 Jun 1 02:50 test
```

**Step 4** Run the following command to delete the pod named **evs-statefulset-example-0**:

```
kubectl delete po evs-statefulset-example-0
```

**Step 5** Check whether the file still exists after the pod is rebuilt.

1. Run the following command to query the name of the rebuilt pod:

```
kubectl get po
```

Expected outputs:

```
evs-statefulset-example-0 1/1 Running 0 2m
```

2. Run the following command to view the file in the **/tmp** directory:

```
kubectl exec evs-statefulset-example-0 -- ls -l /tmp
```

Expected outputs:

```
-rw-r--r-- 1 root root 0 Jun 1 02:50 test
```

3. The **test** file still exists after the pod is rebuilt, indicating that the data in the EVS volume can be persistently stored.

----End

## 17.9.9.2 SFS Turbo Volumes

### 17.9.9.2.1 Using SFS Turbo Volumes

#### Prerequisites

You have created a CCE cluster and installed the CSI plug-in ([everest](#)) in the cluster.

#### Notes and Constraints

- Currently, SFS Turbo file systems cannot be directly created on CCE.
- Only an SFS Turbo file system in the same VPC as the cluster and in the same subnet as the node can be imported.
- Inbound ports (111, 445, 2049, 2051, and 20048) must be enabled for the security group to which the SFS Turbo file system belongs.

#### Importing an SFS Turbo Volume

CCE allows you to import existing SFS Turbo volumes. Currently, only cloud accounts and IAM users with the CCE Administrator permissions can import SFS Turbo volumes.

**Step 1** Log in to the CCE console. In the navigation pane, choose **Resource Management** > **Storage**. On the **SFS Turbo** tab page, click **Import**.

**Step 2** Select one or more SFS Turbo volumes that you want to import.

**Step 3** Select the cluster and namespace to which you want to import the volumes.

**Step 4** Click **Next**. The volumes are displayed in the list. When **PVC Status** becomes **Bound**, the volumes are imported successfully.

----End

#### Adding an SFS Turbo Volume

**Step 1** Create a workload or job by referring to [Creating a Deployment](#), [Creating a StatefulSet](#), [Creating a DaemonSet](#), or [Creating a Job](#). After you have added a

container, choose **Data Storage > Cloud Volume**, and then click **Add Cloud Volume**.

**Step 2** Set the storage volume type to **SFS Turbo**.

**Table 17-106** Parameters for configuring an SFS Turbo volume

Parameter	Parameter Description
Type	<b>SFS Turbo</b> : applicable to DevOps, containerized microservices, and enterprise office applications.
<b>Allocation Mode</b>	
Manual	Select an existing SFS Turbo volume. You need to import SFS Turbo volumes in advance. For details, see <a href="#">Importing an SFS Turbo Volume</a> .
Add Container Path	<p>Configure the following parameters:</p> <ol style="list-style-type: none"> <li><b>subPath</b>: Enter the subpath of the file storage, for example, <code>/tmp</code>. This parameter specifies a subpath inside the referenced volume instead of its root. If this parameter is not specified, the root path is used. Currently, only file storage is supported. The value must be a relative path and cannot start with a slash (/) or ../.</li> <li><b>Container Path</b>: Enter the mount path in the container, for example, <code>/tmp</code>. The mount path must not be a system directory, such as <code>/</code> and <code>/var/run</code>. Otherwise, an exception occurs. You are advised to mount the volume to an empty directory. If the directory is not empty, ensure that there are no files affecting container startup in the directory. Otherwise, such files will be replaced, resulting in failures to start the container and create the workload.</li> </ol> <p><b>NOTICE</b> If the volume is mounted to a high-risk directory, you are advised to use an account with minimum permissions to start the container; otherwise, high-risk files on the host machine may be damaged.</p> <ol style="list-style-type: none"> <li>Set permissions. <ul style="list-style-type: none"> <li><b>Read-only</b>: You can only read the data in the mounted volumes.</li> <li><b>Read/Write</b>: You can modify the data in the mounted volumes. Newly written data is not migrated if the container is migrated, which causes a data loss.</li> </ul> </li> </ol> <p>Click <b>Add Container Path</b> to add multiple settings. Then, click <b>OK</b>.</p>

**Step 3** Click **OK**.

----End



## Unbinding an SFS Turbo Volume

When an SFS Turbo volume is successfully imported to a cluster, the volume is bound to the cluster. The volume can also be imported to other clusters. When the volume is unbound from the cluster, other clusters can still import and use the volume.

If the SFS Turbo volume has been mounted to a workload, the volume cannot be unbound from the cluster.

**Step 1** Log in to the CCE console. In the navigation pane, choose **Resource Management** > **Storage**. In the SFS Turbo volume list, click **Unbind** next to the target volume.

**Step 2** In the dialog box displayed, click **OK**.

----End

### 17.9.9.2.2 Creating a Deployment Mounted with an SFS Turbo Volume

#### Scenario

After an SFS Turbo volume is created or imported to CCE, you can mount the volume to a workload.

#### Prerequisites

You have created a CCE cluster and installed the CSI plug-in ([everest](#)) in the cluster.

#### Notes and Constraints

The following configuration example applies to clusters of Kubernetes 1.15 or later.

#### Procedure

**Step 1** Use `kubectl` to connect to the cluster. For details, see [Connecting to a Cluster Using kubectl](#).

**Step 2** Run the following commands to configure the `sfsturbo-deployment-example.yaml` file, which is used to create a Deployment:

```
touch sfsturbo-deployment-example.yaml
```

```
vi sfsturbo-deployment-example.yaml
```

Example of mounting an SFS Turbo volume to a Deployment (PVC-based, shared volume):

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: sfsturbo-deployment-example           # Workload name
  namespace: default
spec:
  replicas: 1
  selector:
    matchLabels:
```

```

app: sfsturbo-deployment-example
template:
  metadata:
    labels:
      app: sfsturbo-deployment-example
  spec:
    containers:
      - image: nginx
        name: container-0
        volumeMounts:
          - mountPath: /tmp                # Mount path
            name: pvc-sfsturbo-example
        restartPolicy: Always
        imagePullSecrets:
          - name: default-secret
    volumes:
      - name: pvc-sfsturbo-example
        persistentVolumeClaim:
          claimName: pvc-sfsturbo-example    # PVC name

```

**Table 17-107** Key parameters

Parameter	Description
name	Name of the created Deployment.
app	Name of the Deployment.
mountPath	Mount path of the container. In this example, the mount path is <b>/tmp</b> .

 **NOTE**

**spec.template.spec.containers.volumeMounts.name** and **spec.template.spec.volumes.name** must be consistent because they have a mapping relationship.

**Step 3** Run the following command to create the workload:

```

kubectl create -f sfsturbo-deployment-example.yaml
----End

```

### 17.9.9.2.3 Creating a StatefulSet Mounted with an SFS Turbo Volume

#### Scenario

CCE allows you to use an existing SFS Turbo volume to create a StatefulSet.

#### Prerequisites

You have created a CCE cluster and installed the CSI plug-in ([everest](#)) in the cluster.

#### Notes and Constraints

The following configuration example applies to clusters of Kubernetes 1.15 or later.

## Procedure

- Step 1** Create an SFS Turbo volume and record the volume name.
- Step 2** Use kubectl to connect to the cluster. For details, see [Connecting to a Cluster Using kubectl](#).
- Step 3** Create a YAML file for creating the workload. Assume that the file name is `sfsturbo-statefulset-example.yaml`.

**touch sfsturbo-statefulset-example.yaml**

**vi sfsturbo-statefulset-example.yaml**

Configuration example:

```
apiVersion: apps/v1
kind: StatefulSet
metadata:
  name: sfsturbo-statefulset-example
  namespace: default
spec:
  replicas: 1
  selector:
    matchLabels:
      app: sfsturbo-statefulset-example
  template:
    metadata:
      labels:
        app: sfsturbo-statefulset-example
    spec:
      volumes:
        - name: pvc-sfsturbo-example
          persistentVolumeClaim:
            claimName: pvc-sfsturbo-example
      containers:
        - name: container-0
          image: 'nginx:latest'
          volumeMounts:
            - name: pvc-sfsturbo-example
              mountPath: /tmp
          restartPolicy: Always
          imagePullSecrets:
            - name: default-secret
      serviceName: sfsturbo-statefulset-example-headless
  updateStrategy:
    type: RollingUpdate
```

**Table 17-108** Key parameters

Parameter	Description
replicas	Number of pods.
name	Name of the new workload.
image	Image used by the workload.
mountPath	Mount path of a container.
serviceName	Service corresponding to the workload. For details about how to create a Service, see <a href="#">Creating a StatefulSet</a> .
claimName	Name of an existing PVC.

 NOTE

`spec.template.spec.containers.volumeMounts.name` and `spec.template.spec.volumes.name` must be consistent because they have a mapping relationship.

**Step 4** Create the StatefulSet.

```
kubectl create -f sfsturbo-statefulset-example.yaml
----End
```

## Verifying Persistent Storage of an SFS Turbo Volume

**Step 1** Query the pod and SFS Turbo volume of the deployed workload (for example, `sfsturbo-statefulset-example`).

1. Run the following command to query the pod name of the workload:

```
kubectl get po | grep sfsturbo-statefulset-example
```

Expected outputs:

```
sfsturbo-statefulset-example-0 1/1 Running 0 2m5s
```

2. Run the following command to check whether an SFS Turbo volume is mounted to the `/tmp` directory:

```
kubectl exec sfsturbo-statefulset-example-0 -- mount|grep /tmp
```

Expected outputs:

```
192.168.0.108:/ on /tmp type nfs
(rw,relatime,vers=3,rsize=1048576,wsize=1048576,namlen=255,hard,nolock,noresvport,proto=tcp,timeo
=600,retrans=2,sec=sys,mountaddr=192.168.0.108,mountvers=3,mountport=20048,mountproto=tcp,loc
al_lock=all,addr=192.168.0.108)
```

**Step 2** Run the following command to create a file named `test` in the `/tmp` directory:

```
kubectl exec sfsturbo-statefulset-example-0 -- touch /tmp/test
```

**Step 3** Run the following command to view the file in the `/tmp` directory:

```
kubectl exec sfsturbo-statefulset-example-0 -- ls -l /tmp
```

Expected outputs:

```
-rw-r--r-- 1 root root 0 Jun 1 02:50 test
```

**Step 4** Run the following command to delete the pod named `sfsturbo-statefulset-example-0`:

```
kubectl delete po sfsturbo-statefulset-example-0
```

**Step 5** Check whether the file still exists after the pod is rebuilt.

1. Run the following command to query the name of the rebuilt pod:

```
kubectl get po
```

Expected outputs:

```
sfsturbo-statefulset-example-0 1/1 Running 0 2m
```

2. Run the following command to view the file in the `/tmp` directory:

```
kubectl exec sfsturbo-statefulset-example-0 -- ls -l /tmp
```

Expected outputs:

```
-rw-r--r-- 1 root root 0 Jun 1 02:50 test
```

The **test** file still exists after the pod is rebuilt, indicating that the data in the SFS Turbo volume can be persistently stored.

----End

## 17.9.9.3 OBS Volumes

### 17.9.9.3.1 Using OBS Volumes

#### Prerequisites

You have created a CCE cluster and installed the CSI plug-in (**everest**) in the cluster.

#### Notes and Constraints

- CCE clusters of v1.7.3-r8 and earlier do not support OBS volumes. You need to upgrade these clusters or create clusters of a later version that supports OBS.
- Volumes cannot be created in specified enterprise projects. Only the default enterprise project is supported.

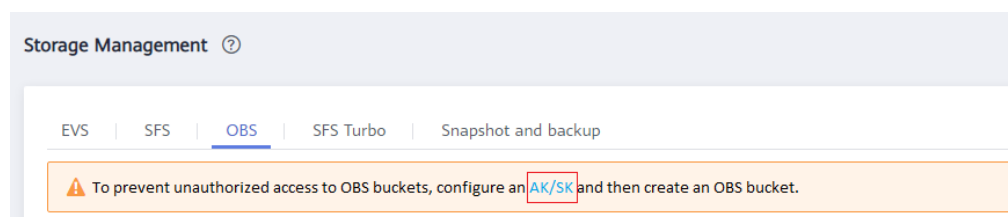
#### Preparations

To mount reliable and stable OBS buckets as volumes, you must create AK/SK before you create OBS buckets.

The procedure for configuring the AK/SK is as follows:

1. Log in to the CCE console. In the navigation pane, choose **Resource Management > Storage**.
2. On the **OBS** tab page, click **AK/SK** in the notice.

**Figure 17-139** Configuring the AK/SK



3. Click , select a key file, and click **Upload** to upload the key file.
4. Select the corresponding workload and click **Restart**.

#### NOTICE

When creating an OBS volume, you must use the AK/SK. If the key file is not uploaded, the pod will fail to be started or OBS data access will be abnormal due to the volume mounting failure.

## Creating an OBS Volume

- Step 1** Log in to the CCE console. In the navigation pane, choose **Resource Management > Storage**.
- Step 2** Click the **OBS** tab and click **Create OBS Bucket**.
- Step 3** Configure basic information, as shown in [Table 17-109](#).

**Table 17-109** Parameters for creating an OBS volume

Parameter	Parameter Description
* PVC Name	Name of the new PVC, which is different from the volume name. The actual volume name is automatically generated when the PV is created by the PVC.  The name contains 3 to 55 characters (excluding the prefix). It must contain lowercase letters, digits, and hyphens (-), and cannot start or end with a hyphen (-).
Cluster Name	Cluster to which the OBS volume belongs.
Namespace	Namespace to which the volume belongs. The default value is <b>default</b> .
Instance Type	Type of the storage instance created on OBS. <ul style="list-style-type: none"> <li>• <b>Parallel file system:</b> If the cluster version is v1.15 or later and the everest add-on version is 1.0.2 or later, parallel file systems that can be mounted by obsfs can be created.</li> <li>• <b>Object bucket:</b> A bucket is a container for storing objects in OBS. OBS provides flat storage in the form of buckets and objects. Unlike the conventional multi-layer directory structure of file systems, all objects in a bucket are stored at the same logical layer.</li> </ul> <p><b>NOTE</b> Parallel file systems are optimized OBS objects. You are advised to use <b>parallel file systems</b> instead of object buckets to mount OBS volumes to containers.</p>
Storage Class	This parameter is displayed when you select <b>Object bucket</b> for <b>Instance Type</b> .  This parameter indicates the storage classes supported by OBS. <ul style="list-style-type: none"> <li>• <b>Standard:</b> applicable to scenarios where a large number of hotspot files or small-sized files need to be accessed frequently (multiple times per month on average) and require fast access response.</li> <li>• <b>Infrequent access:</b> applicable to scenarios where data is not frequently accessed (less than 12 times per year on average) but requires fast access response.</li> </ul>

Parameter	Parameter Description
Storage Policy	Object storage has the following policies: <b>Private:</b> Only the bucket owner has full control over the bucket. Unauthorized users do not have permissions to access the bucket.
Access Mode	Access permissions of user applications on storage resources (PVs). <ul style="list-style-type: none"> <li>• <b>ReadWriteMany (RWX):</b> The volume is mounted as read-write by multiple nodes.</li> </ul>
Storage Format	The default type is <b>CSI</b> . The container storage interface (CSI) is used to establish a set of standard storage management interfaces between Kubernetes and external storage systems to provide storage services for containers.

**Step 4** Click **Create**.

After the OBS volume is successfully created, it is displayed in the OBS volume list. Click the PVC name to view detailed information about the OBS volume.

----End

## Adding an OBS Volume

**Step 1** Create a workload or job by referring to [Creating a Deployment](#), [Creating a StatefulSet](#), [Creating a DaemonSet](#), or [Creating a Job](#). After you have added a container, choose **Data Storage > Cloud Volume**, and then click **Add Cloud Volume**.

**Step 2** Set **Type** to **OBS**.

**Table 17-110** OBS volume parameters

Parameter	Description
<b>Type</b>	Select <b>OBS</b> . <b>OBS:</b> Standard and Infrequent Access OBS buckets are supported. OBS buckets are commonly used for big data analytics, cloud native applications, static website hosting, and backup/active archiving.
<b>Allocation Mode</b>	
Manual	<b>Name:</b> Select a created OBS volume. <b>Sub-Type:</b> class of the selected volume. The value can be <b>Standard</b> or <b>Infrequent access</b> , and you do not need to set this parameter.

Parameter	Description
Automatic	<p>Type of the storage instance created on OBS.</p> <ul style="list-style-type: none"> <li>• <b>Parallel file system:</b> If the cluster version is v1.15 or later and the everest add-on version is 1.0.2 or later, parallel file systems that can be mounted by obsfs can be created. <b>Storage Format:</b> The default value is <b>CSI</b>.</li> <li>• <b>Object bucket:</b> A bucket is a container for storing objects in OBS. <b>Sub-Type:</b> Select <b>Standard</b> or <b>Infrequent access</b>. <b>Storage Format:</b> The default value is <b>CSI</b>.</li> </ul> <p><b>NOTE</b> Parallel file systems are optimized OBS objects. You are advised to <b>use parallel file systems</b> instead of object buckets to mount OBS volumes to containers.</p>
Add Container Path	<p>Configure the following parameters:</p> <ol style="list-style-type: none"> <li>1. <b>Container Path:</b> Enter the mount path in the container, for example, <b>/tmp</b>. The mount path must not be a system directory, such as <b>/</b> and <b>/var/run</b>. Otherwise, an exception occurs. You are advised to mount the volume to an empty directory. If the directory is not empty, ensure that there are no files affecting container startup in the directory. Otherwise, such files will be replaced, resulting in failures to start the container and create the workload. <b>NOTICE</b> If the volume is mounted to a high-risk directory, you are advised to use an account with minimum permissions to start the container; otherwise, high-risk files on the host machine may be damaged.</li> <li>2. Set permissions. <ul style="list-style-type: none"> <li>- <b>Read-only:</b> You can only read the data in the mounted volumes.</li> <li>- <b>Read/Write:</b> You can modify the data in the mounted volumes. Newly written data is not migrated if the container is migrated, which causes a data loss.</li> </ul> </li> </ol> <p>Click <b>Add Container Path</b> to add multiple settings. Then, click <b>OK</b>.</p>

**Step 3** Click **OK**.

----End

## Importing an OBS Volume

CCE allows you to import existing OBS volumes.

**Step 1** Log in to the CCE console. In the navigation pane, choose **Resource Management** > **Storage**. On the **OBS** tab page, click **Import**.

**Step 2** Select one or more OBS volumes that you want to import.



 **NOTE**

Parallel file systems are optimized OBS objects. You are advised to **use parallel file systems** instead of object buckets to mount OBS volumes to containers.

**Step 3** Select the target cluster and namespace.

**Step 4** Click **OK**.

----End

## Unbinding an OBS Volume

When an OBS volume is successfully created, the OBS volume is automatically bound to the current cluster. Other clusters can also use the OBS volume. When the volume is unbound from the cluster, other clusters can still use the volume.

If the volume has been mounted to a workload, the volume cannot be unbound from the cluster.

**Step 1** Log in to the CCE console. In the navigation pane, choose **Resource Management > Storage**. In the OBS volume list, click **Unbind** next to the target OBS volume.

**Step 2** In the dialog box displayed, click **Yes**.

----End

## Related Operations

After an OBS volume is created, you can perform the operation described in [Table 17-111](#).

**Table 17-111** Other Operations

Operation	Description
Deleting an OBS volume	<ol style="list-style-type: none"> <li>1. Select the OBS volume to be deleted and click <b>Delete</b> in the <b>Operation</b> column.</li> <li>2. Follow the prompts to delete the volume.</li> </ol>

### 17.9.9.3.2 Creating a Deployment Mounted with an OBS Volume

#### Scenario

After an OBS volume is created or imported to CCE, you can mount the volume to a workload.

#### Prerequisites

You have created a CCE cluster and installed the CSI plug-in ([everest](#)) in the cluster.

## Notes and Constraints

The following configuration example applies to clusters of Kubernetes 1.15 or later.

## Procedure

- Step 1** Use `kubectl` to connect to the cluster. For details, see [Connecting to a Cluster Using kubectl](#).
- Step 2** Run the following commands to configure the `obs-deployment-example.yaml` file, which is used to create a pod.

```
touch obs-deployment-example.yaml
```

```
vi obs-deployment-example.yaml
```

Example of mounting an OBS volume to a Deployment (PVC-based, shared volume):

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: obs-deployment-example           # Workload name
  namespace: default
spec:
  replicas: 1
  selector:
    matchLabels:
      app: obs-deployment-example
  template:
    metadata:
      labels:
        app: obs-deployment-example
    spec:
      containers:
        - image: nginx
          name: container-0
          volumeMounts:
            - mountPath: /tmp           # Mount path
              name: pvc-obs-example
          restartPolicy: Always
          imagePullSecrets:
            - name: default-secret
          volumes:
            - name: pvc-obs-example
              persistentVolumeClaim:
                claimName: pvc-obs-auto-example # PVC name
```

### NOTE

`spec.template.spec.containers.volumeMounts.name` and `spec.template.spec.volumes.name` must be consistent because they have a mapping relationship.

- Step 3** Run the following command to create the workload:

```
kubectl create -f obs-deployment-example.yaml
```

```
----End
```

### 17.9.9.3.3 Creating a StatefulSet Mounted with an OBS Volume

#### Scenario

CCE allows you to use an existing OBS volume to create a StatefulSet through a PVC.

#### Prerequisites

You have created a CCE cluster and installed the CSI plug-in ([everest](#)) in the cluster.

#### Notes and Constraints

The following configuration example applies to clusters of Kubernetes 1.15 or later.

#### Procedure

- Step 1** Create an OBS volume by referring to [PersistentVolumeClaims \(PVCs\)](#) and obtain the PVC name.
- Step 2** Use `kubectl` to connect to the cluster. For details, see [Connecting to a Cluster Using kubectl](#).
- Step 3** Create a YAML file for creating the workload. Assume that the file name is **obs-statefulset-example.yaml**.

**touch obs-statefulset-example.yaml**

**vi obs-statefulset-example.yaml**

Configuration example:

```
apiVersion: apps/v1
kind: StatefulSet
metadata:
  name: obs-statefulset-example
  namespace: default
spec:
  replicas: 1
  selector:
    matchLabels:
      app: obs-statefulset-example
  template:
    metadata:
      labels:
        app: obs-statefulset-example
    spec:
      volumes:
        - name: pvc-obs-example
          persistentVolumeClaim:
            claimName: pvc-obs-example
      containers:
        - name: container-0
          image: 'nginx:latest'
          volumeMounts:
            - name: pvc-obs-example
              mountPath: /tmp
          restartPolicy: Always
          imagePullSecrets:
```

```
- name: default-secret
serviceName: obs-statefulset-example-headless # Name of the headless Service
```

**Table 17-112** Key parameters

Parameter	Description
replicas	Number of pods.
name	Name of the new workload.
image	Image used by the workload.
mountPath	Mount path of a container.
serviceName	Service corresponding to the workload. For details about how to create a Service, see <a href="#">Creating a StatefulSet</a> .
claimName	Name of an existing PVC.

Example of mounting an OBS volume to a StatefulSet (PVC template-based, dedicated volume):

**Example YAML:**

```
apiVersion: apps/v1
kind: StatefulSet
metadata:
  name: obs-statefulset-example
  namespace: default
spec:
  replicas: 1
  selector:
    matchLabels:
      app: obs-statefulset-example
  template:
    metadata:
      labels:
        app: obs-statefulset-example
    spec:
      containers:
        - name: container-0
          image: 'nginx:latest'
          volumeMounts:
            - name: pvc-obs-auto-example
              mountPath: /tmp
          restartPolicy: Always
          imagePullSecrets:
            - name: default-secret
      volumeClaimTemplates:
        - metadata:
            name: pvc-obs-auto-example
            namespace: default
            annotations:
              everest.io/obs-volume-type: STANDARD
          spec:
            accessModes:
              - ReadWriteMany
            resources:
              requests:
                storage: 1Gi
            storageClassName: csi-obs
      serviceName: obs-statefulset-example-headless
```

**Step 4** Create a StatefulSet.

```
kubectl create -f obs-statefulset-example.yaml
```

----End

## Verifying Persistent Storage of an OBS Volume

**Step 1** Query the pod and OBS volume of the deployed workload (for example, **obs-statefulset-example**).

1. Run the following command to query the pod name of the workload:

```
kubectl get po | grep obs-statefulset-example
```

Expected outputs:

```
obs-statefulset-example-0 1/1 Running 0 2m5s
```

2. Run the following command to check whether an OBS volume is mounted to the **/tmp** directory:

```
kubectl exec obs-statefulset-example-0 -- mount|grep /tmp
```

Expected outputs:

```
s3fs on /tmp type fuse.s3fs (rw,nosuid,nodev,relatime,user_id=0,group_id=0,allow_other)
```

**Step 2** Run the following command to create a file named **test** in the **/tmp** directory:

```
kubectl exec obs-statefulset-example-0 -- touch /tmp/test
```

**Step 3** Run the following command to view the file in the **/tmp** directory:

```
kubectl exec obs-statefulset-example-0 -- ls -l /tmp
```

Expected outputs:

```
-rw-r--r-- 1 root root 0 Jun 1 02:50 test
```

**Step 4** Run the following command to delete the pod named **obs-statefulset-example-0**:

```
kubectl delete po obs-statefulset-example-0
```

**Step 5** Check whether the file still exists after the pod is rebuilt.

1. Run the following command to query the name of the rebuilt pod:

```
kubectl get po
```

Expected outputs:

```
obs-statefulset-example-0 1/1 Running 0 2m
```

2. Run the following command to view the file in the **/tmp** directory:

```
kubectl exec obs-statefulset-example-0 -- ls -l /tmp
```

Expected outputs:

```
-rw-r--r-- 1 root root 0 Jun 1 02:50 test
```

3. The **test** file still exists after the pod is rebuilt, indicating that the data in the OBS volume can be persistently stored.

----End

### 17.9.9.4 SFS Volumes

### 17.9.9.4.1 Using SFS Volumes

#### Prerequisites

You have created a CCE cluster and installed the CSI plug-in ([everest](#)) in the cluster.

#### Notes and Constraints

- Container storage in CCE clusters of Kubernetes 1.13 or later version supports encryption. Currently, E2E encryption is supported only in certain regions.
- Volumes cannot be created in specified enterprise projects. Only the default enterprise project is supported.

#### Creating an SFS Volume

- Step 1** Log in to the CCE console. In the navigation pane, choose **Resource Management > Storage**.
- Step 2** On the **SFS** tab, click **Create SFS File System**.
- Step 3** Configure basic information, as shown in [Table 17-113](#).

**Table 17-113** Parameters for creating an SFS volume

Parameter	Parameter Description
* PVC Name	Name of the new PVC, which is different from the volume name. The actual volume name is automatically generated when the PV is created by the PVC.
Cluster Name	Cluster to which the file system volume belongs.
Namespace	Namespace in which the volume is created.
Total Capacity	The total capacity is the capacity of a single volume. Fees are charged by actual usage.
Access Mode	Access permissions of user applications on storage resources (PVs). <ul style="list-style-type: none"> <li>• <b>ReadWriteMany (RWX)</b>: The SFS volume can be mounted as read-write by multiple nodes.</li> </ul>
Storage Format	The default value is <b>CSI</b> and cannot be changed.

- Step 4** Click **Create**.

The volume is displayed in the list. When **PVS Status** becomes **Bound**, the volume is created successfully.

- Step 5** Click the volume name to view detailed information about the volume.

----End

## Adding an SFS Volume

- Step 1** Create a workload or job by referring to [Creating a Deployment](#), [Creating a StatefulSet](#), [Creating a DaemonSet](#), or [Creating a Job](#). During creation, expand **Data Storage** after adding a container. On the **Cloud Volume** tab page, click **Add Cloud Volume**.
- Step 2** Set the storage class to **SFS**.

**Table 17-114** Parameters for mounting an SFS volume

Parameter	Parameter Description
<b>Type</b>	<b>File Storage (NFS):</b> This type applies to a wide range of scenarios, including media processing, content management, big data, and application analysis.
<b>Allocation Mode</b>	
Manual	<ul style="list-style-type: none"> <li>• <b>Name:</b> Select a created file system. You need to create a file system in advance. For details about how to create a file system, see <a href="#">Creating an SFS Volume</a>.</li> <li>• <b>Sub-Type:</b> subtype of the created file storage.</li> <li>• <b>Storage Capacity:</b> This field is one of the PVC attributes. If the storage capacity has been expanded on the IaaS side, it is normal that the capacity values are inconsistent. The PVC capacity is the same as the storage entity capacity only after end-to-end container storage capacity expansion is supported for CCE clusters of v1.13.</li> </ul>
Automatic	<p>An SFS volume is created automatically. You need to enter the storage capacity.</p> <ul style="list-style-type: none"> <li>• <b>Sub-Type:</b> Select <b>NFS</b>.</li> <li>• <b>Storage Capacity:</b> Specify the total storage capacity, in GB. Ensure that the storage capacity quota is not exceeded; otherwise, creation will fail.</li> <li>• <b>Storage Format:</b> The default value is <b>CSI</b>. The container storage interface (CSI) is used to establish a set of standard storage management interfaces between Kubernetes and external storage systems to provide storage services for containers.</li> </ul>

Parameter	Parameter Description
Add Container Path	<p>Configure the following parameters:</p> <ol style="list-style-type: none"> <li>1. <b>subPath</b>: Enter the subpath of the file storage, for example, <b>/tmp</b>. If this parameter is not specified, the root path of the data volume is used by default. Currently, only file storage is supported. The value must be a relative path and cannot start with a slash (/) or ../.</li> <li>2. <b>Container Path</b>: Enter the path of the container, for example, <b>/tmp</b>. The container path must not be a system directory, such as <b>/</b> and <b>/var/run</b>. Otherwise, an exception occurs. You are advised to mount the volume to an empty directory. If the directory is not empty, ensure that there are no files affecting container startup in the directory. Otherwise, such files will be replaced, resulting in failures to start the container and create the workload. <b>NOTICE</b> If the volume is mounted to a high-risk directory, you are advised to use an account with minimum permissions to start the container; otherwise, high-risk files on the host machine may be damaged.</li> <li>3. Set permissions. <ul style="list-style-type: none"> <li>- <b>Read-only</b>: You can only read the data volumes mounted to the path.</li> <li>- <b>Read/Write</b>: You can modify the data volumes mounted to the path. Newly written data is not migrated if the container is migrated, which may cause a data loss.</li> </ul> </li> </ol> <p>Click <b>Add Container Path</b> to add multiple settings. Then, click <b>OK</b>.</p>

**Step 3** Click **OK**.

----End

## Importing an SFS Volume

CCE allows you to import existing SFS volumes.

**Step 1** Log in to the CCE console. In the navigation pane, choose **Resource Management > Storage**. On the **SFS** tab page, click **Import**.

**Step 2** Select one or more SFS volumes that you want to attach.

**Step 3** Select the target cluster and namespace. Then, click **OK**.

----End

## Unbinding an SFS Volume

When an SFS volume is successfully created or imported, the volume is automatically bound to the current cluster. Other clusters can also use the volume.



When the SFS volume is unbound from the cluster, other clusters can still import and use the volume.

If the SFS volume has been attached to a workload, the volume cannot be unbound from the cluster.

**Step 1** Log in to the CCE console. In the navigation pane, choose **Resource Management > Storage**. In the SFS volume list, click **Unbind** next to the target volume.

**Step 2** Confirm the unbinding, and click **OK**.

----End

## Related Operations

After an SFS volume is created, you can perform the operation described in [Table 17-115](#).

**Table 17-115** Other operations

Operation	Description
Deleting an SFS volume	<ol style="list-style-type: none"> <li>1. Select the SFS volume to be deleted and click <b>Delete</b> in the <b>Operation</b> column.</li> <li>2. Follow the prompts to delete the EVS disk.</li> </ol>
Importing an SFS volume	<p>CCE allows you to import existing SFS volumes.</p> <ol style="list-style-type: none"> <li>1. On the <b>SFS</b> tab page, click <b>Import</b>.</li> <li>2. Select one or more SFS volumes that you want to attach.</li> <li>3. Select the target cluster and namespace.</li> <li>4. Click <b>Yes</b>.</li> </ol>

### 17.9.9.4.2 Creating a Deployment Mounted with an SFS Volume

#### Scenario

After an SFS volume is created or imported to CCE, you can mount the volume to a workload.

#### Prerequisites

You have created a CCE cluster and installed the CSI plug-in ([everest](#)) in the cluster.

#### Notes and Constraints

The following configuration example applies to clusters of Kubernetes 1.15 or later.

## Procedure

- Step 1** Use `kubectl` to connect to the cluster. For details, see [Connecting to a Cluster Using kubectl](#).
- Step 2** Run the following commands to configure the `sfs-deployment-example.yaml` file, which is used to create a pod.

```
touch sfs-deployment-example.yaml
```

```
vi sfs-deployment-example.yaml
```

Example of mounting an SFS volume to a Deployment (PVC-based, shared volume):

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: sfs-deployment-example           # Workload name
  namespace: default
spec:
  replicas: 1
  selector:
    matchLabels:
      app: sfs-deployment-example
  template:
    metadata:
      labels:
        app: sfs-deployment-example
    spec:
      containers:
        - image: nginx
          name: container-0
          volumeMounts:
            - mountPath: /tmp           # Mount path
              name: pvc-sfs-example
      imagePullSecrets:
        - name: default-secret
      restartPolicy: Always
      volumes:
        - name: pvc-sfs-example
          persistentVolumeClaim:
            claimName: pvc-sfs-auto-example   # PVC name
```

### NOTE

`spec.template.spec.containers.volumeMounts.name` and `spec.template.spec.volumes.name` must be consistent because they have a mapping relationship.

- Step 3** Run the following command to create the workload:

```
kubectl create -f sfs-deployment-example.yaml
```

```
----End
```

### 17.9.9.4.3 Creating a StatefulSet Mounted with an SFS Volume

## Scenario

CCE allows you to use an existing SGS volume to create a StatefulSet (by using a PVC).

## Prerequisites

You have created a CCE cluster and installed the CSI plug-in ([everest](#)) in the cluster.

## Notes and Constraints

The following configuration example applies to clusters of Kubernetes 1.15 or later.

## Procedure

- Step 1** Create an SFS volume by referring to [PersistentVolumeClaims \(PVCs\)](#) and record the volume name.
- Step 2** Use `kubectl` to connect to the cluster. For details, see [Connecting to a Cluster Using kubectl](#).
- Step 3** Create a YAML file for creating the workload. Assume that the file name is `sfs-statefulset-example.yaml`.

**touch sfs-statefulset-example.yaml**

**vi sfs-statefulset-example.yaml**

Configuration example:

```
apiVersion: apps/v1
kind: StatefulSet
metadata:
  name: sfs-statefulset-example
  namespace: default
spec:
  replicas: 1
  selector:
    matchLabels:
      app: sfs-statefulset-example
  template:
    metadata:
      labels:
        app: sfs-statefulset-example
    spec:
      volumes:
      - name: pvc-sfs-example
        persistentVolumeClaim:
          claimName: pvc-sfs-example
      containers:
      - name: container-0
        image: 'nginx:latest'
        volumeMounts:
        - name: pvc-sfs-example
          mountPath: /tmp
        restartPolicy: Always
        imagePullSecrets:
        - name: default-secret
      serviceName: sfs-statefulset-example-headless
    updateStrategy:
      type: RollingUpdate
```

**Table 17-116** Key parameters

Parent Parameter	Parameter	Description
spec	replicas	Number of pods.
metadata	name	Name of the new workload.
spec.template.spec.containers	image	Image used by the workload.
spec.template.spec.containers.volumeMounts	mountPath	Mount path of a container.
spec	serviceName	Service corresponding to the workload. For details about how to create a Service, see <a href="#">Creating a StatefulSet</a> .
spec.template.spec.volumeClaims.persistentVolumeClaim	claimName	Name of an existing PVC.

Example of mounting an SFS volume to a StatefulSet (PVC template-based, dedicated volume):

**Example YAML file:**

```
apiVersion: apps/v1
kind: StatefulSet
metadata:
  name: sfs-statefulset-example
  namespace: default
spec:
  replicas: 1
  selector:
    matchLabels:
      app: sfs-statefulset-example
  template:
    metadata:
      labels:
        app: sfs-statefulset-example
    spec:
      containers:
        - name: container-0
          image: 'nginx:latest'
          volumeMounts:
            - name: pvc-sfs-auto-example
              mountPath: /tmp
      restartPolicy: Always
      imagePullSecrets:
        - name: default-secret
      volumeClaimTemplates:
        - metadata:
            name: pvc-sfs-auto-example
            namespace: default
          spec:
            accessModes:
              - ReadWriteMany
            resources:
              requests:
                storage: 10Gi
            storageClassName: csi-nas
      serviceName: sfs-statefulset-example-headless
```

```
updateStrategy:
  type: RollingUpdate
```

 **NOTE**

`spec.template.spec.containers.volumeMounts.name` and `spec.template.spec.volumes.name` must be consistent because they have a mapping relationship.

**Step 4** Create a StatefulSet.

```
kubectl create -f sfs-statefulset-example.yaml
----End
```

## 17.10 Monitoring and Logs

### 17.10.1 Monitoring Overview

CCE works with AOM to comprehensively monitor clusters. When a node is created, the ICAGENT (the DaemonSet named **icagent** in the kube-system namespace of the cluster) of AOM is installed by default. The ICAGENT collects monitoring data of underlying resources and workloads running on the cluster. It also collects monitoring data of custom metrics of the workload.

- Resource metrics  
Basic resource monitoring includes CPU, memory, and disk monitoring. For details, see [Resource Metrics](#). You can view these metrics of clusters, nodes, and workloads on the CCE or AOM console.
- Custom metrics  
The ICAGENT collects custom metrics of applications and uploads them to AOM. For details, see [Custom Monitoring](#).

In addition, you can install the Prometheus add-on in a cluster and use Prometheus to collect and display monitoring data. For details, see [Monitoring by Using the prometheus Add-on](#).


### Resource Metrics

**Table 17-117** Resource metrics

Metric	Description
CPU Allocation Rate	Indicates the percentage of CPUs allocated to workloads.
Memory Allocation Rate	Indicates the percentage of memory allocated to workloads.
CPU Usage	Indicates the CPU usage.
Memory Usage	Indicates the memory usage.
Disk Usage	Indicates the disk usage.

Metric	Description
Down	Indicates the speed at which data is downloaded to a node. The unit is KB/s.
Up	Indicates the speed at which data is uploaded from a node. The unit is KB/s.
Disk Read Rate	Indicates the data volume read from a disk per second. The unit is KB/s.
Disk Write Rate	Indicates the data volume written to a disk per second. The unit is KB/s.

## Viewing Cluster Monitoring Data

In the navigation pane of the CCE console, choose **Resource Management > Clusters**. Click  on the cluster card to access the cluster monitoring page.



The cluster monitoring page displays the monitoring status of cluster resources, CPU, memory, and disk usage of all nodes in a cluster, and CPU and memory allocation rates.


### Explanation of monitoring metrics:

- CPU allocation rate = Sum of CPU quotas requested by pods in the cluster/Sum of CPU quotas that can be allocated of all nodes (excluding master nodes) in the cluster
- Memory allocation rate = Sum of memory quotas requested by pods in the cluster/Sum of memory quotas that can be allocated of all nodes (excluding master nodes) in the cluster
- CPU usage: Average CPU usage of all nodes (excluding master nodes) in a cluster

- Memory usage: Average memory usage of all nodes (excluding master nodes) in a cluster

**NOTE**

Allocatable node resources (CPU or memory) = Total amount – Reserved amount – Eviction thresholds. For details, see [Formula for Calculating the Reserved Resources of a Node](#).

On the cluster monitoring page, you can also view monitoring data of nodes, workloads, and pods. You can click  to view the detailed data.

The screenshot shows four monitoring panels:

- Node Monitoring:** A table with columns for CPU Usage, Memory Usage, Available Mem., Disk Usage, Available Mem., Down (KB/s), and Up (KB/s). It lists two nodes with their respective resource usage.
- Master Node Monitoring:** A panel indicating that no data is available for master nodes.
- Workload Monitoring:** A table with columns for Type, Namespace, CPU Usage, Memory Us., Disk Read R., Disk Write R., Down (KB/s), and Up (KB/s). It lists several workloads like 'custom-metrics-apiserver' and 'cceeddon-prometheus-operator'.
- Pod Monitoring:** A table with columns for Workload, Namespace, CPU Usage, Memory Us., Disk Read R., Disk Write R., Down (KB/s), and Up (KB/s). It lists pods associated with the workloads above.

## Viewing Monitoring Data of Master Nodes

CCE allows you to view monitoring data of master nodes. You can view the monitoring data of a master node in the upper right corner of the cluster details page. Clicking **More** will direct you to the AOM console.

The screenshot shows the 'View Monitor Graphs' interface for a master node. It includes a search bar, a 'Host Template' section, and three line graphs:

- cpuUsage (Average):** Shows CPU usage over time for Host IP: 192.168.0.107.
- memUsedRate (Average):** Shows Physical memory usage over time for Host IP: 192.168.0.107.
- diskUsedRate (Average):** Shows Disk usage over time for Host IP: 192.168.0.107.

## Viewing Monitoring Data of Worker Nodes

In addition to the cluster monitoring page, you can also view node monitoring data on the node console by clicking **Monitoring** in the row where the node resides.

Node Name	Status	Node Pool	Specifications	Allocatable	IP	AZ	Billing Mode	Node Type	Operation
prometheus-38892-wsb84	Available Schedulable	DefaultPool	4 cores   8 GB   c6s.xlarge...	CPU: 1.85 Core Memory: 2.31 GiB	192.168.0.71 (Private)	AZ1	Pay-per-use Nov 05, 2021 10:35...	Node created	Monitoring More
prometheus-38892-cy6dg	Available Schedulable	DefaultPool	4 cores   8 GB   c6s.xlarge...	CPU: 2.15 Core Memory: 4.08 GiB	192.168.0.37 (Private)	AZ1	Pay-per-use Nov 05, 2021 10:35...	Node created	Monitoring More

The node list page also displays the data about the allocable resources of the node. **Allocatable resources** indicate the upper limit of resources that can be requested by pods on a node, and are calculated based on the requests. Allocatable resources do not indicate the actual available resources of the node.

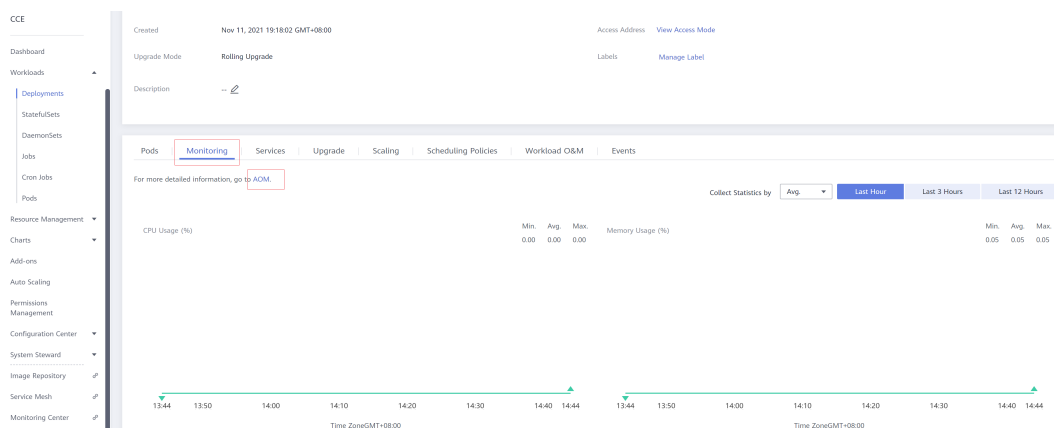
The calculation formulas are as follows:

- Allocatable CPU = Total CPU – Requested CPU of all pods – Reserved CPU for other resources
- Allocatable memory = Total memory – Requested memory of all pods – Reserved memory for other resources

Node Name	Status	Node Pool	Specifications	Allocatable	IP
prometheus-38892-wsb84	Available Schedulable	DefaultPool	4 cores   8 GB   c6s.xlarge...	CPU: 1.85 Core Memory: 2.31 GiB	192.168.0.71 (Private)
prometheus-38892-cy6dg	Available Schedulable	DefaultPool	4 cores   8 GB   c6s.xlarge...	CPU: 2.15 Core Memory: 4.08 GiB	192.168.0.37 (Private)

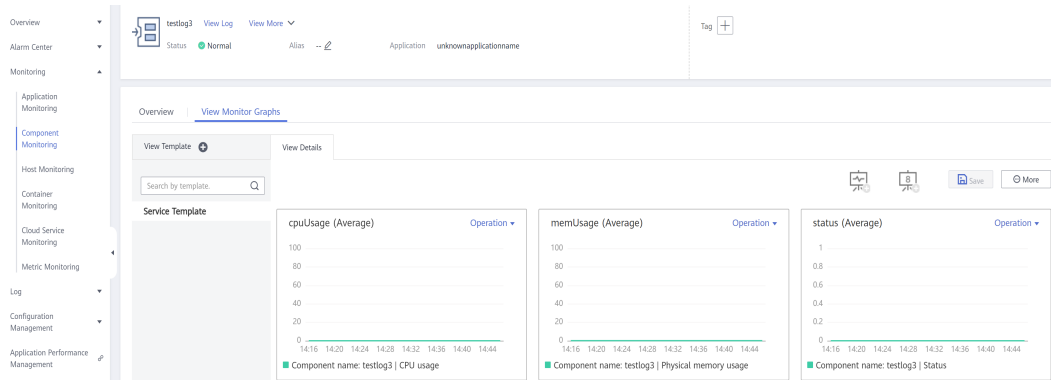
## Viewing Workload Monitoring Data

You can view monitoring data of a workload on the **Monitoring** tab page of the workload details page.



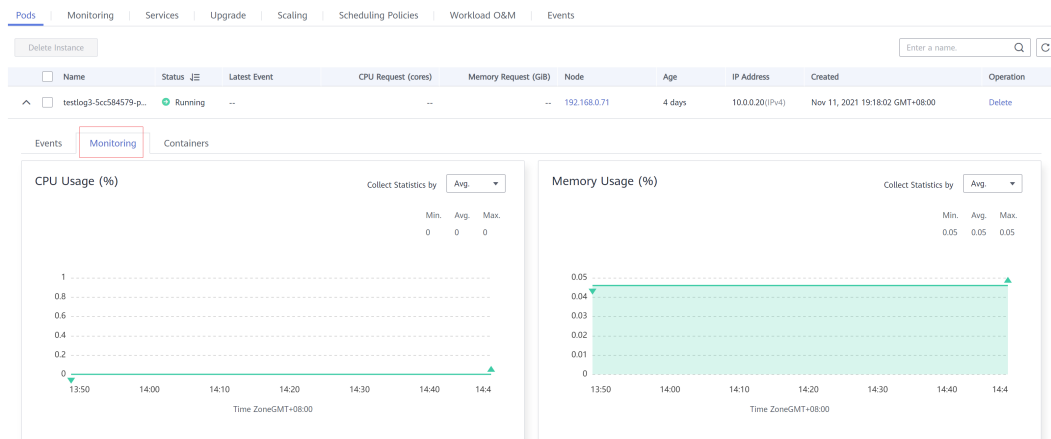
You can also click **AOM** to go to the AOM console to view monitoring data of the workload.





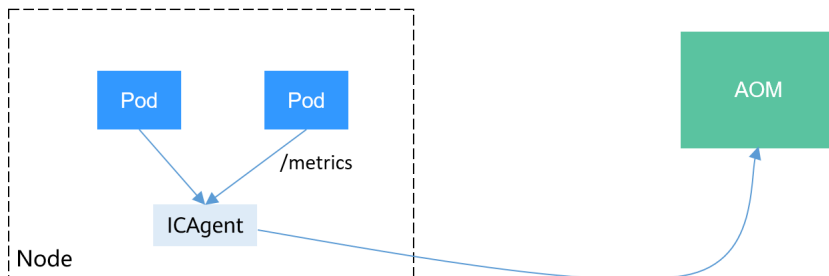
## Viewing Pod Monitoring Data

You can view monitoring data of a pod on the **Pods** tab page of the workload details page.



## 17.10.2 Custom Monitoring

CCE allows you to upload custom metrics to AOM. The ICAgent on a node periodically calls the metric monitoring API configured on a workload to read monitoring data and then uploads the data to AOM.



The custom metric API of a workload can be configured when the workload is created. This section uses an Nginx application as an example to describe how to report custom metrics to AOM.

## Notes and Constraints

- The ICAgent is compatible with the monitoring data specifications of [Prometheus](#). The custom metrics provided by pods can be collected by the ICAgent only when they meet the monitoring data specifications of Prometheus.
- The ICAgent supports only [Gauge](#) metrics.
- The interval for the ICAgent to call the custom metric API is 1 minute, which cannot be changed.

## Prometheus Monitoring Data Collection

Prometheus periodically calls the metric monitoring API (`/metrics` by default) of an application to obtain monitoring data. The application needs to provide the metric monitoring API for Prometheus to call, and the monitoring data must meet the following specifications of Prometheus:

```
# TYPE nginx_connections_active gauge
nginx_connections_active 2
# TYPE nginx_connections_reading gauge
nginx_connections_reading 0
```

Prometheus provides clients in various languages. For details about the clients, see [Prometheus CLIENT LIBRARIES](#). For details about how to develop an exporter, see [WRITING EXPORTERS](#). The Prometheus community provides various third-party exporters that can be directly used. For details, see [EXPORTERS AND INTEGRATIONS](#).

## Preparing an Application

Nginx has a module named `ngx_http_stub_status_module`, which provides basic monitoring functions. You can configure the `nginx.conf` file to provide an API for external systems to access Nginx monitoring data. As shown in the following figure, after the server configuration is added to `http`, Nginx can provide an API for external systems to access Nginx monitoring data.

```
user nginx;
worker_processes auto;

error_log /var/log/nginx/error.log warn;
pid /var/run/nginx.pid;

events {
    worker_connections 1024;
}

http {
    include /etc/nginx/mime.types;
    default_type application/octet-stream;
    log_format main '$remote_addr - $remote_user [$time_local] "$request" '
        '$status $body_bytes_sent "$http_referer" '
        '"$http_user_agent" "$http_x_forwarded_for"';

    access_log /var/log/nginx/access.log main;
    sendfile on;
    #tcp_nopush on;
    keepalive_timeout 65;
    #gzip on;
    include /etc/nginx/conf.d/*.conf;

    server {
```

```
listen 8080;
server_name localhost;
location /stub_status {
    stub_status on;
    access_log off;
}
}
```

Save the preceding configuration to the **nginx.conf** file and use the configuration to create a new image. The Dockerfile file is as follows:

```
FROM nginx:1.27.1-alpine
ADD nginx.conf /etc/nginx/nginx.conf
EXPOSE 80
CMD ["nginx", "-g", "daemon off;"]
```

Use the preceding Dockerfile file to build an image and upload it to SWR. The image name is **nginx:exporter**.

**docker build -t nginx:exporter .**

**docker tag nginx:exporter {swr-address}/{group}/nginx:exporter**

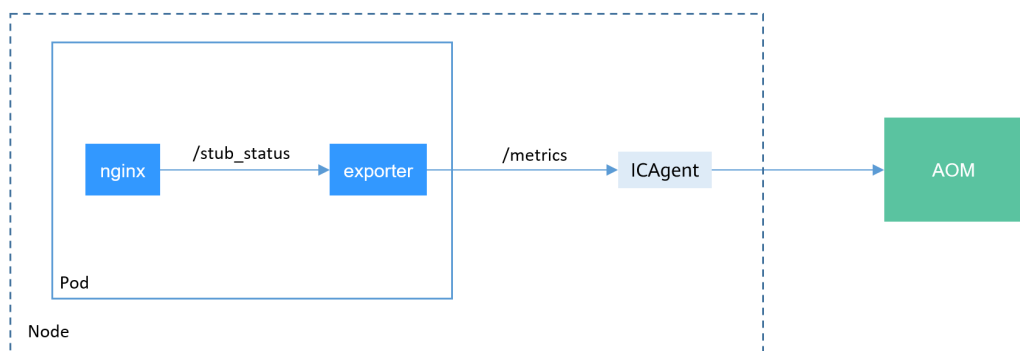
**docker push {swr-address}/{group}/nginx:exporter**

After running a container with image **nginx:exporter**, you can obtain Nginx monitoring data by calling `http://<ip_address>:8080/stub_status`. `<ip_address >` indicates the IP address of the container. The monitoring data is as follows:

```
# curl http://127.0.0.1:8080/stub_status
Active connections: 3
server accepts handled requests
146269 146269 212
Reading: 0 Writing: 1 Waiting: 2
```

## Deploying an Application

The data format of the monitoring data provided by **nginx:exporter** does not meet the requirements of Prometheus. You need to convert the data format to the format required by Prometheus. To convert the format of Nginx metrics, use **nginx-prometheus-exporter**, as shown in the following figure.



Deploy **nginx:exporter** and **nginx-prometheus-exporter** in the same pod.

```
kind: Deployment
apiVersion: apps/v1
metadata:
  name: nginx-exporter
  namespace: default
spec:
```

```

replicas: 1
selector:
  matchLabels:
    app: nginx-exporter
template:
  metadata:
    labels:
      app: nginx-exporter
    annotations:
      metrics.alpha.kubernetes.io/custom-endpoints: '[{"api":"prometheus","path":"/
metrics","port":"9113","names":""}]'
spec:
  containers:
    - name: container-0

      image: '{swr-address}/{group}/nginx:exporter'
      resources:
        limits:
          cpu: 250m
          memory: 512Mi
        requests:
          cpu: 250m
          memory: 512Mi
    - name: container-1
      image: 'nginx/nginx-prometheus-exporter:0.9.0'
      command:
        - nginx-prometheus-exporter
      args:
        - '-nginx.scrape-uri=http://127.0.0.1:8080/stub_status'
  imagePullSecrets:
    - name: default-secret

```

 **NOTE**

The nginx/nginx-prometheus-exporter:0.9.0 image needs to be pulled from the public network. Therefore, each node in the cluster must have a public IP address.

nginx-prometheus-exporter requires a startup command. **nginx-prometheus-exporter -nginx.scrape-uri=http://127.0.0.1:8080/stub\_status** is used to obtain Nginx monitoring data.

In addition, you need to add an annotation **metrics.alpha.kubernetes.io/custom-endpoints: '[{"api":"prometheus","path":"/metrics","port":"9113","names":""}]'** to the pod. The annotation works the same as that you configure on the CCE console. For details about the parameters, see [Table 17-118](#).

Custom Monitoring [You can specify the system to which monitored metrics are reported](#) [Learn how to configure custom monitoring](#)

Report Path	<input type="text" value="/metrics"/>
Report Port	<input type="text" value="9113"/>
Monitoring Metrics	<input \"mem_usage\"]."="" cpu_usage\",="" type="text" value="Enter 5 to 100 characters. Only letters, digits, and underscores (_) are allowed. Enclose metrics in [], for example, [\"/>

**Table 17-118** Parameter description

Parameter	Description	Mandatory
Report Path	URL provided by the exporter for CCE to obtain custom metric data. The path consists of letters, digits, slashes (/), and underscores (_), and must start with a slash (/). For example, <b>/metrics</b> .	Yes
Report Port	Port provided by the exporter for CCE to obtain custom metric data. The value is an integer ranging from 1 to 65535. For example, <b>8080</b> .	Yes
Monitoring Metrics	Name of the custom metric provided by the exporter. The name of a custom metric is a string of 5 to 100 characters. Only letters, digits, and underscores (_) are allowed. The format is as follows: ["Custom metric name 1","Custom metric name 2"]. Use commas (,) to separate multiple custom metric names, for example, <b>["cpu_usage","mem_usage"]</b> . <ul style="list-style-type: none"> <li>• If this parameter is not configured, CCE obtains all custom metric data.</li> <li>• If this parameter is configured, for example, <b>["cpu_usage","mem_usage"]</b>, CCE filters custom metrics and obtains only the data of <code>cpu_usage</code> and <code>mem_usage</code>.</li> </ul>	No

## Verification

After an application is deployed, you can access Nginx to construct some access data and check whether the corresponding monitoring data can be obtained in AOM.

```
$ kubectl get pod
NAME                                READY STATUS RESTARTS AGE
nginx-exporter-78859765db-6j8sw    2/2   Running 0      4m
$ kubectl exec -it nginx-exporter-78859765db-6j8sw -- /bin/sh
Defaulting container name to container-0.
Use 'kubectl describe pod/nginx-exporter-78859765db-6j8sw -n default' to see all of the containers in this pod.
/ # curl http://localhost
<!DOCTYPE html>
<html>
<head>
<title>Welcome to nginx!</title>
<style>
html { color-scheme: light dark; }
body { width: 35em; margin: 0 auto;
font-family: Tahoma, Verdana, Arial, sans-serif; }
</style>
</head>
<body>
<h1>Welcome to nginx!</h1>
```

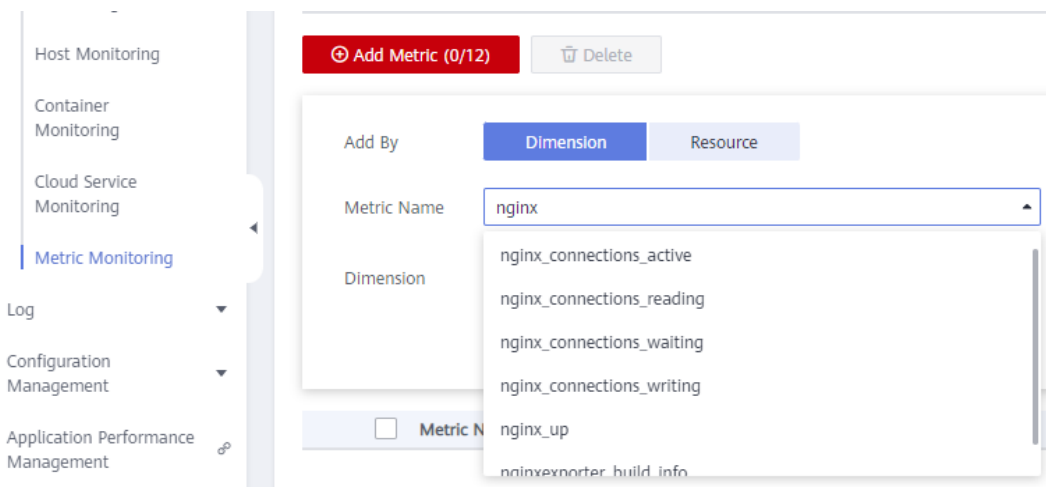
```
<p>If you see this page, the nginx web server is successfully installed and
working. Further configuration is required.</p>

<p>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>.</p>

<p><em>Thank you for using nginx.</em></p>
</body>
</html>
/ #
```

You can see that Nginx has been accessed once.

Log in to AOM. In the navigation pane, choose **Monitoring > Metric Monitoring**. You can view Nginx-related metrics, for example, **nginx\_connections\_active**.



### 17.10.3 Monitoring by Using the prometheus Add-on

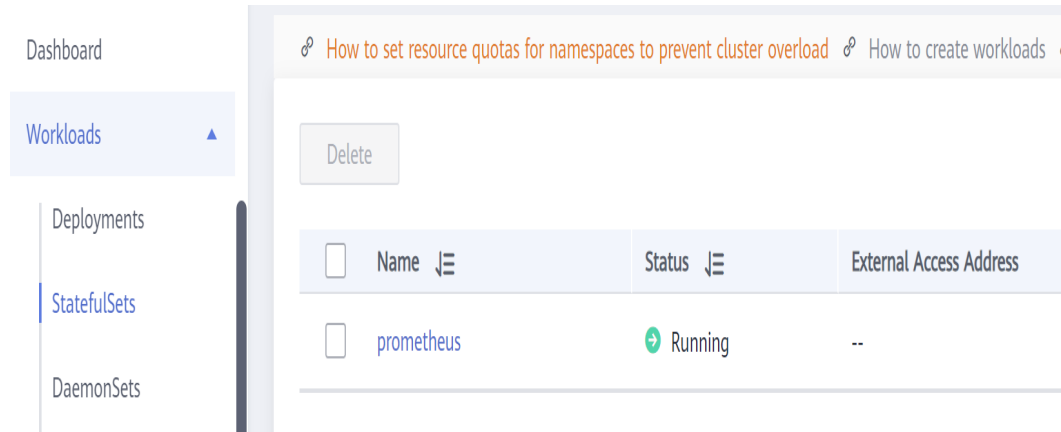
You can use AOM ICAgent to obtain custom metric data of workloads as described in **Custom Monitoring**. You can also install the prometheus add-on in a cluster and use Prometheus as the monitoring platform.

#### Installing the Add-on

For details about how to install the prometheus add-on, see [prometheus](#).

#### Accessing Prometheus

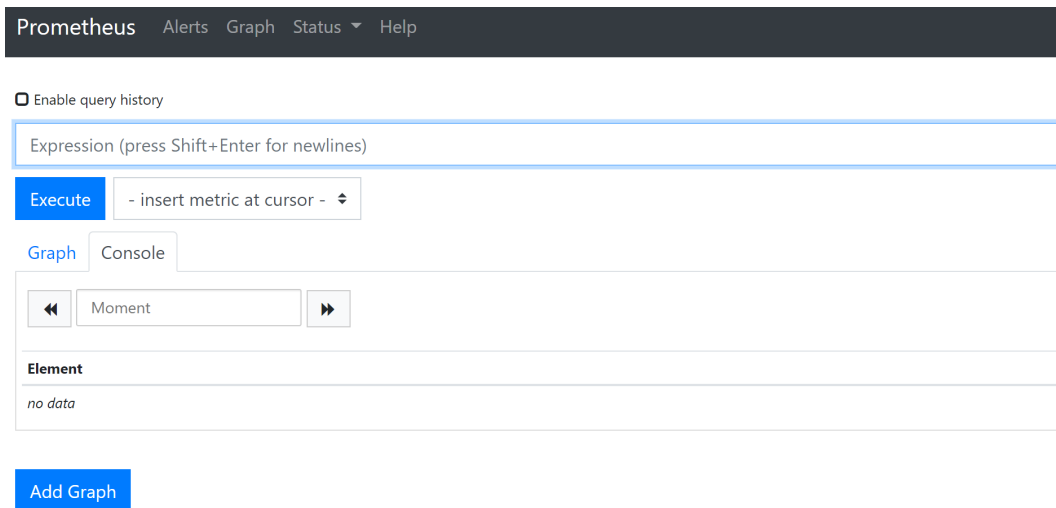
After the prometheus add-on is installed, you can deploy a series of workloads and Services. The Prometheus StatefulSet refers to Prometheus Server.



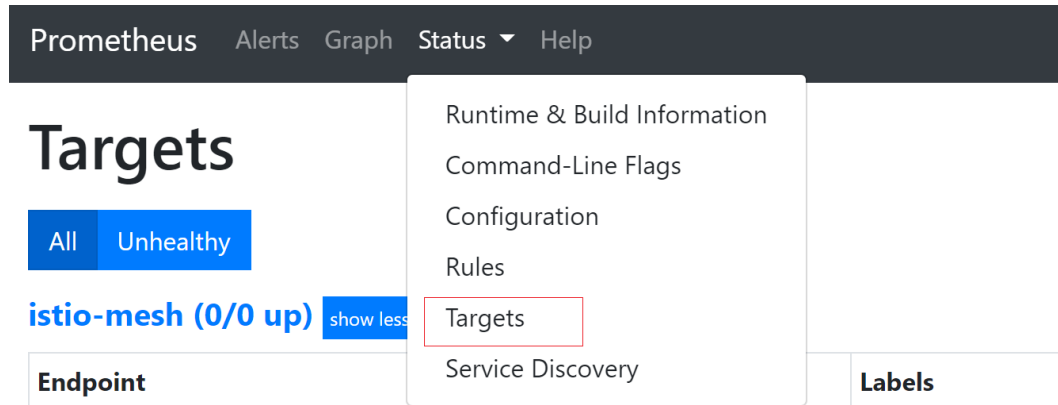
You can create a public network **LoadBalancer Service** so that Prometheus can be accessed from an external network.

Service Name	Internal Domain Name	Workload	Access Address	Access Type	Access Port -> Container Port / Protocol
<input type="checkbox"/> prometheus-access	prometheus-access.monitoring.svc.cluster.local	prometheus	192.168.0.131 (Private)	LoadBalancer (ELB)	9090 -> 9090 / TCP
<input type="checkbox"/> prometheus	prometheus.monitoring.svc.cluster.local	prometheus	10.247.34.200	ClusterIP	80 -> 9090 / TCP
<input type="checkbox"/> grafana	grafana.monitoring.svc.cluster.local	grafana	10.247.197.56	ClusterIP	3000 -> 3000 / TCP
<input type="checkbox"/> custom-metrics-apiserver	custom-metrics-apiserver.monitoring.svc.cluster.local	custom-metrics-apiserver	10.247.213.180	ClusterIP	443 -> 6443 / TCP
<input type="checkbox"/> cceaddon-prometheus-k...	cceaddon-prometheus-kube-state-metrics.monitoring.s...	cceaddon-prometheus-kub...	None	ClusterIP	80 -> 8080 / TCP

After the creation is complete, click the access address to access Prometheus.



Choose **Status > Targets** to view the targets monitored by Prometheus.



## Monitoring Custom Metrics

Custom metrics can also be monitored in Prometheus. The configuration method is simple. For example, for the **nginx:exporter** application in [Custom Monitoring](#), you only need to add the following annotations during deployment. Then Prometheus can automatically collect metrics.

```
kind: Deployment
apiVersion: apps/v1
metadata:
  name: nginx-exporter
  namespace: default
spec:
  replicas: 1
  selector:
    matchLabels:
      app: nginx-exporter
  template:
    metadata:
      labels:
        app: nginx-exporter
      annotations:
        prometheus.io/scrape: "true"
        prometheus.io/port: "9113"
        prometheus.io/path: "/metrics"
        prometheus.io/scheme: "http"
    spec:
      containers:
        - name: container-0

          image: '{swr-address}/{group}/nginx:exporter'
          resources:
            limits:
              cpu: 250m
              memory: 512Mi
            requests:
              cpu: 250m
              memory: 512Mi
        - name: container-1
          image: 'nginx/nginx-prometheus-exporter:0.9.0'
          command:
            - nginx-prometheus-exporter
          args:
            - '-nginx.scrape-uri=http://127.0.0.1:8080/stub_status'
      imagePullSecrets:
        - name: default-secret
```

In the preceding description:



- **prometheus.io/scrape** indicates whether to enable Prometheus to collect pod monitoring data. The value is **true**.
- **prometheus.io/port** indicates the port for collecting monitoring data.
- **prometheus.io/path** indicates the URL of the API for collecting monitoring data. If this parameter is not set, the default value **/metrics** is used.
- **prometheus.io/scheme**: protocol used for data collection. The value can be **http** or **https**.

After the application is deployed, a pod with a collection path of port 9113 can be found under **Status > Targets**.

Endpoint	State	Labels	Last Scrape	Scrape Duration	Error
http://10.0.0.133:8080/metrics	UP	cluster="15d748b4-3de1-11ec-9199-0255ac1000c9" instance="10.0.0.133:8080" job="kubernetes-pods" kubernetes_namespace="monitoring" kubernetes_pod="cceaddon-prometheus-kube-state-metrics-66dcbd49b-2qhmh"	7.047s ago	4.989ms	
http://10.0.0.141:9113/metrics	UP	cluster="15d748b4-3de1-11ec-9199-0255ac1000c9" instance="10.0.0.141:9113" job="kubernetes-pods" kubernetes_namespace="default" kubernetes_pod="nginx-exporter-53cb99f7b-b9qvm"	12.914s ago	6.639ms	
http://10.0.0.7:8080/metrics	UP	cluster="15d748b4-3de1-11ec-9199-0255ac1000c9" instance="10.0.0.7:8080" job="kubernetes-pods" kubernetes_namespace="monitoring" kubernetes_pod="cceaddon-prometheus-operator-574cc9b1f4-jnc1b"	2.156s ago	1.536ms	
http://10.0.0.8:9090/metrics	UP	cluster="15d748b4-3de1-11ec-9199-0255ac1000c9" instance="10.0.0.8:9090" job="kubernetes-pods" kubernetes_namespace="monitoring" kubernetes_pod="prometheus-0"	5.283s ago	5.402ms	

On the **Graph** tab page, enter **nginx**. The Nginx-related metrics are displayed in Prometheus.

Enable query history

nginx

- nginx\_connections\_accepted
- nginx\_connections\_active
- nginx\_connections\_handled
- nginx\_connections\_reading
- nginx\_connections\_waiting
- nginx\_connections\_writing
- nginx\_http\_requests\_total
- nginx\_up
- nginxexporter\_build\_info
- prometheus\_engine\_queries\_concurrent\_max
- kube\_deployment\_spec\_strategy\_rollingupdate\_max\_surge
- kube\_deployment\_spec\_strategy\_rollingupdate\_max\_unavailable

## Accessing Grafana

The prometheus add-on has **Grafana** (an open-source visualization tool) installed and interconnected with Prometheus. You can create a public network **LoadBalancer Service** so that you can access Grafana from the public network and view Prometheus monitoring data on Grafana.

Service Name	Internal Domain Name	Workload	Access Address	Access Type	Access Port -> Container Port / Protocol
<input type="checkbox"/> grafana-access	grafana-access.monitoring.svc.cluster.local	grafana	192.168.0.131 (Private)	LoadBalancer (ELB)	3000 -> 3000 / TCP

Click the access address to access Grafana and select a proper dashboard to view the aggregated content.



## Grafana Data Persistence

Currently, Grafana data in the prometheus add-on is not persistent. If the Grafana container is restarted, the data will be lost. You can mount cloud storage to the Grafana container to achieve Grafana data persistence.

**Step 1** Use `kubectl` to connect to the cluster where the Grafana cluster resides. For details, see [Connecting to a Cluster Using kubectl](#).

**Step 2** Create the PVC of an EVS disk.

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: grafana-pvc
  namespace: monitoring
  annotations:
    everest.io/disk-volume-type: SSD
  labels:
    failure-domain.beta.kubernetes.io/region: eu-west-101
    failure-domain.beta.kubernetes.io/zone: eu-west-101a
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 10Gi
  storageClassName: csi-disk
```

The EVS disk and the node where Grafana resides must be in the same AZ. Otherwise, the EVS disk cannot be attached.

- **failure-domain.beta.kubernetes.io/region**: region where the EVS disk resides.

- **failure-domain.beta.kubernetes.io/zone**: AZ where the EVS disk resides.
- **storage**: EVS disk size. Set this parameter as required.

You can also create EVS disks on the CCE console. For details, see [Using a Storage Class to Create a PVC](#).

**Step 3** Modify the Grafana workload configuration and mount the EVS disk.

#### **kubectl edit deploy grafana -n monitoring**

Add the EVS disk to the container in the YAML file, as shown in the following figure. The PVC name must be the same as that in [Step 2](#), and the mount path must be **/var/lib/grafana**.

In addition, the upgrade policy must be modified for the Grafana workload. The maximum number of pods is 1.

```
...
template:
  spec:
    volumes:
      - name: cce-pvc-grafana
        persistentVolumeClaim:
          claimName: grafana-pvc
    ...
    containers:
      - volumeMounts:
          - name: cce-pvc-grafana
            mountPath: /var/lib/grafana
    ...
  strategy:
    type: RollingUpdate
    rollingUpdate:
      maxUnavailable: 1
      maxSurge: 1
  ...
```

Save the configuration. The Grafana workload will be upgraded and the EVS disk will be mounted.

----End

## 17.10.4 Container Logs

### Scenario

CCE allows you to configure policies for collecting, managing, and analyzing workload logs periodically to prevent logs from being over-sized.

CCE works with AOM to collect workload logs. When a node is created, the ICAgent (the DaemonSet named **icagent** in the kube-system namespace of the cluster) of AOM is installed by default. After the ICAgent collects workload logs and reports them to AOM, you can view workload logs on the CCE or AOM console.

- By default, the ICAgent collects the standard outputs of containers. You do not need to perform any configuration.
- You can also configure the path for storing container logs when creating a workload so that the ICAgent collects logs from this path.

You can select either of the following modes for container logs:

- HostPath: The host path is mounted to the specified container path (mount path). In the node host path, you can view the container logs output into the mount path.
- EmptyDir: The temporary path of the node is mounted to the specified path (mount path). Log data that exists in the temporary path but is not reported by the collector to AOM will disappear after the pod is deleted.

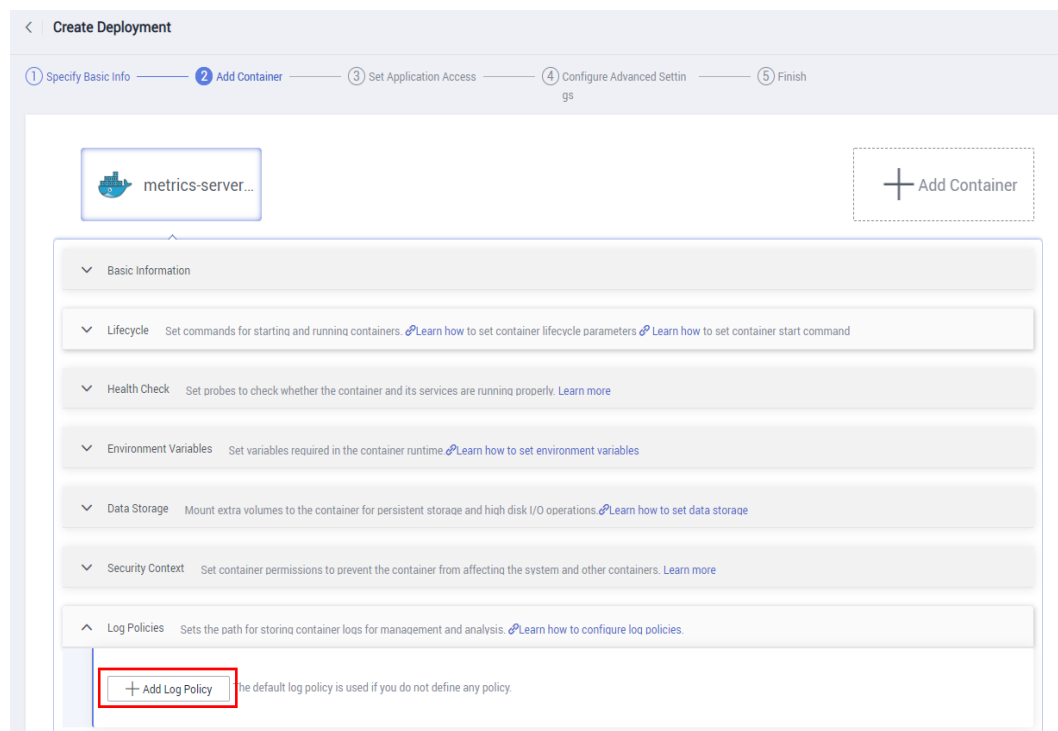
## Precautions

The ICAgent only collects \*.log, \*.trace, and \*.out text log files.

## Setting the Path for Storing Container Logs

**Step 1** When creating a **workload** on the CCE console, add a container and expand **Log Policies**.

**Figure 17-140** Container logs



**Step 2** In the **Log Policies** area, click **Add Log Policy**. Configure parameters in the log policy. The following uses Nginx as an example.

**Figure 17-141** Adding a log policy

**Step 3** Set **Storage Type** to **Host Path** or **Container Path**.

**Table 17-119** Configuring log policies

Parameter	Description
Storage Type	<ul style="list-style-type: none"> <li><b>Host Path:</b> In HostPath mode, the host path is mounted to the specified container path (mount path). In the node host path, you can view the container logs output into the mount path.</li> <li><b>Container Path:</b> In EmptyDir mode, the temporary path of the node is mounted to the specified path (mount path). Log data that exists in the temporary path but is not reported by the collector to AOM will disappear after the pod is deleted.</li> </ul>
<b>Add Container Path</b>	
*Host Path	Enter the host path, for example, <code>/var/paas/sys/log/nginx</code> .

Parameter	Description
Container Path	<p>Container path (for example, <b>/tmp</b>) to which the storage resources will be mounted.</p> <p><b>NOTICE</b></p> <ul style="list-style-type: none"> <li>Do not mount storage to a system directory such as <b>/</b> or <b>/var/run</b>; this action may cause a container error to occur. You are advised to mount the container to an empty directory. If the directory is not empty, ensure that there are no files affecting container startup in the directory. Otherwise, such files will be replaced, resulting in failures to start the container and create the workload.</li> <li>When the container is mounted to a high-risk directory, you are advised to use an account with minimum permissions to start the container; otherwise, high-risk files on the host machine may be damaged.</li> <li>AOM collects only the first 20 log files that have been modified recently. It collects files from 2 levels of subdirectories by default.</li> <li>AOM only collects <b>.log</b>, <b>.trace</b>, and <b>.out</b> text log files in mounting paths.</li> <li>For details about how to set permissions for mount points in a container, see <a href="#">Configure a Security Context for a Pod or Container</a>.</li> </ul>
Extended Host Path	<p>This parameter is mandatory only if <b>Storage Type</b> is set to <b>Host Path</b>.</p> <p>Extended host paths contain pod IDs or container names to distinguish different containers into which the host path is mounted.</p> <p>A level-3 directory is added to the original volume directory/subdirectory. You can easily obtain the files output by a single Pod.</p> <ul style="list-style-type: none"> <li><b>None</b>: No extended path is configured.</li> <li><b>PodUID</b>: ID of a pod.</li> <li><b>PodName</b>: name of a pod.</li> <li><b>PodUID/ContainerName</b>: ID of a pod or name of a container.</li> <li><b>PodName/ContainerName</b>: name of a pod or container.</li> </ul>

Parameter	Description
Collection Path	<p>A collection path narrows down the scope of collection to specified logs.</p> <ul style="list-style-type: none"> <li>If no collection path is specified, log files in <b>.log</b>, <b>.trace</b>, and <b>.out</b> formats will be collected from the specified path.</li> <li><b>/Path/**/</b> indicates that all log files in <b>.log</b>, <b>.trace</b>, and <b>.out</b> formats will be recursively collected from the specified path and all subdirectories at 5 levels deep.</li> <li><b>*</b> in log file names indicates a fuzzy match.</li> </ul> <p>Example: The collection path <b>/tmp/**/test*.log</b> indicates that all <b>.log</b> files prefixed with <b>test</b> will be collected from <b>/tmp</b> and subdirectories at 5 levels deep.</p> <p><b>CAUTION</b> Ensure that the ICAgent version is 5.12.22 or later.</p>
Log Dumping	<p>Log dump refers to rolling log files on a local host.</p> <ul style="list-style-type: none"> <li><b>Enabled:</b> AOM scans log files every minute. When a log file exceeds 50 MB, it is dumped immediately. A new <b>.zip</b> file is generated in the directory where the log file locates. For a log file, AOM stores only the latest 20 <b>.zip</b> files. When the number of <b>.zip</b> files exceeds 20, earlier <b>.zip</b> files will be deleted. After the dump is complete, the log file in AOM will be cleared.</li> <li><b>Disabled:</b> AOM does not dump log files.</li> </ul> <p><b>NOTE</b></p> <ul style="list-style-type: none"> <li>Log file rolling of AOM is implemented in the copytruncate mode. Before enabling log dumping, ensure that log files are written in the append mode. Otherwise, file holes may occur.</li> <li>Currently, mainstream log components such as Log4j and Logback support log file rolling. If your log files already support rolling, skip the configuration. Otherwise, conflicts may occur.</li> <li>You are advised to configure log file rolling for your own services to flexibly control the size and number of rolled files.</li> </ul>

**Step 4** Click **OK**.

----End

## Using kubectl to Set the Container Log Storage Path

You can set the container log storage path by defining a YAML file.

As shown in the following figure, EmptyDir is mounted a temporary path to **/var/log/nginx**. In this way, the ICAgent collects logs in **/var/log/nginx**. The **policy** field is customized by CCE and allows the ICAgent to identify and collect logs.

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: testlog
```

```

namespace: default
spec:
  selector:
    matchLabels:
      app: testlog
  template:
    replicas: 1
    metadata:
      labels:
        app: testlog
    spec:
      containers:
        - image: 'nginx:alpine'
          name: container-0
          resources:
            requests:
              cpu: 250m
              memory: 512Mi
            limits:
              cpu: 250m
              memory: 512Mi
          volumeMounts:
            - name: vol-log
              mountPath: /var/log/nginx
              policy:
                logs:
                  rotate: "
      volumes:
        - emptyDir: {}
          name: vol-log
    imagePullSecrets:
      - name: default-secret

```

The following shows how to use the HostPath mode. Compared with the EmptyDir mode, the type of volume is changed to hostPath, and the path on the host needs to be configured for this hostPath volume. In the following example, **/tmp/log** on the host is mounted to **/var/log/nginx**. In this way, the ICAgent can collect logs in **/var/log/nginx**, without deleting the logs from **/tmp/log**.

```

apiVersion: apps/v1
kind: Deployment
metadata:
  name: testlog
  namespace: default
spec:
  replicas: 1
  selector:
    matchLabels:
      app: testlog
  template:
    metadata:
      labels:
        app: testlog
    spec:
      containers:
        - image: 'nginx:alpine'
          name: container-0
          resources:
            requests:
              cpu: 250m
              memory: 512Mi
            limits:
              cpu: 250m
              memory: 512Mi
          volumeMounts:
            - name: vol-log
              mountPath: /var/log/nginx
              readOnly: false
              extendPathMode: PodUID

```



```

policy:
  logs:
    rotate: Hourly
    annotations:
      pathPattern: '**'

volumes:
- hostPath:
  path: /tmp/log
  name: vol-log
imagePullSecrets:
- name: default-secret
    
```

**Table 17-120** Parameter description

Parameter	Explanation	Description
extendPath Mode	Extended host path	<p>Extended host paths contain pod IDs or container names to distinguish different containers into which the host path is mounted.</p> <p>A level-3 directory is added to the original volume directory/subdirectory. You can easily obtain the files output by a single Pod.</p> <ul style="list-style-type: none"> <li>• <b>None:</b> No extended path is configured.</li> <li>• <b>PodUID:</b> ID of a pod.</li> <li>• <b>PodName:</b> name of a pod.</li> <li>• <b>PodUID/ContainerName:</b> ID of a pod or name of a container.</li> <li>• <b>PodName/ContainerName:</b> name of a pod or container.</li> </ul>
policy.logs.rotate	Log dumping	<p>Log dump refers to rolling log files on a local host.</p> <ul style="list-style-type: none"> <li>• <b>Enabled:</b> AOM scans log files every minute. When a log file exceeds 50 MB, it is dumped immediately. A new <b>.zip</b> file is generated in the directory where the log file locates. For a log file, AOM stores only the latest 20 <b>.zip</b> files. When the number of <b>.zip</b> files exceeds 20, earlier <b>.zip</b> files will be deleted. After the dump is complete, the log file in AOM will be cleared.</li> <li>• <b>Disabled:</b> AOM does not dump log files.</li> </ul> <p><b>NOTE</b></p> <ul style="list-style-type: none"> <li>• Log file rolling of AOM is implemented in the copytruncate mode. Before enabling log dumping, ensure that log files are written in the append mode. Otherwise, file holes may occur.</li> <li>• Currently, mainstream log components such as Log4j and Logback support log file rolling. If your log files already support rolling, skip the configuration. Otherwise, conflicts may occur.</li> <li>• You are advised to configure log file rolling for your own services to flexibly control the size and number of rolled files.</li> </ul>

Parameter	Explanation	Description
policy.logs.annotations.pathPattern	Collection path	<p>A collection path narrows down the scope of collection to specified logs.</p> <ul style="list-style-type: none"> <li>If no collection path is specified, log files in <b>.log</b>, <b>.trace</b>, and <b>.out</b> formats will be collected from the specified path.</li> <li><b>/Path/**/</b> indicates that all log files in <b>.log</b>, <b>.trace</b>, and <b>.out</b> formats will be recursively collected from the specified path and all subdirectories at 5 levels deep.</li> <li><b>*</b> in log file names indicates a fuzzy match.</li> </ul> <p>Example: The collection path <b>/tmp/**/test*.log</b> indicates that all <b>.log</b> files prefixed with <b>test</b> will be collected from <b>/tmp</b> and subdirectories at 5 levels deep.</p> <p><b>CAUTION</b> Ensure that the ICAgent version is 5.12.22 or later.</p>

## Viewing Logs

After a log collection path is configured and the workload is created, the ICAgent collects log files from the configured path. The collection takes about 1 minute.

After the log collection is complete, go to the workload details page and click **Logs** in the upper right corner to view logs.

You can also view logs on the AOM console.

You can also run the **kubectl logs** command to view the standard output of a container.

```
# View logs of a specified pod.
kubectl logs <pod_name>
kubectl logs -f <pod_name> # Similar to tail -f

# View logs of a specified container in a specified pod.
kubectl logs <pod_name> -c <container_name>

kubectl logs pod_name -c container_name -n namespace (one-off query)
kubectl logs -f <pod_name> -n namespace (real-time query in tail -f mode)
```

## 17.11 Namespaces

### 17.11.1 Creating a Namespace

#### When to Use Namespaces

A namespace is a collection of resources and objects. Multiple namespaces can be created inside a cluster and isolated from each other. This enables namespaces to share the same cluster Services without affecting each other.

For example, you can deploy workloads in a development environment into one namespace, and deploy workloads in a testing environment into another namespace.

## Prerequisites

At least one cluster has been created. For details, see [Buying a CCE Cluster](#).

## Notes and Constraints

A maximum of 6,000 Services can be created in each namespace. The Services mentioned here indicate the Kubernetes Service resources added for workloads.

## Namespace Types

Namespaces can be created in either of the following ways:

- Created automatically: When a cluster is up, the **default**, **kube-public**, **kube-system**, and **kube-node-lease** namespaces are created by default.
  - **default**: All objects for which no namespace is specified are allocated to this namespace.
  - **kube-public**: Resources in this namespace can be accessed by all users (including unauthenticated users), such as public add-ons and container charts.
  - **kube-system**: All resources created by Kubernetes are in this namespace.
  - **kube-node-lease**: Each node has an associated Lease object in this namespace. The object is periodically updated by the node. Both NodeStatus and NodeLease are considered as heartbeats from a node. In versions earlier than v1.13, only NodeStatus is available. The NodeLease feature is introduced in v1.13. NodeLease is more lightweight than NodeStatus. This feature significantly improves the cluster scalability and performance.
- Created manually: You can create namespaces to serve separate purposes. For example, you can create three namespaces, one for a development environment, one for joint debugging environment, and one for test environment. You can also create one namespace for login services and one for game services.

## Creating a Namespace

**Step 1** Log in to the CCE console. In the navigation pane, choose **Resource Management** > **Namespaces**. Click **Create Namespace**.

**Step 2** Set the parameters listed in [Table 17-121](#). The parameters marked with an asterisk (\*) are mandatory.

**Table 17-121** Parameters for creating a namespace

Parameter	Description
* Namespace	Unique name of the created namespace.

Parameter	Description
* Cluster	Cluster to which the namespace belongs.
Node Affinity	<p>If this parameter is set to on, workloads in this namespace will be scheduled only to nodes with specified labels. To add labels to a node, choose <b>Resource Management &gt; Nodes &gt; Manage Labels</b>.</p> <p>This parameter is displayed only for clusters of v1.13.10-r0 and later.</p>
Description	Description about the namespace.
Set Resource Quotas	<p>Resource quotas can limit the amount of resources available in namespaces, achieving resource allocation by namespace.</p> <p><b>NOTICE</b>  <b>You are advised to set resource quotas in the namespace as required to prevent cluster or node exceptions caused by resource overload.</b></p> <p>For example, the default number of pods that can be created on each node in a cluster is 110. If you create a cluster with 50 nodes, you can create a maximum of 5,500 pods. Therefore, you can set a resource quota to ensure that the total number of pods in all namespaces does not exceed 5,500.</p> <p>Quotas can be configured for the following resources:</p> <ul style="list-style-type: none"> <li>● CPU (cores)</li> <li>● Memory (MiB)</li> <li>● StatefulSet</li> <li>● Deployment</li> <li>● Job</li> <li>● Cron job</li> <li>● Pod</li> <li>● Service</li> </ul> <p>Enter an integer. If the quota of a resource is set to <b>0</b>, no limit is posed on the resource.</p> <p>If you want to limit the CPU or memory quota, you must specify the CPU or memory request value when creating a workload.</p>

**Step 3** When the configuration is complete, click **OK**.

----End

## 17.11.2 Managing Namespaces

### Selecting a Namespace

- When creating a workload, you can select a namespace to isolate resources or users.

- When querying workloads, you can select a namespace to view all workloads in the namespace.

## Isolating Namespaces

- **Isolating namespaces by environment**

An application generally goes through the development, joint debugging, and testing stages before it is launched. In this process, the workloads deployed in each environment (stage) are the same, but are logically defined. There are two ways to define them:

- Group them in different clusters for different environments.

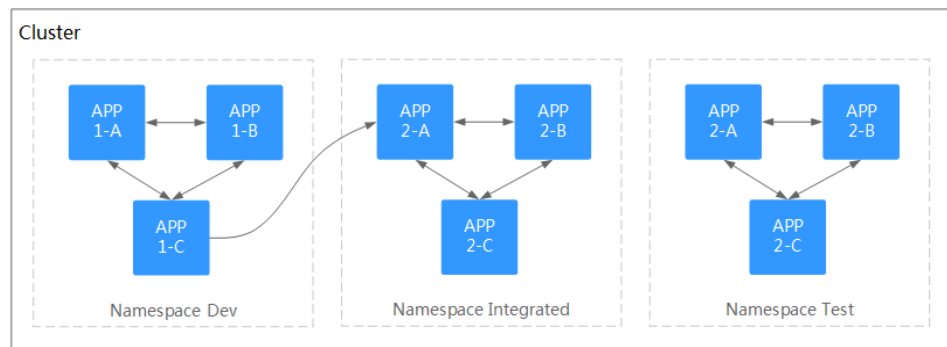
Resources cannot be shared among different clusters. In addition, services in different environments can access each other only through load balancing.

- Group them in different namespaces for different environments.

Workloads in the same namespace can be mutually accessed by using the Service name. Cross-namespace access can be implemented by using the Service name or namespace name.

The following figure shows namespaces created for the development, joint debugging, and testing environments, respectively.

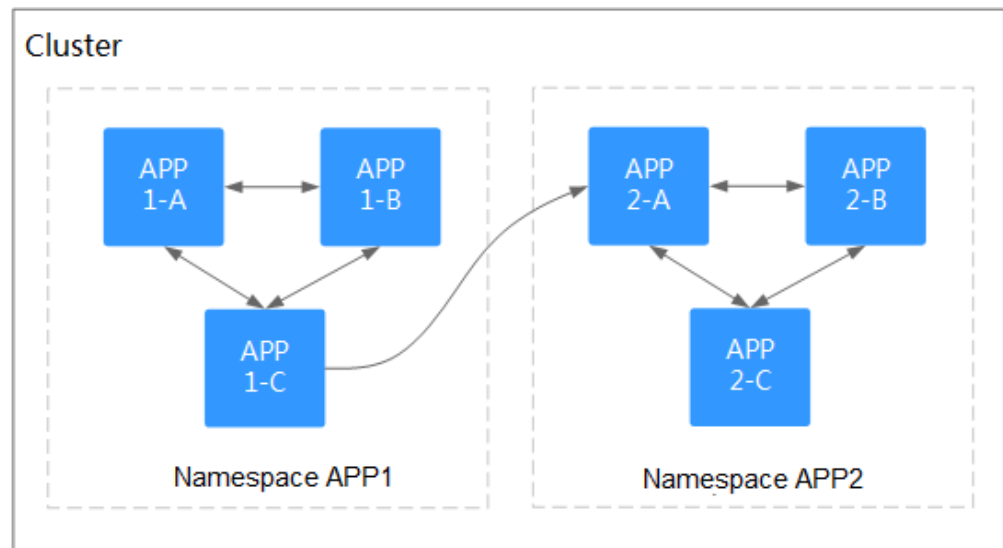
**Figure 17-142** One namespace for one environment



- **Isolating namespaces by application**

You are advised to use this method if a large number of workloads are deployed in the same environment. For example, in the following figure, different namespaces (APP1 and APP2) are created to logically manage workloads as different groups. Workloads in the same namespace access each other using the Service name, and workloads in different namespaces access each other using the Service name or namespace name.

**Figure 17-143** Grouping workloads into different namespaces



## Deleting a Namespace

If a namespace is deleted, all resources (such as workloads, jobs, and ConfigMaps) in this namespace will also be deleted. Exercise caution when deleting a namespace.

- Step 1** Log in to the CCE console. In the navigation pane, choose **Resource Management > Namespaces**.
- Step 2** Select the cluster to which the namespace belongs from the **Clusters** drop-down list.
- Step 3** Select the namespace to be deleted and click **Delete**.

Follow the prompts to delete the namespace. The default namespaces cannot be deleted.

----End

### 17.11.3 Configuring a Namespace-level Network Policy

You can configure a namespace-level network policy after enabling network isolation.

By default, **Network Isolation** is disabled for namespaces. For example, if network isolation is off for namespace **default**, **all workloads in the current cluster** can access the workloads in namespace **default**.

To prevent other workloads from accessing the workloads in namespace **default**, perform the following steps:

---

#### NOTICE

Only clusters that use the tunnel network model support network isolation.

---

## Prerequisites

- You have created a Kubernetes cluster. For details, see [Buying a CCE Cluster](#).
- You have created a namespace. For details, see [Creating a Namespace](#).

## Procedure

- Step 1** Log in to the CCE console. In the navigation pane, choose **Resource Management > Namespaces**.
- Step 2** Select the cluster to which the namespace belongs from the **Clusters** drop-down list.
- Step 3** At the row of a namespace (for example, **default**), switch on **Network Isolation**.

After network isolation is enabled, workloads in namespace **default** can access each other but they cannot be accessed by workloads in other namespaces.

**Figure 17-144** Setting network isolation for a namespace

<input type="checkbox"/> Namespace	Status	Description	Created	Network Isolation	Node Affinity
<input type="checkbox"/> default	<span style="color: green;">●</span> Available		May 18, 2020 14:30:27 GMT+08:00	<input type="checkbox"/> Off	<input type="checkbox"/>
<input type="checkbox"/> [blurred]	<span style="color: green;">●</span> Available		May 18, 2020 14:30:25 GMT+08:00	<input type="checkbox"/> Off	<input type="checkbox"/>
<input type="checkbox"/> [blurred]	<span style="color: green;">●</span> Available		May 18, 2020 14:30:25 GMT+08:00	<input type="checkbox"/> Off	<input type="checkbox"/>
<input type="checkbox"/> [blurred]	<span style="color: green;">●</span> Available		May 18, 2020 14:30:25 GMT+08:00	<input type="checkbox"/> Off	<input type="checkbox"/>

----End

## Network Isolation Description

Enabling network isolation is to create a network policy in a namespace. The network policy selects all pods in the namespace and prevents pods in other namespaces from accessing.

```
kind: NetworkPolicy
apiVersion: networking.k8s.io/v1
metadata:
  name: deny-default
  namespace: default
spec:
  ingress:
    - from:
      - podSelector: {}
    podSelector: {} # {} indicates that all pods are selected.
```

You can also customize a network policy. For details, see [Configuring Network Policies to Restrict Pod Access](#).

## 17.11.4 Setting a Resource Quota

Namespace-level resource quotas limit the amount of resources available to teams or users when these teams or users use the same cluster. The quotas include the total number of a type of objects and the total amount of compute resources (CPU and memory) consumed by the objects.

## Prerequisites

- You have created a Kubernetes cluster. For details, see [Buying a CCE Cluster](#).
- You have created a namespace. For details, see [Creating a Namespace](#).

## Usage

By default, running pods can use the CPUs and memory of a node without restrictions. This means the pods in a namespace may exhaust all resources of the cluster.

Kubernetes provides namespaces for you to group workloads in a cluster. By setting resource quotas for each namespace, you can prevent resource exhaustion and ensure cluster reliability.

You can configure quotas for resources such as CPU, memory, and the number of pods in a namespace. For more information, see [Resource Quotas](#).

The following table recommends how many pods you can configure for your clusters of different sizes.

Cluster Scale	Recommended Number of Pods
50 nodes	2,500 pods
200 nodes	10,000 pods
1,000 nodes	30,000 pods
2,000 nodes	50,000 pods

Starting from clusters of v1.21 and later, the default [Resource Quotas](#) are created when a namespace is created. [Table 17-122](#) lists the resource quotas based on cluster specifications. You can modify them according to your service requirements.

**Table 17-122** Default resource quotas

Cluster Scale	Pod	Deployment	Secret	ConfigMap	Service
50 nodes	2000	1000	1000	1000	1000
200 nodes	2000	1000	1000	1000	1000
1,000 nodes	5000	2000	2000	2000	2000
2,000 nodes	5000	2000	2000	2000	2000

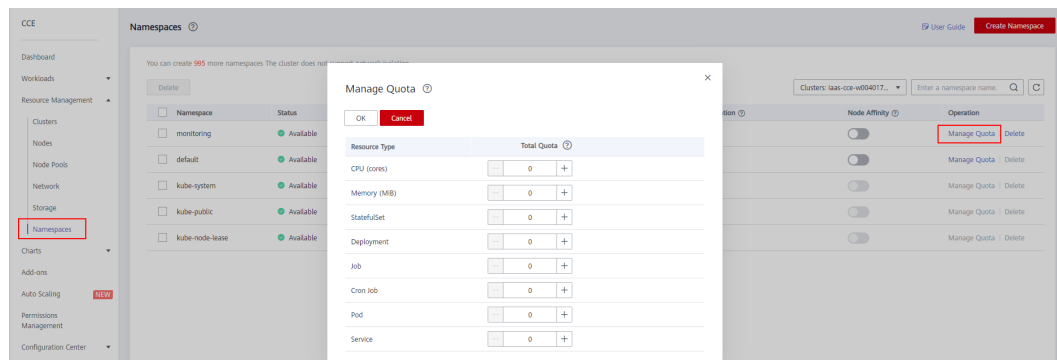


## Procedure

- Step 1** Log in to the CCE console. In the navigation pane, choose **Resource Management > Namespaces**.
- Step 2** Select the cluster to which the namespace belongs from the **Clusters** drop-down list.
- Step 3** In the **Operation** column of a namespace, click **Manage Quota**.

This operation cannot be performed on system namespaces **kube-system** and **kube-public**.

**Figure 17-145** Managing quotas



- Step 4** Set the resource quotas and click **OK**.
  - **CPU (cores):** maximum number of CPU cores that can be allocated to workload pods in the namespace.
  - **Memory (MiB):** maximum amount of memory that can be allocated to workload pods in the namespace.
  - **StatefulSet:** maximum number of StatefulSets that can be created in the namespace.
  - **Deployment:** maximum number of Deployments that can be created in the namespace.
  - **Job:** maximum number of one-off jobs that can be created in the namespace.
  - **Cron Job:** maximum number of cron jobs that can be created in the namespace.
  - **Pod:** maximum number of pods that can be created in the namespace.
  - **Service:** maximum number of Services that can be created in the namespace.

#### NOTICE

- After setting CPU and memory quotas for a namespace, you must specify the request and limit values of CPU and memory resources when creating a workload. Otherwise, the workload cannot be created. If the quota of a resource is set to **0**, the resource usage is not limited.
- Accumulated quota usage includes the resources used by CCE to create default components, such as the Kubernetes Services (which can be viewed using `kubectl`) created under the **default** namespace. Therefore, you are advised to set a resource quota greater than expected to reserve resource for creating default components.

----End

## 17.12 Configuration Center

### 17.12.1 Creating a ConfigMap

#### Scenario

A ConfigMap is a type of resource that stores configuration information required by a workload. Its content is user-defined. After creating ConfigMaps, you can use them as files or environment variables in a containerized workload.

ConfigMaps allow you to decouple configuration files from container images to enhance the portability of containerized workloads.

Benefits of ConfigMaps:

- Manage configurations of different environments and services.
- Deploy workloads in different environments. Multiple versions are supported for configuration files so that you can update and roll back workloads easily.
- Quickly import configurations in the form of files to containers.

#### Prerequisites

Cluster and node resources have been created. For more information, see [Buying a CCE Cluster](#). If you have available clusters and node resources, skip this operation.

#### Procedure

- Step 1** Log in to the CCE console. In the navigation pane, choose **Configuration Center > ConfigMaps**. Click **Create ConfigMap**.
- Step 2** You can create a ConfigMap directly or based on YAML. If you create a ConfigMap based on YAML, go to [Step 4](#).
- Step 3** Method 1: Create a ConfigMap directly.  
Set the parameters by referring to [Table 17-123](#).

**Table 17-123** Parameters for creating a ConfigMap

Parameter	Description
Name	Name of a ConfigMap, which must be unique in a namespace.
Cluster	Cluster that will use the ConfigMap you create.
Namespace	Namespace to which the ConfigMap belongs. If you do not specify this parameter, the value <b>default</b> is used by default.
Description	Description of the ConfigMap.
Data	The workload configuration data can be used in a container or used to store the configuration data. <b>Key</b> indicates a file name. <b>Value</b> indicates the content in the file. <ol style="list-style-type: none"> <li>1. Click <b>Add Data</b>.</li> <li>2. Set <b>Key</b> and <b>Value</b>.</li> </ol>
Labels	Labels are attached to objects such as workloads, nodes, and Services in key-value pairs. Labels define the identifiable attributes of these objects and are used to manage and select the objects. <ol style="list-style-type: none"> <li>1. Click <b>Add Label</b>.</li> <li>2. Set <b>Key</b> and <b>Value</b>.</li> </ol>

**Step 4** Method 2: Create a ConfigMap based on YAML.

 **NOTE**

To create ConfigMaps by uploading a file, ensure that the resource description file has been created. CCE supports files in YAML format. For more information, see [ConfigMap Requirements](#).

Click **Create YAML** on the right of the page.

- Method 1: Import the orchestration file.  
Click **Add File** to import the file in YAML format. The orchestration content can be directly displayed.
- Method 2: Directly orchestrate the content.  
In the orchestration content area, enter the content of the YAML file.

**Step 5** After the configuration is complete, click **Create**.

The new ConfigMap is displayed in the ConfigMap list.

----End

## ConfigMap Requirements

A ConfigMap resource file must be in YAML format, and the file size cannot exceed 2 MB.

The file name is **configmap.yaml** and the following shows a configuration example.

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: test-configmap
data:
  data-1: value-1
  data-2: value-2
```

## Creating a ConfigMap Using kubectl

**Step 1** Configure the **kubectl** command to connect an ECS to the cluster. For details, see [Connecting to a Cluster Using kubectl](#).

**Step 2** Create and edit the **cce-configmap.yaml** file.

**vi cce-configmap.yaml**

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: cce-configmap
data:
  SPECIAL_LEVEL: Hello
  SPECIAL_TYPE: CCE
```

**Step 3** Run the following commands to create a ConfigMap.

**kubectl create -f cce-configmap.yaml**

**kubectl get cm**

NAME	DATA	AGE
cce-configmap	3	3h
cce-configmap1	3	7m

----End

## Related Operations

After creating a configuration item, you can update or delete it as described in [Table 17-124](#).

**Table 17-124** Related operations

Operation	Description
Viewing a YAML file	Click <b>View YAML</b> next to the ConfigMap name to view the YAML file corresponding to the current ConfigMap.
Updating a ConfigMap	<ol style="list-style-type: none"> <li>Select the name of the ConfigMap to be updated and click <b>Update</b>.</li> <li>Modify the secret data. For more information, see <a href="#">Table 17-123</a>.</li> <li>Click <b>Update</b>.</li> </ol>

Operation	Description
Deleting a ConfigMap	Select the configuration you want to delete and click <b>Delete</b> . Follow the prompts to delete the ConfigMap.

## 17.12.2 Using a ConfigMap

After a ConfigMap is created, it can be used in three workload scenarios: environment variables, command line parameters, and data volumes.

- [Setting Workload Environment Variables](#)
- [Setting Command Line Parameters](#)
- [Attaching a ConfigMap to the Workload Data Volume](#)

The following example shows how to use a ConfigMap.

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: cce-configmap
data:
  SPECIAL_LEVEL: Hello
  SPECIAL_TYPE: CCE
```

### NOTICE

When a ConfigMap is used in a pod, the pod and ConfigMap must be in the same cluster and namespace.

## Setting Workload Environment Variables

When creating a workload, you can use a ConfigMap to set environment variables. The **valueFrom** parameter indicates the key-value pair to be referenced.

```
apiVersion: v1
kind: Pod
metadata:
  name: configmap-pod-1
spec:
  containers:
    - name: test-container
      image: busybox
      command: [ "/bin/sh", "-c", "env" ]
      env:
        - name: SPECIAL_LEVEL_KEY
          valueFrom:          ## Use valueFrom to specify the value of the env that refers to the
                             ConfigMap.
            configMapKeyRef:
              name: cce-configmap          ## Name of the referenced configuration file.
              key: SPECIAL_LEVEL          ## Key of the referenced ConfigMap.
  restartPolicy: Never
```

If you need to define the values of multiple ConfigMaps as the environment variables of the pods, add multiple environment variable parameters to the pods.

```
env:
  - name: SPECIAL_LEVEL_KEY
```

```
valueFrom:
  configMapKeyRef:
    name: cce-configmap
    key: SPECIAL_LEVEL
- name: SPECIAL_TYPE_KEY
valueFrom:
  configMapKeyRef:
    name: cce-configmap
    key: SPECIAL_TYPE
```

To add all data in a ConfigMap to environment variables, use the **envFrom** parameter. The keys in the ConfigMap will become names of environment variables in a pod.

```
apiVersion: v1
kind: Pod
metadata:
  name: configmap-pod-2
spec:
  containers:
  - name: test-container
    image: busybox
    command: [ "/bin/sh", "-c", "env" ]
    envFrom:
    - configMapRef:
        name: cce-configmap
  restartPolicy: Never
```

## Setting Command Line Parameters

You can use a ConfigMap to set commands or parameter values for a container by using the environment variable substitution syntax `$(VAR_NAME)`. The following shows an example.

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

After the pod runs, the following information is displayed:

```
Hello CCE
```

## Attaching a ConfigMap to the Workload Data Volume

A ConfigMap can also be used in the data volume. You only need to attach the ConfigMap to the workload when creating the workload. After the mounting is complete, a configuration file with key as the file name and value as the file content is generated.

```
apiVersion: v1
kind: Pod
metadata:
  name: configmap-pod-4
spec:
  containers:
    - name: test-container
      image: busybox
      command: [ "/bin/sh", "-c", "ls /etc/config/" ] ## Lists the file names in the directory.
      volumeMounts:
        - name: config-volume
          mountPath: /etc/config ## Attaches to the /etc/config directory.
  volumes:
    - name: config-volume
      configMap:
        name: cce-configmap
      restartPolicy: Never
```

After the pod is run, the **SPECIAL\_LEVEL** and **SPECIAL\_TYPE** files are generated in the **/etc/config** directory. The contents of the files are Hello and CCE, respectively. Also, the following file names will be displayed.

```
SPECIAL_TYPE
SPECIAL_LEVEL
```

To mount a ConfigMap to a data volume, you can also perform operations on the CCE console. When creating a workload, set advanced settings for the container, choose **Data Storage > Local Volume**, click **Add Local Volume**, and select **ConfigMap**. For details, see [ConfigMap](#).

## 17.12.3 Creating a Secret

### Scenario

A secret is a type of resource that holds sensitive data, such as authentication and key information. Its content is user-defined. After creating secrets, you can use them as files or environment variables in a containerized workload.

### Prerequisites

Cluster and node resources have been created. For more information, see [Buying a CCE Cluster](#). If you have available clusters and node resources, skip this operation.

### Procedure

- Step 1** Log in to the CCE console. In the navigation pane, choose **Configuration Center > Secrets**. Click **Create Secret**.
- Step 2** You can create a secret directly or based on YAML. If you want to create a secret based on YAML, go to [Step 4](#).
- Step 3** Method 1: Create a secret directly.

Set the basic information by referring to [Table 17-125](#).

**Table 17-125** Parameters for creating a secret

Parameter	Description
Name	Name of the secret you create, which must be unique.
Cluster	Cluster that will use the secret you create.
Namespace	Namespace to which the secret belongs. If you do not specify this parameter, the value <b>default</b> is used by default.
Description	Description of a secret.
Type	Type of the secret you create. <ul style="list-style-type: none"> <li>• Opaque: common secret.</li> <li>• kubernetes.io/dockerconfigjson: a secret that stores the authentication information required for pulling images from a private repository.</li> <li>• IngressTLS: a secret that stores the certificate required by ingresses (layer-7 load balancing Services).</li> <li>• Other: another type of secret, which is specified manually.</li> </ul>
Secret Data	Workload secret data can be used in containers. <ul style="list-style-type: none"> <li>• If the secret is of the Opaque type: <ol style="list-style-type: none"> <li>1. Click <b>Add Data</b>.</li> <li>2. Enter the key and value. The value must be based on the Base64 coding method. For details about the method, see <a href="#">Base64 Encoding</a>.</li> </ol> </li> <li>• If the secret type is kubernetes.io/dockerconfigjson, enter the account and password of the private image repository.</li> <li>• If the secret type is IngressTLS, upload the certificate file and private key file.</li> </ul> <p><b>NOTE</b></p> <ul style="list-style-type: none"> <li>- A certificate is a self-signed or CA-signed credential used for identity authentication.</li> <li>- A certificate request is a request for a signature with a private key.</li> </ul>
Secret Label	Labels are attached to objects such as workloads, nodes, and Services in key-value pairs. Labels define the identifiable attributes of these objects and are used to manage and select the objects. <ol style="list-style-type: none"> <li>1. Click <b>Add Label</b>.</li> <li>2. Enter the key and value.</li> </ol>

**Step 4** Method 2: Create a secret based on the YAML file.



**NOTE**

To create a resource by uploading a file, ensure that the resource description file has been created. CCE supports files in JSON or YAML format. For more information, see [Secret Resource File Configuration](#).

You can import or directly write the file content in YAML or JSON format.

- Method 1: Import the orchestration file.  
Click **Add File** to import the file in YAML or JSON format. The orchestration content can be directly displayed.
- Method 2: Directly orchestrate the content.  
In the orchestration content area, enter the content of the YAML or JSON file.

**Step 5** After the configuration is complete, click **Create**.

The new secret is displayed in the key list.

----End

## Secret Resource File Configuration

This section describes configuration examples of secret resource description files.

For example, you can retrieve the username and password for a workload through a secret.

- YAML format

The **secret.yaml** file is defined as shown below. The value must be based on the Base64 coding method. For details about the method, see [Base64 Encoding](#).

```
apiVersion: v1
kind: Secret
metadata:
  name: mysecret      #Secret name
  namespace: default #Namespace. The default value is default.
data:
  username: ***** #The value must be Base64-encoded.
  password: ***** #The value must be encoded using Base64.
type: Opaque        #You are advised not to change this parameter value.
```

## Creating a Secret Using kubectl

**Step 1** According to [Connecting to a Cluster Using kubectl](#), configure the **kubectl** command to connect an ECS to the cluster.

**Step 2** Create and edit the Base64-encoded **cce-secret.yaml** file.

```
# echo -n "content to be encoded" | base64
*****
```

**vi cce-secret.yaml**

```
apiVersion: v1
kind: Secret
metadata:
  name: mysecret
type: Opaque
data:
  username: *****
  password: *****
```

**Step 3** Create a secret.

```
kubectl create -f cce-secret.yaml
```

You can query the secret after creation.

```
kubectl get secret
```

```
----End
```

## Related Operations

After creating a secret, you can update or delete it as described in [Table 17-126](#).

### NOTE

The secret list contains system secret resources that can be queried only. The system secret resources cannot be updated or deleted.

**Table 17-126** Related Operations

Operation	Description
Viewing a YAML file	Click <b>View YAML</b> next to the secret name to view the YAML file corresponding to the current secret.
Updating a secret	<ol style="list-style-type: none"> <li>1. Select the name of the secret to be updated and click <b>Update</b>.</li> <li>2. Modify the secret data. For more information, see <a href="#">Table 17-125</a>.</li> <li>3. Click <b>Update</b>.</li> </ol>
Deleting a secret	Select the secret you want to delete and click <b>Delete</b> . Follow the prompts to delete the secret.
Deleting secrets in batches	<ol style="list-style-type: none"> <li>1. Select the secrets to be deleted.</li> <li>2. Click <b>Delete</b> above the secret list.</li> <li>3. Follow the prompts to delete the secrets.</li> </ol>

## Base64 Encoding

To encrypt a character string using Base64, run the `echo -n to-be-encoded content | base64` command. The following is an example.

```
root@ubuntu:~# echo -n "content to be encoded" | base64
*****
```

### 17.12.4 Using a Secret

After secrets are created, they can be mounted as data volumes or be exposed as environment variables to be used by a container in a pod.

**NOTICE**

The following secrets are used by the CCE system. Do not perform any operations on them.

- Do not operate secrets under kube-system.
- Do not operate default-secret and paas.elb in any of the namespaces. The default-secret is used to pull the private image of SWR, and the paas.elb is used to connect the service in the namespace to the ELB service.

- [Configuring the Data Volume of a Pod](#)
- [Setting Environment Variables of a Pod](#)

The following example shows how to use a secret.

```
apiVersion: v1
kind: Secret
metadata:
  name: mysecret
type: Opaque
data:
  username: ***** #The value must be Base64-encoded.
  password: ***** #The value must be encoded using Base64.
```

**NOTICE**

When a secret is used in a pod, the pod and secret must be in the same cluster and namespace.

## Configuring the Data Volume of a Pod

A secret can be used as a file in a pod. As shown in the following example, the username and password of the **mysecret** secret are saved in the **/etc/foo** directory as files.

```
apiVersion: v1
kind: Pod
metadata:
  name: mypod
spec:
  containers:
    - name: mypod
      image: redis
      volumeMounts:
        - name: foo
          mountPath: "/etc/foo"
          readOnly: true
  volumes:
    - name: foo
      secret:
        secretName: mysecret
```

In addition, you can specify the directory and permission to access a secret. The username is stored in the **/etc/foo/my-group/my-username** directory of the container.

```
apiVersion: v1
kind: Pod
metadata:
  name: mypod
spec:
```

```
containers:
- name: mypod
  image: redis
  volumeMounts:
  - name: foo
    mountPath: "/etc/foo"
volumes:
- name: foo
  secret:
    secretName: mysecret
    items:
    - key: username
      path: my-group/my-username
      mode: 511
```

To mount a secret to a data volume, you can also perform operations on the CCE console. When creating a workload, set advanced settings for the container, choose **Data Storage > Local Volume**, click **Add Local Volume**, and select **Secret**. For details, see [Secret](#).

## Setting Environment Variables of a Pod

A secret can be used as an environment variable of a pod. As shown in the following example, the username and password of the **mysecret** secret are defined as an environment variable of the pod.

```
apiVersion: v1
kind: Pod
metadata:
  name: secret-env-pod
spec:
  containers:
  - name: mycontainer
    image: redis
    env:
    - name: SECRET_USERNAME
      valueFrom:
        secretKeyRef:
          name: mysecret
          key: username
    - name: SECRET_PASSWORD
      valueFrom:
        secretKeyRef:
          name: mysecret
          key: password
  restartPolicy: Never
```

### 17.12.5 Cluster Secrets

By default, CCE creates the following secrets in each namespace:

- default-secret
- paas.elb
- default-token-xxxxx (xxxxx is a random number.)



The functions of these secrets are described as follows.

## default-secret

The type of **default-secret** is **kubernetes.io/dockerconfigjson**. The data is the credential for logging in to the SWR image repository and is used to pull images from SWR. If you need to pull an image from SWR when creating a workload on CCE, set **imagePullSecrets** to **default-secret**.

```
apiVersion: v1
kind: Pod
metadata:
  name: nginx
spec:
  containers:
    - image: nginx:alpine
      name: container-0
  resources:
    limits:
      cpu: 100m
      memory: 200Mi
    requests:
      cpu: 100m
      memory: 200Mi
  imagePullSecrets:
    - name: default-secret
```

The data of **default-secret** is updated periodically, and the current data will expire after a certain period of time. You can run the **describe** command to view the expiration time in of default-secret.

### NOTICE

Use default-secret directly instead of copying the secret content to create a new one. The credential in the copied secret will expire and the image cannot be pulled.

```
$ kubectl describe secret default-secret
Name:         default-secret
Namespace:    default
Labels:       secret-generated-by=cce
Annotations:  temporary-ak-sk-expires-at: 2021-11-26 20:55:31.380909 +0000 UTC
```

```
Type: kubernetes.io/dockerconfigjson
```

```
Data
```

```
====
```

```
.dockerconfigjson: 347 bytes
```

## paas.elb

The data of **paas.elb** is the temporary AK/SK data, which is used to create ELB load balancers during Service and ingress creation. The data of paas.elb is periodically updated and expires after a certain period of time.

In practice, you will not directly use paas.elb. However, do not delete it. Otherwise, ELB load balancers will fail to be created.

## default-token-xxxxx

By default, Kubernetes creates a service account named **default** for each namespace. **default-token-xxxxx** is the key of the service account, and **xxxxx** is a random number.

```
$ kubectl get sa
NAME      SECRETS  AGE
default  1         30d
$ kubectl describe sa default
Name:      default
Namespace: default
Labels:    <none>
Annotations: <none>
Image pull secrets: <none>
Mountable secrets: default-token-vssmw
Tokens:    default-token-vssmw
Events:    <none>
```

# 17.13 Charts (Helm)

## 17.13.1 My Charts

### 17.13.1.1 Overview

CCE uses Helm, a Kubernetes package manager, to simplify deployment and management of packages (also called charts). A chart is a collection of files that describe a related set of Kubernetes resources. The use of charts handles all the complexity in Kubernetes resource installation and management, making it possible to achieve unified resource scheduling and management.

#### NOTE

Helm is a tool for packaging Kubernetes applications. For more information, see [Helm documentation](#).

Custom charts simplify workload deployment.

This section describes how to create a workload using a custom chart. You can use multiple methods to create an orchestration chart on the CCE console.

## Notes and Constraints

- The number of charts that can be uploaded by a single user is limited. The value displayed on the console of each region is the allowed quantity.
- CCE uses Helm v3.8.2. If you use Helm v2 to manage CCE, compatibility problems may occur.
- A chart with multiple versions consumes the same amount of portion of chart quota.
- Users with chart operation permissions can perform multiple operations on clusters. Therefore, exercise caution when assigning users the chart lifecycle management permissions, including uploading charts and creating, deleting, and updating chart releases.

### 17.13.1.2 Preparing a Chart

You can prepare a chart using one of the following methods:

- [Customizing a Chart](#)
- [Using a Kubernetes Official Chart](#)

## Customizing a Chart

**Step 1** Customize the content of a chart as required.

For details about how to create a chart, see [https://helm.sh/docs/chart\\_template\\_guide/getting\\_started/](https://helm.sh/docs/chart_template_guide/getting_started/).

**Step 2** Set the chart directory structure and name the chart based on the requirements defined in [Chart Specifications](#).

----End

## Using a Kubernetes Official Chart

**Step 1** Visit <https://artifacthub.io/> to obtain the required chart.

**Step 2** Log in to a Linux host.

**Step 3** Upload the chart obtained in [Step 1](#).

**Step 4** Run the following command to compress the chart.

- If the Helm client is not installed on the Linux host, run the following command:

```
tar pzc {name}-{version}.tgz {name}/
```

In the preceding command,

**{name}** indicates the actual chart name.

**{version}** indicates the actual chart version.

---

### NOTICE

The values of **{name}** and **{version}** must be the same as the values of name and version in the **Chart.yaml** file in the chart.

---

- If the Helm client is installed on the Linux host, run the following command:  
**helm package {name}/**  
In the preceding command, replace **{name}** with the actual chart name.

**Step 5** Set the chart directory structure and name the chart based on the requirements defined in [Chart Specifications](#).

----End

## Chart Specifications

This section uses the redis chart as an example to illustrate the chart specifications.

- **Naming Requirement**

A chart package is named in the format of **{name}-{version}.tgz**, where **{version}** indicates the version number in the format of *Major version number.Minor version number.Revision number*, for example, **redis-0.4.2.tgz**.

 **NOTE**

The chart name {name} can contain a maximum of 64 characters.

The version number must comply with the [semantic versioning](#) rules.

- The main and minor version numbers are mandatory, and the revision number is optional.
- The major and minor version numbers and revision number must be integers, greater than or equal to 0, and less than or equal to 99.

- **Directory Structure**

The directory structure of a chart is as follows:


```
redis/
  templates/
  values.yaml
  README.md
  Chart.yaml
  .helmignore
```

As listed in [Table 17-127](#), the parameters marked with \* are mandatory.

**Table 17-127** Parameters in the directory structure of a chart

Parameter	Description
* templates	Stores all templates.



Parameter	Description
* values.yaml	<p>Describes configuration parameters required by templates.</p> <p><b>NOTICE</b> Make sure that the image address set in the <b>values.yaml</b> file is the same as the image address in the container image repository. Otherwise, an exception occurs when you create a workload, and the system displays a message indicating that the image fails to be pulled.</p> <p>To obtain the image address, perform the following operations: Log in to the CCE console. In the navigation pane, choose <b>Image Repository</b> to access the SWR console. Choose <b>My Images &gt; Private Images</b> and click the name of the uploaded image. On the <b>Image Tags</b> tab page, obtain the image address from the pull command. You can click  to copy the command in the <b>Image Pull Command</b> column.</p>
README.md	<p>A markdown file, including:</p> <ul style="list-style-type: none"> <li>• The workload or services provided by the chart.</li> <li>• Prerequisites for running the chart.</li> <li>• Configurations in the <b>values.yaml</b> file.</li> <li>• Information about chart installation and configuration.</li> </ul>
* Chart.yaml	Basic information about the chart.
.helmignore	Files or data that does not need to read templates during workload installation.

### 17.13.1.3 Uploading a Chart

#### Scenario

Upload a chart to **Charts > Uploaded Charts** for subsequent workload creation.

#### Procedure

**Step 1** Log in to the CCE console. In the navigation pane, choose **Charts > Uploaded Charts** and click **Upload Chart**.

**Step 2** Click **Select File**, select the chart to be uploaded, and click **Upload**.

 **NOTE**

When you upload a chart, the naming rule of the OBS bucket is changed from `cce-charts-{region}-{domain_name}` to `cce-charts-{region}-{domain_id}`. In the old naming rule, the system converts the **domain\_name** value into a Base64 string and uses the first 63 characters. If you cannot find the chart in the OBS bucket with the new name, search for the bucket with the old name.

----End

## Related Operations

After a chart is created, you can perform operations listed in [Table 17-128](#) on the **Uploaded Charts** page.

**Table 17-128** Related operations

Operation	Description
Installing a chart	Click <b>Install Chart</b> to install the chart for creating workloads. For details, see <a href="#">Creating a Workload from a Chart</a> .
Updating a chart	The chart content will be updated while the chart version remains unchanged. The procedure is similar to that of uploading a chart.
Downloading a chart	Click <b>More &gt; Download</b> to download the chart to the local host.
Deleting a chart	Click <b>More &gt; Delete</b> to delete the installed chart. Deleted charts cannot be restored. Exercise caution when performing this operation.

### 17.13.1.4 Creating a Workload from a Chart

#### Creating a Chart-based Workload

- Step 1** Log in to the CCE console. In the navigation pane, choose **Charts > Uploaded Chart**.
- Step 2** In the list of uploaded charts, click **Install**.
- Step 3** Set the installation parameters listed in [Table 17-129](#). The parameters marked with an asterisk (\*) are mandatory.

**Table 17-129** Installation parameters

Parameter	Description
* Release Name	Unique name of the chart release.
* Chart Version	Chart version by default.
* Cluster	Cluster where the workload will be deployed.
* Namespace	Namespace to which the workload will be deployed.

Parameter	Description
Advanced Settings	<p>You can import and replace the <b>values.yaml</b> file or directly edit the chart parameters online.</p> <p><b>NOTE</b> An imported <b>values.yaml</b> file must comply with YAML specifications, that is, KEY:VALUE format. The fields in the file are not restricted. The key value of the imported values.yaml must be the same as that of the selected chart package. Otherwise, the values.yaml does not take effect. That is, the key cannot be changed.</p> <ol style="list-style-type: none"> <li>1. Click <b>Import Configuration File</b>.</li> <li>2. Select the corresponding <b>values.yaml</b> file and click <b>Open</b>.</li> </ol>

**Step 4** After the configuration is complete, click **Next**.

**Step 5** Confirm the configuration and click **Submit**.

**Step 6** Click **Back to Release List** to view the running status of the chart-based workload (also called release), or click **View Release Details** to view details about the release.

----End

## Upgrading a Chart-based Workload

**Step 1** Log in to the CCE console. In the navigation pane, choose **Charts > Uploaded Charts**. Click the **Template Instances** tab.

**Step 2** Click **Upgrade** in the row where the desired workload resides and set the parameters for the workload.

**Step 3** Select a chart version for **Chart Version**.

**Step 4** Follow the prompts to modify the chart parameters. Click **Upgrade**, and then click **Submit**.

**Step 5** Click **Back to Release List**. If the chart status changes to **Upgrade successful**, the workload is successfully upgraded.

----End

## Rolling Back a Chart-based Workload

**Step 1** Log in to the CCE console. In the navigation pane, choose **Charts > Uploaded Charts**. Click the **Template Instances** tab.

**Step 2** Click **More > Roll Back** for the workload to be rolled back, select the workload version, and click **Roll back to this version**.

In the workload list, if the status is **Rollback successful**, the workload is rolled back successfully.

----End

## Uninstalling a Chart-based Workload

- Step 1** Log in to the CCE console. In the navigation pane, choose **Charts > Uploaded Charts**. Click the **Template Instances** tab.
- Step 2** Click **More > Uninstall** next to the release to be uninstalled, and click **Yes**. Exercise caution when performing this operation because releases cannot be restored after being uninstalled.

----End

## 17.14 Add-ons

### 17.14.1 Overview

CCE provides multiple types of add-ons to extend cluster functions and meet feature requirements. You can install add-ons as required.

**Table 17-130** Add-on list

Add-on Name	Introduction
<a href="#">coredns</a>	The coredns add-on is a DNS server that provides domain name resolution services for Kubernetes clusters. coredns chains plug-ins to provide additional features.
<a href="#">everest</a>	Everest is a cloud native container storage system. Based on CSI, clusters of Kubernetes v1.15.6 and later can connect to storage services such as EVS, OBS, SFS, and SFS Turbo.
<a href="#">autoscaler</a>	The autoscaler add-on resizes a cluster based on pod scheduling status and resource usage.
<a href="#">metrics-server</a>	metrics-server is an aggregator for monitoring data of core cluster resources.
<a href="#">cce-hpa-controller</a>	cce-hpa-controller is a CCE-developed add-on, which can be used to flexibly scale in or out Deployments based on metrics such as CPU usage and memory usage.
<a href="#">prometheus</a>	Prometheus is an open-source system monitoring and alerting framework. CCE allows you to quickly deploy Prometheus as an add-on.
<a href="#">gpu-beta</a>	gpu-beta is a device management add-on that supports GPUs in containers. It supports only NVIDIA drivers.

## 17.14.2 coredns (System Resource Add-on, Mandatory)

### Introduction

The coredns add-on is a DNS server that provides domain name resolution services for Kubernetes clusters. coredns chains plug-ins to provide additional features.

coredns is an open-source software and has been a part of CNCF. It provides a means for cloud services to discover each other in cloud-native deployments. Each of the plug-ins chained by coredns provides a particular DNS function. You can integrate coredns with only the plug-ins you need to make it fast, efficient, and flexible. When used in a Kubernetes cluster, coredns can automatically discover services in the cluster and provide domain name resolution for these services. By working with a cloud DNS server, coredns can resolve external domain names for workloads in a cluster.

**coredns is a system resource add-on. It is installed by default when a cluster of Kubernetes v1.11 or later is created.**

Kubernetes v1.11 and later back CoreDNS as the official default DNS for all clusters going forward.

CoreDNS official website: <https://coredns.io/>

Open source community: <https://github.com/coredns/coredns>

#### NOTE

For details, see [DNS Configuration](#).

### Notes and Constraints

When CoreDNS is running properly or being upgraded, ensure that the number of available nodes is greater than or equal to the number of CoreDNS instances and all CoreDNS instances are running. Otherwise, the upgrade will fail.

### Installing the Add-on

This add-on has been installed by default. If it is uninstalled due to some reasons, you can reinstall it by performing the following steps:

- Step 1** Log in to the CCE console. In the navigation pane, choose **Add-ons**. On the **Add-on Marketplace** tab page, click **Install Add-on** under **coredns**.
- Step 2** On the **Install Add-on** page, select the cluster and the add-on version, and click **Next: Configuration**.
- Step 3** In the **Configuration** step, set the following parameters:

**Table 17-131** coredns add-on parameters

Parameter	Description
Add-on Specifications	Concurrent domain name resolution ability. Select add-on specifications that best fit your needs.
Instances	Number of pods that will be created to match the selected add-on specifications. The number cannot be modified.
Container	CPU and memory quotas of the container allowed for the selected add-on specifications. The quotas cannot be modified.
Notes	Add-on precautions. Read the precautions before you proceed with the step.
stub domain	A domain name server for a user-defined domain name. The format is a key-value pair. The key is a suffix of DNS domain name, and the value is one or more DNS IP addresses. For example, <b>acme.local -- 1.2.3.4,6.7.8.9</b> means that DNS requests with the <b>.acme.local</b> suffix are forwarded to a DNS listening at 1.2.3.4,6.7.8.9.

**Step 4** After the preceding configurations are complete, click **Install**.

After the add-on is installed, click **Go Back to Previous Page**. On the **Add-on Instance** tab page, select the corresponding cluster to view the running instance. This indicates that the add-on has been installed on each node in the cluster.

----End

## Configuring the Stub Domain for CoreDNS

Cluster administrators can modify the ConfigMap for the CoreDNS Corefile to change how service discovery works. They can configure stub domains for CoreDNS using the proxy plug-in.

Assume that a cluster administrator has a Consul DNS server located at 10.150.0.1 and all Consul domain names have the suffix **.consul.local**.

To configure this Consul DNS server in CoreDNS, run the following command to edit the CoreDNS ConfigMap:

**kubectl edit configmap coredns -n kube-system**

Example configuration:

```
consul.local:5353 {
  errors
  cache 30
  proxy . 10.150.0.1
}
```

**In clusters of v1.15.11 and later, the modified ConfigMap is as follows:**

```
apiVersion: v1
metadata:
  name: coredns
```

```

namespace: kube-system
selfLink: /api/v1/namespaces/kube-system/configmaps/coredns
uid: 00cb8f29-62d7-4df8-a769-0a16237903c1
resourceVersion: '2074614'
creationTimestamp: '2021-04-07T03:52:42Z'
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
    }

    consul.local:5353 {
      errors
      cache 30
      proxy . 10.150.0.1
    }
  
```

**In clusters earlier than v1.15.11, the modified ConfigMap is as follows:**

```

apiVersion: v1
data:
  Corefile: |-
    .:5353 {
      cache 30
      errors
      health
      kubernetes cluster.local in-addr.arpa ip6.arpa {
        pods insecure
        upstream /etc/resolv.conf
        fallthrough in-addr.arpa ip6.arpa
      }
      loadbalance round_robin
      prometheus 0.0.0.0:9153
      proxy . /etc/resolv.conf
      reload
    }

    consul.local:5353 {
      errors
      cache 30
      proxy . 10.150.0.1
    }
  }
kind: ConfigMap
metadata:
  name: coredns
  namespace: kube-system
  
```

## How Does Domain Name Resolution Work in Kubernetes?

DNS policies can be set on a per-pod basis. Currently, Kubernetes supports four types of DNS policies: **Default**, **ClusterFirst**, **ClusterFirstWithHostNet**, and **None**.

For details, see <https://kubernetes.io/docs/concepts/services-networking/dns-pod-service/>. These policies are specified in the **dnsPolicy** field in the pod-specific.

- **Default:** Pods inherit the name resolution configuration from the node that the pods run on. The custom upstream DNS server and the stub domain cannot be used together with this policy.
- **ClusterFirst:** Any DNS query that does not match the configured cluster domain suffix, such as **www.kubernetes.io**, is forwarded to the upstream name server inherited from the node. Cluster administrators may have extra stub domains and upstream DNS servers configured.
- **ClusterFirstWithHostNet:** For pods running with hostNetwork, set its DNS policy **ClusterFirstWithHostNet**.
- **None:** It allows a pod to ignore DNS settings from the Kubernetes environment. All DNS settings are supposed to be provided using the **dnsPolicy** field in the pod-specific.

#### NOTE

- Clusters of Kubernetes v1.10 and later support **Default**, **ClusterFirst**, **ClusterFirstWithHostNet**, and **None**. Clusters earlier than Kubernetes v1.10 support only **Default**, **ClusterFirst**, and **ClusterFirstWithHostNet**.
- **Default** is not the default DNS policy. If **dnsPolicy** is not explicitly specified, **ClusterFirst** is used.

## Routing

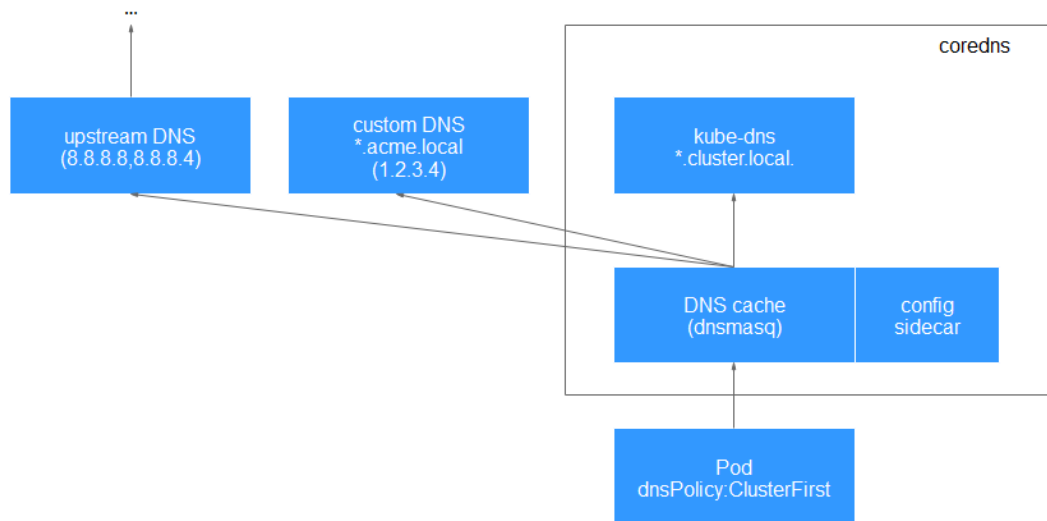
**Without stub domain configurations:** Any query that does not match the configured cluster domain suffix, such as **www.kubernetes.io**, is forwarded to the upstream DNS server inherited from the node.

**With stub domain configurations:** If stub domains and upstream DNS servers are configured, DNS queries are routed according to the following flow:

1. The query is first sent to the DNS caching layer in coredns.
2. From the caching layer, the suffix of the request is examined and then the request is forwarded to the corresponding DNS:
  - Names with the cluster suffix, for example, **.cluster.local**: The request is sent to coredns.
  - Names with the stub domain suffix, for example, **.acme.local**: The request is sent to the configured custom DNS resolver that listens, for example, on 1.2.3.4.
  - Names that do not match the suffix (for example, **widget.com**): The request is forwarded to the upstream DNS.



**Figure 17-146** Routing



## Upgrading the Add-on

**Step 1** Log in to the CCE console. In the navigation pane, choose **Add-ons**. On the **Add-on Instance** tab page, click **Upgrade** under **coredns**.

**NOTE**

- If the **Upgrade** button is unavailable, the current add-on is already up-to-date and no upgrade is required.
- During the upgrade, the previous configurations are lost and need to be specified again.
- When the upgrade is complete, the original coredns version on cluster nodes will be replaced by the latest version. If an exception occurs during the upgrade, uninstall the add-on and then re-install it.

**Step 2** On the **Basic Information** page, select the add-on version and click **Next**.

**Step 3** Configure the parameters listed in **Table 17-132**. After the configuration is complete, click **Upgrade** to upgrade the coredns add-on.

**Table 17-132** Parameters for installing coredns

Parameter	Description
Add-on Specifications	Concurrent domain name resolution ability. Select add-on specifications that best fit your needs.
stub domain	A domain name server for a user-defined domain name. The format is a key-value pair. The key is a suffix of DNS domain name, and the value is one or more DNS IP addresses. For example, <b>acme.local -- 1.2.3.4,6.7.8.9</b> means that DNS requests with the <b>.acme.local</b> suffix are forwarded to a DNS listening at 1.2.3.4,6.7.8.9.

----End

## Uninstalling the Add-on

**Step 1** Log in to the CCE console. In the navigation pane, choose **Add-ons**. On the **Add-on Instance** tab page, click **Uninstall** under **coredns**.

**Step 2** In the dialog box displayed, click **Yes** to uninstall the add-on.

----End

## 17.14.3 everest (System Resource Add-on, Mandatory)

### Introduction

Everest is a cloud-native container storage system. Based on Container Storage Interface (CSI), clusters of Kubernetes v1.15 or later can interconnect with cloud storage services such as EVS, OBS, SFS, and SFS Turbo.

**everest is a system resource add-on. It is installed by default when a cluster of Kubernetes v1.15 or later is created.**

### Notes and Constraints

- In version 1.2.0 of the everest add-on, **key authentication** is optimized when OBS is used. After the everest add-on is upgraded from a version earlier than 1.2.0, you need to restart all workloads that use OBS in the cluster. Otherwise, workloads may not be able to use OBS.

### Installing the Add-on

This add-on has been installed by default. If it is uninstalled due to some reasons, you can reinstall it by performing the following steps:

**Step 1** Log in to the CCE console. In the navigation pane, choose **Add-ons**. On the **Add-on Marketplace** tab page, click **Install Add-on** under **everest**.

**Step 2** On the **Install Add-on** page, select the cluster and the add-on version, and click **Next: Configuration**.

**Step 3** Select **Single** or **HA** for **Add-on Specifications**, and click **Install**.

After the add-on is installed, click **Go Back to Previous Page**. On the **Add-on Instance** tab page, select the corresponding cluster to view the running instance. This indicates that the add-on has been installed on each node in the cluster.

----End

### Upgrading the Add-on

**Step 1** Log in to the CCE console. In the navigation pane, choose **Add-ons**. On the **Add-on Instance** tab page, click **Upgrade** under **everest**.

#### NOTE

- If the **Upgrade** button is unavailable, the current add-on is already up-to-date and no upgrade is required.
- When the upgrade is complete, the original everest version on cluster nodes will be replaced by the latest version.

**Step 2** On the **Basic Information** page, select the add-on version and click **Next**.

**Step 3** Select **Single** or **HA** for **Add-on Specifications**, and click **Upgrade**.

----End

## Uninstalling the Add-on

**Step 1** Log in to the CCE console. In the navigation pane, choose **Add-ons**. On the **Add-on Instance** tab page, click **Uninstall** under **everest**.

**Step 2** In the dialog box displayed, click **Yes** to uninstall the add-on.

----End

## 17.14.4 autoscaler

### Introduction

Autoscaler is an important Kubernetes controller. It supports microservice scaling and is key to serverless design.

When the CPU or memory usage of a microservice is too high, horizontal pod autoscaling is triggered to add pods to reduce the load. These pods can be automatically reduced when the load is low, allowing the microservice to run as efficiently as possible.

CCE simplifies the creation, upgrade, and manual scaling of Kubernetes clusters, in which traffic loads change over time. To balance resource usage and workload performance of nodes, Kubernetes introduces the autoscaler add-on to automatically resize a cluster based on the resource usage required for workloads deployed in the cluster. For details, see [Creating a Node Scaling Policy](#).

Open source community: <https://github.com/kubernetes/autoscaler>

### How the Add-on Works

autoscaler controls auto scale-out and scale-in.

- **Auto scale-out**

If pods in a cluster cannot be scheduled due to insufficient worker nodes, cluster scaling is triggered to add nodes. The nodes to be added have the same specification as configured for the node pool to which the nodes belong. For details, see [Creating a Node Scaling Policy](#).

The add-on follows the "No Less, No More" policy. For example, if three cores are required for creating a pod and the system supports four-core and eight-core nodes, autoscaler will preferentially create a four-core node.

#### NOTE

Auto scale-out will be performed when:

- Node resources are insufficient.
- No node affinity policy is set in the pod scheduling configuration. That is, if a node has been configured as an affinity node for pods, no node will not be automatically added when pods cannot be scheduled. For details about how to configure the node affinity policy, see [Node Affinity](#).

- **Auto scale-in**

When a cluster node is idle for a period of time (10 minutes by default), cluster scale-in is triggered, and the node is automatically deleted. However, a node cannot be deleted from a cluster if the following pods exist:

- Pods that do not meet specific requirements set in PodDisruptionBudget
- Pods that cannot be scheduled to other nodes due to constraints such as affinity and anti-affinity policies
- Pods that have the **cluster-autoscaler.kubernetes.io/safe-to-evict: 'false'** annotation
- Pods (except those created by kube-system DaemonSet) that exist in the kube-system namespace on the node
- Pods that are not created by the controller (Deployment/ReplicaSet/job/StatefulSet)

### Notes and Constraints

- Ensure that there are sufficient resources for installing the add-on.
- Only **pay-per-use VM nodes** can be added or removed by autoscaler.
- The default node pool does not support auto scaling. For details, see [Description of DefaultPool](#).

### Installing the Add-on

**Step 1** Log in to the CCE console. In the navigation pane, choose **Add-ons**. On the **Add-on Marketplace** tab page, click **Install Add-on** under **autoscaler**.

**Step 2** On the **Install Add-on** page, select the cluster and the add-on version, and click **Next: Configuration**.

**Step 3** Configure add-on installation parameters listed in [Table 17-133](#).

**Table 17-133** Basic settings

Parameter	Add-on Version	Description
Add-on Specifications	Available in all versions	<p>The add-on can be deployed in the following specifications:</p> <ul style="list-style-type: none"> <li>● <b>Single</b>: The add-on is deployed with only one pod.</li> <li>● <b>HA50</b>: The add-on is deployed with two pods, serving a cluster with 50 nodes and ensuring high availability.</li> <li>● <b>HA200</b>: The add-on is deployed with two pods, serving a cluster with 50 nodes and ensuring high availability. Each pod uses more resources than those of the <b>HA50</b> specification.</li> <li>● <b>Custom</b>: You can customize the number of pods and specifications as required.</li> </ul>

Parameter	Add-on Version	Description
Instances	Available in all versions	Number of pods that will be created to match the selected add-on specifications. The number cannot be modified.
Container	Available in all versions	CPU and memory quotas of the container allowed for the selected add-on specifications. The quotas cannot be modified.
Login Mode	Available only in certain versions	<p>Select a login mode for the worker nodes to be added during auto scale-up. Passwords and key pairs are supported for login.</p> <p>If you select <b>Password</b>:</p> <ul style="list-style-type: none"> <li>● <b>Password</b>: Set a password for logging in to the added worker nodes as user <b>root</b></li> <li>● <b>Confirm Password</b>: Enter the password again.</li> </ul> <p>If you select <b>Key pair</b>:</p> <p><b>Key pair</b>: Select an existing key pair or create a new one for identity authentication during remote login to the added nodes.</p>

Parameter	Add-on Version	Description
Auto Scale-In	Available in all versions	<p><b>Off:</b> Auto scale-down is not allowed. Only auto scale-up is allowed.</p> <p><b>On:</b> Enable auto scale-in. The scale-in policy is valid for node pools in the cluster with auto scaling enabled.</p> <ul style="list-style-type: none"> <li>• <b>Idle Time (min):</b> Time for which a node should be unneeded before it is eligible for scale-down. Default value: 10 minutes.</li> <li>• <b>Resource Usage:</b> If the percentage of both CPU and memory usage on a node is below this threshold, auto scale-down will be triggered to delete the node from the cluster. The default value is 0.5, which means 50%.</li> <li>• <b>Scale-in Cooldown After Scale-out:</b> The time after scale-up that the scale-down evaluation will resume. Default value: 10 minutes.</li> </ul> <p><b>NOTE</b> If both auto scale-out and scale-in exist in a cluster, you are advised to set <b>Scale-in Cooldown After Scale-out</b> to 0 minutes. This can prevent the node scale-in from being blocked due to continuous scale-out of some node pools or retries upon a scale-out failure, resulting in unexpected waste of node resources.</p> <ul style="list-style-type: none"> <li>• <b>Scale-in Cooldown After Node Deletion:</b> The time after node deletion that the scale-down evaluation will resume. Default value: 10 minutes.</li> <li>• <b>Scale-in Cooldown After Failure:</b> The time after a scale-down failure that the scale-down evaluation will resume. Default value: 3 minutes. For details about the impact and relationship between the scale-in cooling intervals configured in the node pool and autoscaler, see <a href="#">Scale-in Cooling Interval</a>.</li> <li>• <b>Max empty bulk delete:</b> The maximum number of empty nodes that can be deleted at the same time. Default value: 10.</li> <li>• <b>Node Recheck Timeout:</b> The timeout before autoscaler checks again the node that could not be previously removed. Default value: 5 minutes.</li> </ul>

Parameter	Add-on Version	Description
Node Pool Configuration	Available only in certain versions	<p>Configuration of the default node pool. A node pool is a group of compute nodes with the same node type (VM or BMS), specifications, and labels. When a cluster needs to be scaled up, autoscaler will automatically add nodes from node pools to the cluster. If no custom node pool is available, autoscaler will use the default node pool.</p> <p>Click <b>Add Node Pool Configuration</b> and set the following parameters:</p> <ul style="list-style-type: none"> <li>● <b>AZ:</b> A physical region where resources use independent power supplies and networks. AZs are physically isolated but interconnected through the internal network.</li> <li>● <b>OS:</b> OS of the nodes to be created.</li> <li>● <b>Taints:</b> No taints are added by default. Taints allow nodes to repel a set of pods. You can add a maximum of 10 taints for each node pool. Each taint contains the following parameters: <ul style="list-style-type: none"> <li>- <b>Key:</b> A key must contain 1 to 63 characters starting with a letter or digit. Only letters, digits, hyphens (-), underscores (_), and periods (.) are allowed. A DNS subdomain name can be used as the prefix of a key.</li> <li>- <b>Value:</b> A value must start with a letter or digit and can contain a maximum of 63 characters, including letters, digits, hyphens (-), underscores (_), and periods (.).</li> <li>- <b>Effect:</b> Available options are <b>NoSchedule</b>, <b>PreferNoSchedule</b>, and <b>NoExecute</b>.</li> </ul> </li> </ul> <p><b>NOTICE</b></p> <ul style="list-style-type: none"> <li>- If taints are used, you must configure tolerations in the YAML files of pods. Otherwise, scale-up may fail or pods cannot be scheduled onto the added nodes.</li> <li>- Taints cannot be modified after configuration. Incorrect taints may cause a scale-up failure or prevent pods from being scheduled onto the added nodes.</li> </ul> <ul style="list-style-type: none"> <li>● <b>Resource Tags:</b> Resource tags can be added to classify resources.</li> </ul>

Parameter	Add-on Version	Description
		<p><b>NOTE</b> You can create <b>predefined tags</b> in Tag Management Service (TMS). Predefined tags are visible to all service resources that support the tagging function. You can use these tags to improve tagging and resource migration efficiency.</p> <ul style="list-style-type: none"> <li>• <b>Specifications:</b> CPU and memory of the added nodes.</li> </ul>

To configure more add-on parameters, click **Advanced Settings** at the bottom of this page.

**Table 17-134** Advanced settings

Parameter	Add-on Version	Description
Total Nodes	Available in all versions	Maximum number of nodes that can be managed by the cluster, within which cluster scale-out is performed.
Total Cores	Available in all versions	Maximum sum of CPU cores of all nodes in a cluster, within which cluster scale-out is performed.
Total Memory (GB)	Available in all versions	Maximum sum of memory of all nodes in a cluster, within which cluster scale-out is performed.
Auto Scale-Out	Available only in certain versions	<b>Triggered when there are pods unscheduled:</b> Selected by default.

**Step 4** When the configuration is complete, click **Install**.

After the add-on is installed, click **Go Back to Previous Page**. On the **Add-on Instance** tab page, select the corresponding cluster to view the running instance. This indicates that the add-on has been installed on each node in the cluster.

----End

## Upgrading the Add-on

**Step 1** Log in to the CCE console. In the navigation pane, choose **Add-ons**. On the **Add-on Instance** tab page, click **Upgrade** under **autoscaler**.



 **NOTE**

- If the **Upgrade** button is unavailable, the current add-on is already up-to-date and no upgrade is required.
- If the **Upgrade** button is available, click **Upgrade** to upgrade the add-on.
- During the upgrade, the coredns add-on of the original version on cluster nodes will be discarded, and the add-on of the target version will be installed.

**Step 2** In the dialog box displayed, set parameters and upgrade the add-on. For details about the parameters, see the parameter description in [Installing the Add-on](#).

----End

## Uninstalling the Add-on

**Step 1** Log in to the CCE console. In the navigation pane, choose **Add-ons**. On the **Add-on Instance** tab page, select the target cluster and click **Uninstall** under **autoscaler**.

**Step 2** In the dialog box displayed, click **Yes** to uninstall the add-on.

----End

## 17.14.5 nginx-ingress

### Introduction

Kubernetes uses kube-proxy to expose Services and provide load balancing. The implementation is at the transport layer. When it comes to Internet applications, where a bucket-load of information is generated, forwarding needs to be more fine-grained, precisely and flexibly controlled by policies and load balancers to deliver higher performance.

This is where ingresses enter. Ingresses provide application-layer forwarding functions, such as virtual hosts, load balancing, SSL proxy, and HTTP routing, for Services that can be directly accessed outside a cluster.

Kubernetes has officially released the Nginx-based ingress controller. nginx-ingress is an add-on that uses ConfigMaps to store Nginx configurations. The Nginx ingress controller generates Nginx configurations for an ingress and writes the configurations to the pod of Nginx through Kubernetes API. These configurations can be modified and updated by reloading.

The nginx-ingress add-on in CCE is implemented using the open-source community chart and image. CCE does not maintain the add-on. Therefore, it is not recommended that the nginx-ingress add-on be used commercially.

You can visit the [open source community](#) for more information.

 NOTE

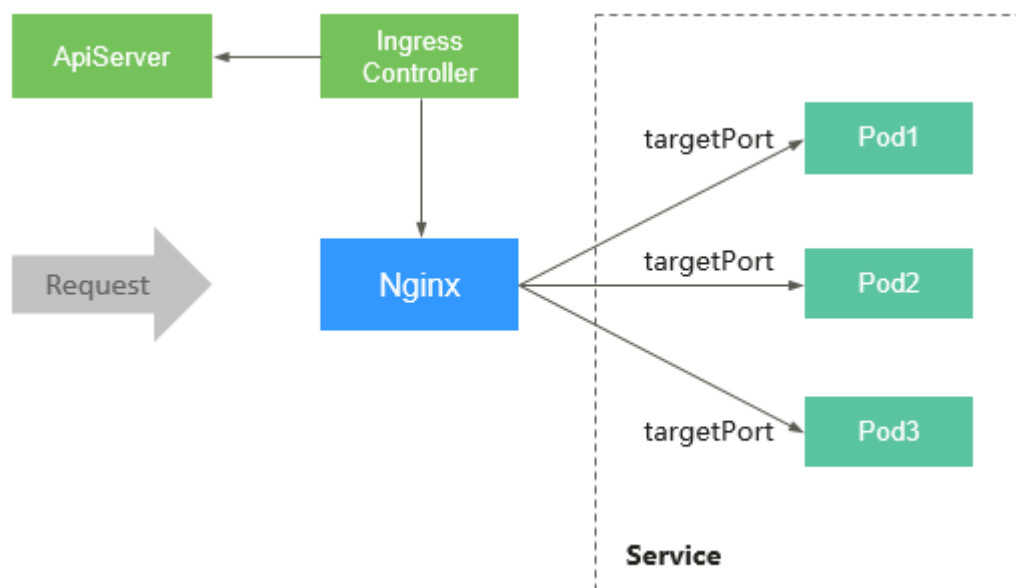
- When installing the add-on, you can add configurations by defining the Nginx configuration. The configurations take effect globally. This parameter is generated by configuring the **nginx.conf** file and affects all managed ingresses. You can search for related parameters in the [ConfigMap](#). If the configured parameters are not included in the options listed in the ConfigMap, the configurations do not take effect.
- Do not manually modify or delete the load balancer and listener that are automatically created by CCE. Otherwise, the workload will be abnormal. If you have modified or deleted them by mistake, you need to uninstall the nginx-ingress add-on and re-install it.

## How nginx-ingress Works

nginx-ingress consists of the ingress object, ingress controller, and Nginx. The ingress controller assembles ingresses into the Nginx configuration file (nginx.conf) and reloads Nginx to make the changed configurations take effect. When it detects that the pod in a Service changes, it dynamically changes the upstream server group configuration of Nginx. In this case, the Nginx process does not need to be reloaded. [Figure 17-147](#) shows how nginx-ingress works.

- An ingress is a group of access rules that forward requests to specified Services based on domain names or URLs. Ingresses are stored in the object storage service etcd and are added, deleted, modified, and queried through APIs.
- The ingress controller monitors the changes of resource objects such as ingresses, Services, endpoints, secrets (mainly TLS certificates and keys), nodes, and ConfigMaps in real time and automatically performs operations on Nginx.
- Nginx implements load balancing and access control at the application layer.

**Figure 17-147** Working principles of nginx-ingress



## Constraints

- **kubernetes.io/ingress.class: "nginx"** must be added to the annotation of the ingress created by calling an API.

## Installing the Add-on

- Step 1** Log in to the CCE console. In the navigation pane, choose **Add-ons**. On the **Add-on Marketplace** tab page, click **Install Add-on** under **nginx-ingress**.
- Step 2** On the **Install Add-on** page, select the cluster and the add-on version, and click **Next: Configuration**.
- Step 3** In the **Configuration** step, set the parameters listed in [Table 17-135](#). Parameters marked with an asterisk (\*) are mandatory.

**Table 17-135** nginx-ingress add-on parameters

Parameter	Description
Add-on Specifications	Select add-on specifications based on service requirements. You can customize resource specifications.
Instances	Number of pods that will be created to match the selected add-on specifications.
Container	<p>CPU and memory quotas of the container allowed for the selected add-on specifications.</p> <p><b>NOTE</b></p> <ul style="list-style-type: none"> <li>• Ensure that there are sufficient nodes in the cluster. If not, the add-on instance cannot be scheduled. In this case, you need to reinstall the add-on.</li> <li>• The request value must be less than or equal to the limit value. Otherwise, the creation fails.</li> <li>• You are advised to set the request value to be the same as the limit value. If nodes are insufficient, containers whose request value is less than the limit value are preferentially cleared.</li> <li>• For details about the performance results of different configurations, see the <a href="#">Nginx performance test report</a>.</li> </ul>
ConfigMap Config	<p>The configuration takes effect globally. This parameter is generated by configuring the <b>nginx.conf</b> file and affects all managed ingresses. You can search for related parameters in the <a href="#">ConfigMap</a>. If your configuration is not included in the options listed in the ConfigMap, your configuration will not take effect.</p> <ul style="list-style-type: none"> <li>• <b>worker-processes</b>: number of worker processes when Nginx provides web services for external systems. The default value is <b>auto</b>.</li> <li>• <b>max-worker-connections</b>: maximum number of concurrent connections of a single worker process. The default value is <b>16384</b>.</li> <li>• <b>keep-alive</b>: timeout interval for a keep-alive connection, in unit of seconds. The default value is <b>75</b>.</li> </ul>

Parameter	Description
Custom Header	By default, Nginx filters out custom headers. This parameter allows you to redefine or add request headers to be sent to backend servers.
Default backend enabled	The nginx-ingress add-on provides the 404 backend service by default. If you need to customize the 404 backend service, enter a value in format of <namespace/serviceName>.
Elastic Load Balancer	<p>You can select an existing public or private network load balancer. This function enables the traffic from a public or private network to be forwarded to the Service backing the add-on.</p> <p><b>Once this parameter is set, do not modify the configuration on the ELB console. Otherwise, the Service will be abnormal.</b> If you have modified the configuration, uninstall the add-on and reinstall it.</p> <p><b>NOTE</b></p> <ul style="list-style-type: none"> <li>• Ensure that the load balancer you select or create is in the same VPC as the cluster and routes requests over the Internet.</li> <li>• The load balancer has at least two listeners, and ports 80 and 443 are not occupied by listeners.</li> </ul>

**Step 4** After the configuration is complete, click **Install**. After the add-on is installed, click **Back to Add-on List**.

**Step 5** On the **Add-on Instance** tab page, select the corresponding cluster. If the installed add-on is displayed and in running state, it has been successfully installed in the current cluster.

----End

## Upgrading the Add-on

**Step 1** Log in to the CCE console. In the navigation pane, choose **Add-ons**. On the **Add-on Instance** tab page, click **Upgrade** under **nginx-ingress**.

### NOTE

- If the **Upgrade** button is not available, the current add-on is already up-to-date and no upgrade is required.
- During the upgrade, the nginx-ingress add-on of the original version on cluster nodes will be discarded, and the add-on of the target version will be installed.
- After the upgrade, nginx-ingress will be restarted. Related services may be interrupted. Therefore, you are advised to upgrade the add-on during off-peak hours.

**Step 2** On the **Basic Information** page, select the add-on version and click **Next**.

**Step 3** In the **Configuration** step, set the parameters listed in [Table 17-135](#). Parameters marked with an asterisk (\*) are mandatory.

**Step 4** After setting the parameters, click **Upgrade** to upgrade the add-on.

----End

## Uninstalling the Add-on

**Step 1** Log in to the CCE console. In the navigation pane, choose **Add-ons**. On the **Add-on Instance** tab page, click **Uninstall** under **nginx-ingress**.

**Step 2** In the dialog box displayed, click **Yes** to uninstall the add-on.

----End

## 17.14.6 metrics-server

From version 1.8 onwards, Kubernetes provides resource usage metrics, such as the container CPU and memory usage, through the Metrics API. These metrics can be directly accessed by users (for example, by using the **kubectl top** command) or used by controllers (for example, Horizontal Pod Autoscaler) in a cluster for decision-making. The specific component is metrics-server, which is used to substitute for heapster for providing the similar functions. heapster has been gradually abandoned since v1.11.

metrics-server is an aggregator for monitoring data of core cluster resources. You can quickly install this add-on on the CCE console.

After metrics-server is installed, you can create an HPA policy on the **Workload Scaling** tab page of the **Auto Scaling** page. For details, see [Creating an HPA Policy for Workload Auto Scaling](#).

The official community project and documentation are available at <https://github.com/kubernetes-sigs/metrics-server>.

## Notes and Constraints

This add-on can be installed only in CCE clusters of v1.13 or later.

## Installing the Add-on

**Step 1** Log in to the CCE console. In the navigation pane, choose **Add-ons**. On the **Add-on Marketplace** tab page, click **Install Add-on** under **metrics-server**.

**Step 2** On the **Install Add-on** page, select the cluster and the add-on version, and click **Next: Configuration**.

**Step 3** Select **Single** or **HA** for **Add-on Specifications**, and click **Install**.

After the add-on is installed, click **Go Back to Previous Page**. On the **Add-on Instance** tab page, select the corresponding cluster to view the running instance. This indicates that the add-on has been installed on each node in the cluster.

----End

## Upgrading the Add-on

**Step 1** Log in to the CCE console. In the navigation pane, choose **Add-ons**. On the **Add-on Instance** tab page, click **Upgrade** under **metrics-server**.

 **NOTE**

- If the **Upgrade** button is not available, the current add-on is already up-to-date and no upgrade is required.
- During the upgrade, the metrics-server add-on of the original version on cluster nodes will be discarded, and the add-on of the target version will be installed.

**Step 2** On the **Basic Information** page, select the add-on version and click **Next**.

**Step 3** Set the parameters by referring to the parameter description in [Installing the Add-on](#) and click **Upgrade**.

----End

## Uninstalling the Add-on

**Step 1** Log in to the CCE console. In the navigation pane, choose **Add-ons**. On the **Add-on Instance** tab page, click **Uninstall** under **metrics-server**.

**Step 2** In the dialog box displayed, click **Yes** to uninstall the add-on.

----End

## 17.14.7 cce-hpa-controller

cce-hpa-controller is a CCE-developed add-on, which can be used to flexibly scale in or out Deployments based on metrics such as CPU usage and memory usage.

After installing this add-on, you can create a CustomedHPA policy on the **Workload Scaling** tab page of the **Auto Scaling** page.

### Main Functions

- Scaling can be performed based on the percentage of the current number of pods.
- The minimum scaling step can be set.
- Different scaling operations can be performed based on the actual metric values.

### Notes and Constraints

- This add-on can be installed only in CCE clusters of v1.15 or later.
- If the cce-hpa-controller version is earlier than 1.2.11, the [prometheus](#) add-on must be installed. If the cce-hpa-controller version is 1.2.11 or later, the add-ons that can provide metrics API, such as metrics-server and prometheus, must be installed. If Prometheus is used, you need to register Prometheus as the service that provides metrics API. For details, see [Providing Resource Metrics](#).

### Installing the Add-on

**Step 1** Log in to the CCE console. In the navigation pane, choose **Add-ons**. On the **Add-on Marketplace** tab page, click **Install Add-on** under **cce-hpa-controller**.

**Step 2** On the **Install Add-on** page, select the cluster and the add-on version, and click **Next: Configuration**.

**Step 3** Select **Single** or **Custom** for **Add-on Specifications**, and click **Install**.

After the add-on is installed, click **Go Back to Previous Page**. On the **Add-on Instance** tab page, select the corresponding cluster to view the running instance. This indicates that the add-on has been installed on each node in the cluster.

----End

## Upgrading the Add-on

**Step 1** Log in to the CCE console. In the navigation pane, choose **Add-ons**. On the **Add-on Instance** tab page, click **Upgrade** under **cce-hpa-controller**.

### NOTE

- If the **Upgrade** button is not available, the current add-on is already up-to-date and no upgrade is required.
- During the upgrade, the cce-hpa-controller add-on of the original version on cluster nodes will be discarded, and the add-on of the target version will be installed.

**Step 2** Set the parameters by referring to the parameter description in [Installing the Add-on](#) and click **Upgrade**.

----End

## Uninstalling the Add-on

**Step 1** Log in to the CCE console. In the navigation pane, choose **Add-ons**. On the **Add-on Instance** tab page, click **Uninstall** under **cce-hpa-controller**.

**Step 2** In the dialog box displayed, click **Yes** to uninstall the add-on.

----End

## 17.14.8 prometheus

### Introduction

Prometheus is an open-source system monitoring and alerting framework. It is derived from Google's borgmon monitoring system, which was created by former Google employees working at SoundCloud in 2012. Prometheus was developed as an open-source community project and officially released in 2015. In 2016, Prometheus officially joined the Cloud Native Computing Foundation, after Kubernetes.

CCE allows you to quickly install Prometheus as an add-on.

Official website of Prometheus: <https://prometheus.io/>

Open source community: <https://github.com/prometheus/prometheus>

### Features

As a next-generation monitoring framework, Prometheus has the following features:

- Powerful multi-dimensional data model
  - a. Time series data is identified by metric name and key-value pair.
  - b. Multi-dimensional labels can be set for all metrics.
  - c. Data models do not require dot-separated character strings.
  - d. Data models can be aggregated, cut, and sliced.
  - e. The double floating-point format is supported. Labels can all be set to unicode.
- Flexible and powerful query statement (PromQL): One query statement supports addition, multiplication, and connection for multiple metrics.
- Easy to manage: The Prometheus server is a separate binary file that can work locally. It does not depend on distributed storage.
- Efficient: Each sampling point occupies only 3.5 bytes, and one Prometheus server can process millions of metrics.
- The pull mode is used to collect time series data, which facilitates local tests and prevents faulty servers from pushing bad metrics.
- Time series data can be pushed to the Prometheus server in push gateway mode.
- Users can obtain the monitored targets through service discovery or static configuration.
- Multiple visual GUIs are available.
- Easy to scale

As collected data may be lost, Prometheus is not applicable if there is a high requirement on accuracy of the collected data. However, Prometheus has great query advantages if it is used to record time series data. In addition, Prometheus is applicable to the microservice architecture.

## Notes and Constraints

This add-on can be installed only in CCE clusters of v1.11 or later.

## Installing the Add-on

- Step 1** Log in to the CCE console. In the navigation pane, choose **Add-ons**. On the **Add-on Marketplace** tab page, click **Install Add-on** under **prometheus**.
- Step 2** On the **Install Add-on** page, select the cluster and the add-on version, and click **Next: Configuration**.
- Step 3** In the **Configuration** step, set the following parameters:



**Table 17-136** prometheus add-on parameters

Parameter	Description
Add-on Specifications	<p>Select add-on specifications based on service requirements. The options are as follows:</p> <ul style="list-style-type: none"> <li>● <b>Demo(&lt;= 100 containers)</b>: The specification type is applicable to the experience and function demonstration environment. In this specification, Prometheus occupies few resources but has limited processing capabilities. You are advised to use this specification when the number of containers in the cluster does not exceed 100.</li> <li>● <b>Small(&lt;= 2000 containers)</b>: You are advised to use this specification when the number of containers in the cluster does not exceed 2,000.</li> <li>● <b>Medium(&lt;= 5000 containers)</b>: You are advised to use this specification when the number of containers in the cluster does not exceed 5,000.</li> <li>● <b>Large(&gt; 5000 containers)</b>: You are advised to use this specification when the number of containers in the cluster exceeds 5,000.</li> </ul>
Instances	Number of pods that will be created to match the selected add-on specifications. The number cannot be modified.
Container	CPU and memory quotas of the container allowed for the selected add-on specifications. The quotas cannot be modified.
Remote Write	<p>Select a value.</p> <ul style="list-style-type: none"> <li>● <b>Local</b>: Data collected by the prometheus add-on is stored only in local data disks.</li> <li>● <b>CIE</b>: Data collected by the prometheus add-on is stored in both local data disks and CIE.</li> <li>● <b>Custom</b>: Data collected by the prometheus add-on is stored in both local data disks and a custom remote end. The remote end address and HTTPS authentication information need to be obtained from third-party services.</li> </ul>
Monitoring Data Retention Period	Number of days for storing customized monitoring data. The default value is 15 days.

Parameter	Description
Storage	<p>Set the following parameters as prompted:</p> <ul style="list-style-type: none"> <li>• <b>Type:</b> EVS is supported.</li> <li>• <b>AZ:</b> Set this parameter based on the site requirements. An AZ is a physical region where resources use independent power supply and networks. AZs are physically isolated but interconnected through an internal network.</li> <li>• <b>Disk Type:</b> Common I/O, high I/O, and ultra-high I/O are supported. For details about the comparison among these disk types, see <a href="#">System Disks and Data Disks</a>.</li> <li>• <b>Capacity:</b> Enter the storage capacity based on service requirements. The default value is 10 GB.</li> </ul> <p><b>NOTE</b> If a PVC already exists in the namespace monitoring, the configured storage will be used as the storage source.</p>

**Step 4** Click **Install**.

After the add-on is installed, click **Go Back to Previous Page**. On the **Add-on Instance** tab page, select the corresponding cluster to view the running instance. This indicates that the add-on has been installed on each node in the cluster.

**Step 5** In the navigation pane on the left, choose **Add-ons**. On the **Add-on Instance** tab page, click **prometheus** to view details about the add-on instance.

----End

## Providing Resource Metrics

Resource metrics of containers and nodes, such as CPU and memory usage, can be obtained through the Kubernetes Metrics API. Resource metrics can be directly accessed, for example, by using the **kubectl top** command, or used by HPA or CustomedHPA policies for auto scaling.

The prometheus add-on can provide the Kubernetes Metrics API that is disabled by default. To enable the API, create the following APIService object:

```
apiVersion: apiregistration.k8s.io/v1
kind: APIService
metadata:
  labels:
    app: custom-metrics-apiserver
    release: cceaddon-prometheus
  name: v1beta1.metrics.k8s.io
spec:
  group: metrics.k8s.io
  groupPriorityMinimum: 100
  insecureSkipTLSVerify: true
  service:
    name: custom-metrics-apiserver
    namespace: monitoring
    port: 443
  version: v1beta1
  versionPriority: 100
```

You can save the object as a file, name it **metrics-apiservice.yaml**, and run the following command:

```
kubectl create -f metrics-apiservice.yaml
```

Run the **kubectl top** command. If the following information is displayed, the Metrics API can be accessed:

```
# kubectl top pod -n monitoring
NAME                                CPU(cores)  MEMORY(bytes)
cceaddon-prometheus-kube-state-metrics-7b77694f48-zc9pl  4m          16Mi
cceaddon-prometheus-node-exporter-4jvwv                1m          16Mi
cceaddon-prometheus-node-exporter-85zl4                2m          39Mi
cceaddon-prometheus-node-exporter-qbrmb                0m          15Mi
cceaddon-prometheus-operator-659547567d-j6484         0m          48Mi
custom-metrics-apiserver-d4f556ff9-l2j2m              38m         44Mi
grafana-78f9966c99-xprkx                              0m          25Mi
prometheus-0                                           18m         706Mi
```

## Upgrading the Add-on

**Step 1** Log in to the CCE console. In the navigation pane, choose **Add-ons**. On the **Add-on Instance** tab page, click **Upgrade** under **prometheus**.

### NOTE

- If the **Upgrade** button is not available, the current add-on is already up-to-date and no upgrade is required.
- During the upgrade, the prometheus add-on of the original version on cluster nodes will be discarded, and the add-on of the target version will be installed.

**Step 2** On the **Basic Information** page, select the add-on version and click **Next**.

**Step 3** Set the parameters by referring to the parameter description in [Installing the Add-on](#) and click **Upgrade**.

----End

## Uninstalling the Add-on

**Step 1** Log in to the CCE console. In the navigation pane, choose **Add-ons**. On the **Add-on Instance** tab page, click **Uninstall** under **prometheus**.

**Step 2** In the dialog box displayed, click **Yes** to uninstall the add-on.

----End

## Reference

- For details about the Prometheus concepts and configurations, see the [Prometheus Official Documentation](#).
- For details about how to install Node Exporter, see the [node\\_exporter GitHub](#).
- For details about how to send Slack messages, see [Incoming Webhooks](#).

## 17.14.9 gpu-beta

### Introduction

gpu-beta is a device management add-on that supports GPUs in containers. It supports only NVIDIA Tesla drivers.

### Notes and Constraints

- This add-on is available only in certain regions.
- This add-on can be installed only in CCE clusters of v1.11 or later.
- If GPU nodes are used in the cluster, the gpu-beta add-on must be installed.
- The driver to be downloaded must be a **.run** file.
- Only Tesla drivers are supported, not GRID drivers.

---

#### NOTICE

- If the download link is a public network address, for example, **https://us.download.nvidia.com/tesla/396.37/NVIDIA-Linux-x86\_64-396.37.run**, bind an EIP to each GPU node. For details about how to obtain the driver link, see [Obtaining the Driver Link from Public Network](#).
  - If the download link is an OBS URL, you do not need to bind an EIP to GPU nodes. For details about how to obtain the driver link, see [Obtaining the Driver Link from OBS](#).
  - Ensure that the NVIDIA driver version matches the GPU node.
  - After the driver version is changed, restart the node for the change to take effect.
- 

### Installing the Add-on

**Step 1** Log in to the CCE console. In the navigation pane, choose **Add-ons**. On the **Add-on Marketplace** tab page, click **Install Add-on** under **gpu-beta**.

**Step 2** On the **Install Add-on** page, select the cluster and the add-on version, and click **Next: Configuration**.

**Step 3** In the **Configuration** step, enter the link to download the NVIDIA driver.

**Step 4** Click **Install**.

After the add-on is installed, click **Go Back to Previous Page**. On the **Add-on Instance** tab page, select the corresponding cluster to view the running instance. This indicates that the add-on has been installed on each GPU node in the cluster.

----End

### Verifying the Add-on

After the add-on is installed, run the **nvidia-smi** command on the GPU node and the container that schedules GPU resources to verify the availability of the GPU device and driver.

- GPU node:  
# If the add-on version is earlier than 2.0.0, run the following command:  
`cd /opt/cloud/cce/nvidia/bin && ./nvidia-smi`  
  
# If the add-on version is 2.0.0 or later and the driver installation path is changed, run the following command:  
`cd /usr/local/nvidia/bin && ./nvidia-smi`
- Container:  
`cd /usr/local/nvidia/bin && ./nvidia-smi`

If GPU information is returned, the device is available and the add-on has been installed.

```
+-----+
| NVIDIA-SMI 440.118.02    Driver Version: 440.118.02    CUDA Version: 10.2    |
+-----+-----+
| GPU  Name           Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp   Perf    Pwr:Usage/Cap|      Memory-Usage | GPU-Util  Compute M. |
+-----+-----+-----+-----+
|   0   Tesla V100-SXM2...    Off   | 00000000:21:01.0 Off  |      0          0     |
| N/A   31C    P0      23W / 300W |  0MiB / 16160MiB |   0%      Default   |
+-----+-----+-----+-----+

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

## Obtaining the Driver Link from Public Network

- Step 1** Log in to the CCE console.
- Step 2** Click **Create Node** and select the GPU node to be created in the **Specifications** area. The GPU card model of the node is displayed in the lower part of the page.
- Step 3** Visit <https://www.nvidia.com/Download/Find.aspx?lang=en>.
- Step 4** Select the driver information on the **NVIDIA Driver Downloads** page, as shown in **Figure 17-148**. **Operating System** must be **Linux 64-bit**.

Figure 17-148 Setting parameters

## NVIDIA Driver Downloads

Official Advanced Driver Search | NVIDIA

<b>Product Type:</b> Data Center / Tesla	<b>Operating System:</b> Linux 64-bit
<b>Product Series:</b> V-Series	<b>CUDA Toolkit:</b> 11.6
<b>Product:</b> Tesla V100	<b>Language:</b> English (US)
	<b>Recommended/Beta:</b> All ?

SEARCH

Click the Search button to perform your search.

**Step 5** After confirming the driver information, click **SEARCH**. A page is displayed, showing the driver information, as shown in [Figure 17-149](#). Click **DOWNLOAD**.

Figure 17-149 Driver information

## TESLA DRIVER FOR LINUX X64

Version:	396.37
Release Date:	2018.7.9
Operating System:	Linux 64-bit
CUDA Toolkit:	9.2
Language:	Chinese (Simplified)
File Size:	81.88 MB

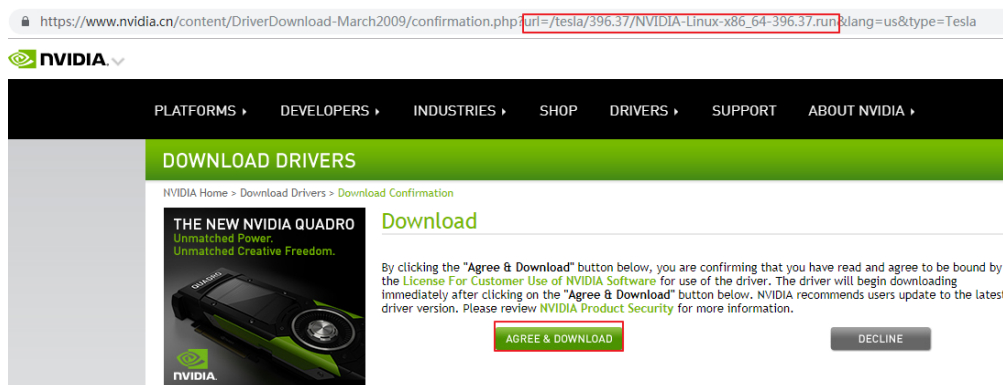
DOWNLOAD

RELEASE HIGHLIGHTS	SUPPORTED PRODUCTS	ADDITIONAL INFORMATION
<p>What's New</p> <ul style="list-style-type: none"><li>Various security issues were addressed, for additional details on the med-high severity issues please review <a href="#">NVIDIA Product Security</a> for more information</li></ul>		

**Step 6** Obtain the driver link in either of the following ways:

- Method 1: As shown in [Figure 17-150](#), find `url=/tesla/396.37/NVIDIA-Linux-x86_64-396.37.run` in the browser address box. Then, supplement it to obtain the driver link [https://us.download.nvidia.com/tesla/396.37/NVIDIA-Linux-x86\\_64-396.37.run](https://us.download.nvidia.com/tesla/396.37/NVIDIA-Linux-x86_64-396.37.run). By using this method, you must bind an EIP to each GPU node.
- Method 2: As shown in [Figure 17-150](#), click **AGREE & DOWNLOAD** to download the driver. Then, upload the driver to OBS and record the OBS URL. By using this method, you do not need to bind an EIP to GPU nodes.

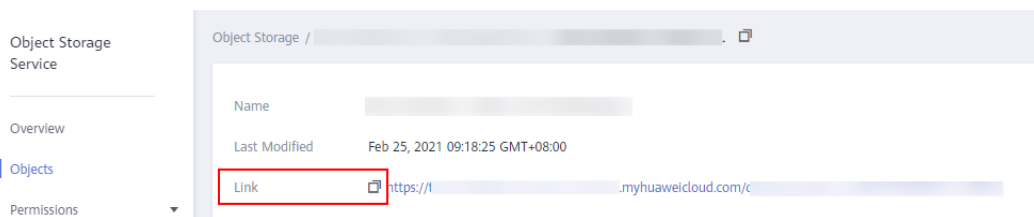
Figure 17-150 Obtaining the link



----End

## Obtaining the Driver Link from OBS

- Step 1** Upload the driver to OBS and set the driver file to public read.
- Step 2** In the navigation pane on the OBS console, select **Object Storage**.
- Step 3** In the bucket list, click a bucket name, and then the **Overview** page of the bucket is displayed.
- Step 4** In the navigation pane, choose **Objects**.
- Step 5** Select the target object and copy the driver link on the object details page.



----End

## Uninstalling the Add-on

- Step 1** Log in to the CCE console. In the navigation pane, choose **Add-ons**. On the **Add-on Instance** tab page, select the cluster and click **Uninstall** under **gpu-beta**.
- Step 2** In the dialog box displayed, click **Yes** to uninstall the add-on.

### NOTE

The driver will not be uninstalled during gpu-beta add-on uninstall. If the driver is reinstalled, you must restart all GPU nodes.

----End

## 17.14.10 huawei-npu

### Introduction

huawei-npu is a management add-on for Huawei NPU devices in containers.

## Notes and Constraints

- This add-on can be installed in CCE clusters of v1.13 or later.
- If Ascend-accelerated nodes are used in the cluster, the huawei-npu add-on must be installed.

## Installing the Add-on

**Step 1** Log in to the CCE console. In the navigation pane, choose **Add-ons**. On the **Add-on Marketplace** tab page, click **Install Add-on** under **huawei-npu**.

**Step 2** On the **Install Add-on** page, select the cluster and the add-on version, and click **Next: Configuration**.

**Step 3** Click **Install** to directly install the add-on. Currently, the huawei-npu add-on has no configurable parameters.

After the add-on is installed, click **Go Back to Previous Page**. On the **Add-on Instance** tab page, select the corresponding cluster to view the running instance. This indicates that the add-on has been installed on each node in the cluster.

----End

## Upgrading the Add-on

**Step 1** Log in to the CCE console. In the navigation pane, choose **Add-ons**. On the **Add-on Instance** tab page, click **Upgrade** under **huawei-npu**.

### NOTE

- If the **Upgrade** button is not available, the current add-on is already up-to-date and no upgrade is required.
- During the upgrade, the huawei-npu add-on of the original version on cluster nodes will be discarded, and the add-on of the target version will be installed.

**Step 2** On the **Basic Information** page, select the add-on version and click **Next**.

**Step 3** Click **Upgrade**.

----End

## Uninstalling the Add-on

**Step 1** Log in to the CCE console. In the navigation pane, choose **Add-ons**. On the **Add-on Instance** tab page, click **Uninstall** under **huawei-npu**.

**Step 2** In the dialog box displayed, click **Yes** to uninstall the add-on.

----End

## 17.14.11 node-local-dns

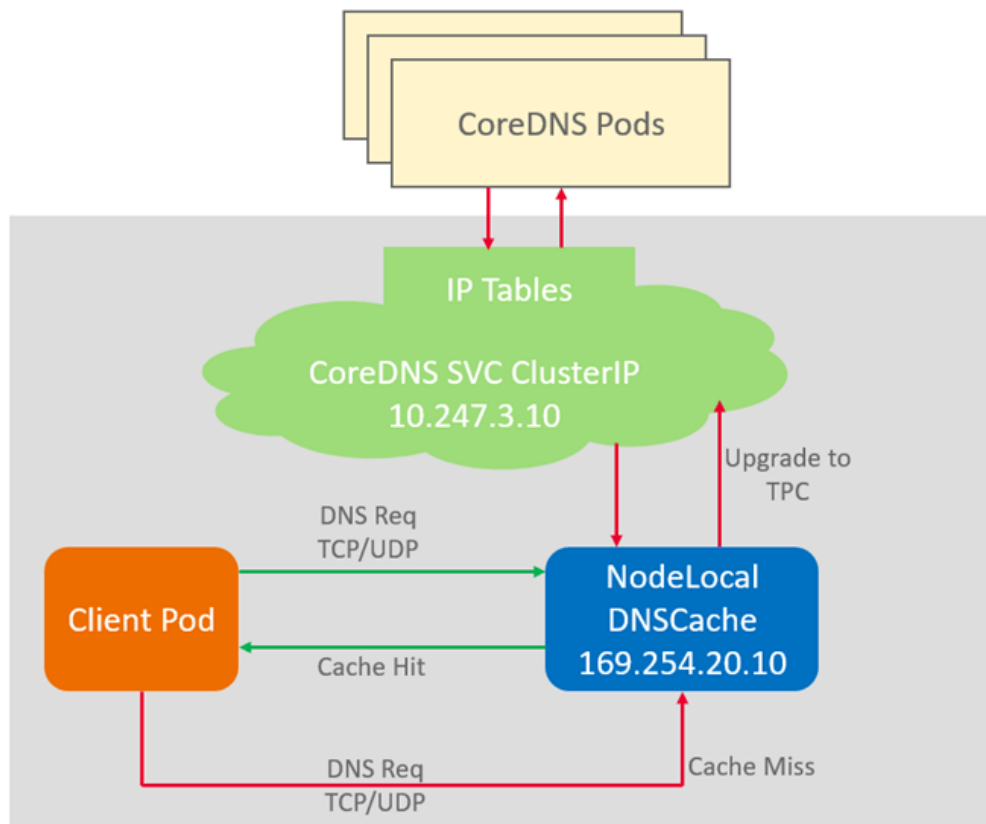
### Introduction

NodeLocal DNSCache improves cluster DNS performance by running DNS cache proxies as DaemonSets on cluster nodes.



Open source community: <https://github.com/kubernetes/dns>

Figure 17-151 NodeLocal DNSCache query path



## Notes and Constraints

- This feature is available only to clusters of v1.19 and later.

## Installing the Add-on

**Step 1** Log in to the CCE console and access the cluster details page. Choose **Add-ons** in the navigation pane, locate **node-local-dns** on the right, and click **Install**.

**Step 2** On the **Install Add-on** page, select the add-on specifications and set related parameters.

**enable\_dnsconfig\_admission**: whether to enable auto injection. Defaults to **false**. Value **true** enables auto injection.

**Step 3** Click **Install**.

----End

## Helpful Links

[Using NodeLocal DNSCache to Improve DNS Performance](#)

## 17.15 Auto Scaling

### 17.15.1 Overview

Auto scaling is a service that automatically and economically adjusts service resources based on your service requirements and configured policies.

#### Context

More and more applications are developed based on Kubernetes. It becomes increasingly important to quickly scale out applications on Kubernetes to cope with service peaks and to scale in applications during off-peak hours to save resources and reduce costs.

In a Kubernetes cluster, auto scaling involves pods and nodes. A pod is an application instance. Each pod contains one or more containers and runs on a node (VM). If a cluster does not have sufficient nodes to run new pods, you need to add nodes to the cluster to ensure service running.

In CCE, auto scaling is used for online services, large-scale computing and training, deep learning GPU or shared GPU training and inference, periodic load changes, and many other scenarios.

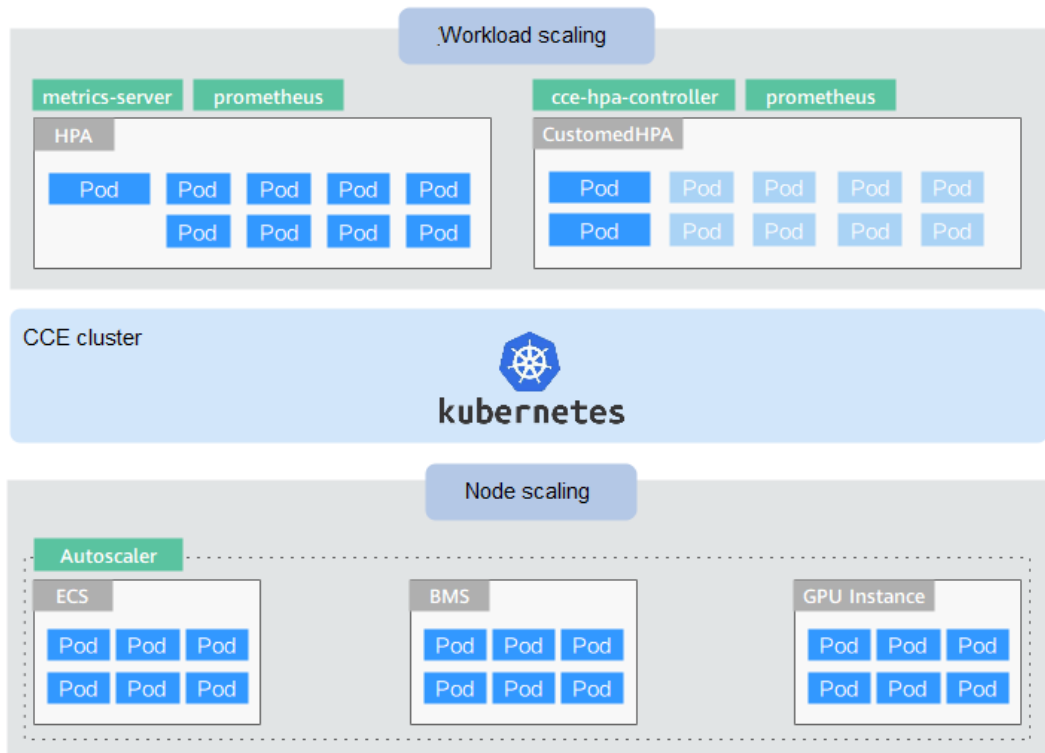
#### Auto Scaling in CCE

**CCE supports auto scaling for workloads and nodes.**

- **Workload scaling:** Auto scaling at the scheduling layer to change the scheduling capacity of workloads. For example, you can use the HPA, a scaling component at the scheduling layer, to adjust the number of replicas of an application. Adjusting the number of replicas changes the scheduling capacity occupied by the current workload, thereby enabling scaling at the scheduling layer.
- **Node scaling:** Auto scaling at the resource layer. When the planned cluster nodes cannot allow workload scheduling, ECS resources are provided to support scheduling.

Workload scaling and node scaling can work separately or together. For details, see [Using HPA and CA for Auto Scaling of Workloads and Nodes](#).

#### Components



Workload scaling components are described as follows:

Table 17-137 Workload scaling components

Type	Component Name	Component Description	Reference
HPA	<a href="#">metrics-server</a>	A built-in component of Kubernetes, which enables horizontal scaling of pods. It adds the application-level cooldown time window and scaling threshold functions based on the HPA.	<a href="#">Creating an HPA Policy for Workload Auto Scaling</a>
Customed HPA	<a href="#">cce-hpa-controller</a>	An enhanced auto scaling feature, used for auto scaling of Deployments based on metrics (CPU usage and memory usage) or at a periodic interval (a specific time point every day, every week, every month, or every year).	<a href="#">Creating a CustomedHPA Policy for Workload Auto Scaling</a>

Type	Component Name	Component Description	Reference
	<a href="#">prometheus</a>	An open-source system monitoring and alarm framework, which collects public metrics (CPU usage and memory usage) of kubelet in the Kubernetes cluster.	

Node scaling components are described as follows:

Table 17-138 Node scaling components

Component Name	Component Description	Application Scenario	Reference
<a href="#">autoscaler</a>	An open-source Kubernetes component for horizontal scaling of nodes, which is optimized in terms of scheduling and auto scaling capabilities.	Online services, deep learning, and large-scale computing with limited resource budgets	<a href="#">Creating a Node Scaling Policy</a>

## 17.15.2 Scaling a Workload

### 17.15.2.1 Workload Scaling Mechanisms

CCE supports HPA and CustomedHPA policies for workload scaling. The following table describes the differences between these two types of policies.

Table 17-139 Comparison between HPA and CustomedHPA policies

Item	HPA Policy	CustomedHPA Policy
Implementation	Kubernetes Horizontal Pod Autoscaling	Enhanced auto scaling capabilities
Rules	Scales Deployments based on <b>metrics</b> (CPU usage and memory usage).	Scales Deployments based on <b>metrics</b> (CPU usage and memory usage) or at a <b>periodic</b> interval (a specific time point every day, every week, every month, or every year).

Item	HPA Policy	Customized HPA Policy
Enhancement	Adds the application-level cooldown time window and scaling threshold functions based on the Kubernetes HPA.	<p><b>Metric-based:</b></p> <ul style="list-style-type: none"> <li>Scaling can be performed based on the percentage of the current number of pods.</li> <li>The minimum scaling step can be set. Scaling can be performed step by step.</li> <li>Different scaling operations can be performed based on the actual metric values.</li> </ul> <p><b>Periodic:</b></p> <p>You can select a specific time point every day, every week, every month, or every year or a period as the trigger time.</p>

 **NOTE**

**Scaling policy priority:** If you do not manually adjust the number of pods, auto scaling policies will take effect for resource scheduling. If **manual scaling** is triggered, auto scaling policies will be temporarily invalid.

## How HPA Works

HPA is a controller that controls horizontal pod scaling. HPA periodically checks the pod metrics, calculates the number of replicas required to meet the target values configured for HPA resources, and then adjusts the value of the **replicas** field in the target resource object (such as a Deployment).

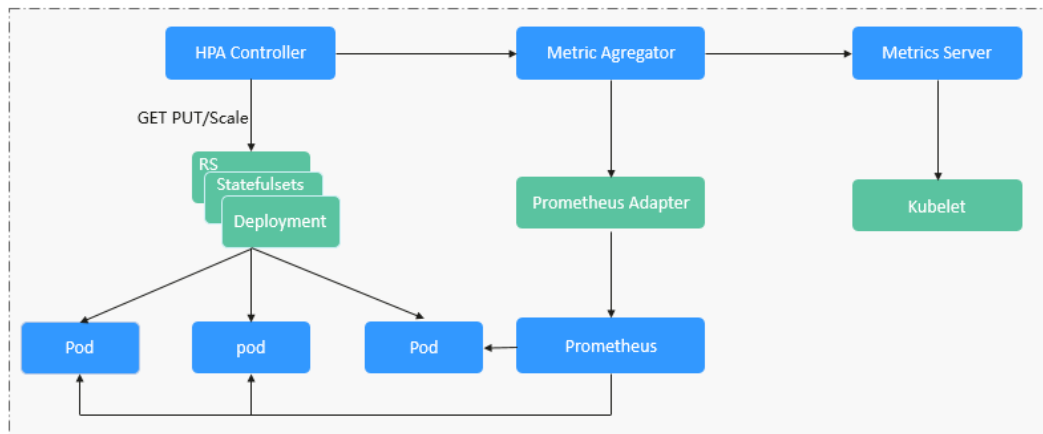
A prerequisite for auto scaling is that your container running data can be collected, such as number of cluster nodes/pods, and CPU and memory usage of containers. Kubernetes does not provide such monitoring capabilities itself. You can use extensions to monitor and collect your data. CCE integrates **Prometheus** and **Metrics Server** to realize such capabilities:

- **Prometheus** is an open-source monitoring and alarming framework that can collect multiple types of metrics. Prometheus has been a standard monitoring solution of Kubernetes.
- **Metrics Server** is a cluster-wide aggregator of resource utilization data. Metrics Server collects metrics from the Summary API exposed by kubelet. These metrics are set for core Kubernetes resources, such as pods, nodes, containers, and Services. Metrics Server provides a set of standard APIs for external systems to collect these metrics.

HPA can work with Metrics Server to implement auto scaling based on the CPU and memory usage. It can also work with Prometheus to implement auto scaling based on custom monitoring metrics.

**Figure 17-152** shows how HPA works.

Figure 17-152 HPA working process



### Two core modules of HPA:

- Data Source Monitoring

The community provided only CPU- and memory-based HPA at the early stage. With the population of Kubernetes, developers need more custom metrics or monitoring information at the access layer for their own applications, for example, the QPS of the load balancer and the number of online users of the website. In response, the community defines a set of standard metric APIs to provide services externally through these aggregated APIs.

- **metrics.k8s.io** provides monitoring metrics related to the CPU and memory of pods and nodes.
- **custom.metrics.k8s.io** provides custom monitoring metrics related to Kubernetes objects.
- **external.metrics.k8s.io** provides metrics that come from external systems and are irrelevant to any Kubernetes resource metrics.

- Scaling Decision-Making Algorithms

The HPA controller calculates the scaling ratio based on the current metric values and desired metric values using the following formula:

$$\text{desiredReplicas} = \text{ceil}[\text{currentReplicas} \times (\text{currentMetricValue} / \text{desiredMetricValue})]$$

For example, if the current metric value is 200m and the target value is 100m, the desired number of pods will be doubled according to the formula. In practice, pods may be constantly added or reduced. To ensure stability, the HPA controller is optimized from the following aspects:

- Cooldown interval: In v1.11 and earlier versions, Kubernetes introduced the startup parameters **horizontal-pod-autoscaler-downscale-stabilization-window** and **horizontal-pod-autoScaler-upscale-stabilization-window** to indicate the cooldown intervals after a scale-in and scale-out, respectively, in which no scaling operation will not be performed. In versions later than v1.14, the scheduling queue is introduced to store all decision-making suggestions detected within a period of time. Then, the system makes decisions based on all valid decision-making suggestions to minimize changes of the desired number of replicas to ensure stability.

- Tolerance: It can be considered as a buffer zone. If the pod number changes can be tolerated, the number of pods remains unchanged.

Use the formula:  $\text{ratio} = \text{currentMetricValue} / \text{desiredMetricValue}$

When  $|\text{ratio} - 1.0| \leq \text{tolerance}$ , scaling will not be performed.

When  $|\text{ratio} - 1.0| > \text{tolerance}$ , the desired value is calculated using the formula mentioned above.

The default value is 0.1 in the current community version.

The HPA performs scaling based on metric thresholds. Common metrics include the CPU and memory usage. You can also set custom metrics, such as the QPS and number of connections, to trigger scaling. However, metric-based scaling brings in latency of minutes generated during data collection, determination, and scaling phases. Such latency may cause high CPU usage and slow response. To solve this problem, CCE allows you to configure scheduled policies to scale resources regularly for applications with periodic changes.

### 17.15.2.2 Creating an HPA Policy for Workload Auto Scaling

Horizontal Pod Autoscaling (HPA) in Kubernetes implements horizontal scaling of pods. In a CCE HPA policy, you can configure different cooldown time windows and scaling thresholds for different applications based on the Kubernetes HPA.

#### Prerequisites

To use HPA policies, you need to install add-ons that can provide the metrics API, such as metrics-server and prometheus.

#### Notes and Constraints


- HPA policies can be created only for clusters of v1.13 or later.
- Only one policy can be created for each workload. That is, if you have created an HPA policy, you cannot create **Customized HPA policies** or other HPA policies for the workload. You can delete the created HPA policy and create a new one.
- For clusters earlier than v1.19.10, if an HPA policy is used to scale out a workload with EVS volumes mounted, the existing pods cannot be read or written when a new pod is scheduled to another node.

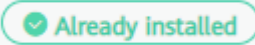
For clusters of v1.19.10 and later, if an HPA policy is used to scale out a workload with EVS volume mounted, a new pod cannot be started because EVS disks cannot be attached.

#### Procedure

**Step 1** Log in to the CCE console. In the navigation pane, choose **Auto Scaling**. On the **Workload Scaling** tab page, click **Create HPA Policy**.

**Step 2** In the **Check Add-ons** step:

- If  is displayed next to the add-on name, click **Install**, set add-on parameters as required, and click **Install** to install the add-on.

- If  is displayed next to the add-on name, the add-on has been installed.

**Step 3** After the required add-ons have been installed, click **Next: Policy configuration**.

 **NOTE**

If the add-ons have been installed, after you click **Create HPA Policy**, you will directly land on the second step to configure the policy. The first step (checking the add-ons) has been completed almost instantly.

**Step 4** Set policy parameters by referring to [Table 17-140](#).

**Table 17-140** HPA policy parameters

Parameter	Description
Policy Name	Name of the policy to be created. Set this parameter as required.
Cluster Name	Cluster to which the workload belongs.
Namespace	Namespace to which the workload belongs.
Associated Workload	Workload with which the HPA policy is associated.
Pod Range	Minimum and maximum numbers of pods. When a policy is triggered, the workload pods are scaled within this range.
Cooldown Period	Interval between a scale-in and a scale-out. The unit is minute. <b>The interval cannot be shorter than 1 minute.</b> <b>This parameter is available only for clusters of v1.15 and later. It is not supported in clusters of v1.13 or earlier.</b> This parameter indicates the interval between consecutive scaling operations. The cooldown period ensures that a scaling operation is initiated only when the previous one is completed and the system is running stably.



Parameter	Description
Rules	<p>Policy rules can be based on system metrics.</p> <p><b>System metrics</b></p> <ul style="list-style-type: none"> <li>• <b>Metric:</b> You can select <b>CPU usage</b> or <b>Memory usage</b>.</li> </ul> <p><b>NOTE</b> Usage = CPUs or memory used by pods/Requested CPUs or memory.</p> <ul style="list-style-type: none"> <li>• <b>Expected Value:</b> Enter the expected average resource usage. This parameter indicates the expected value of the selected metric. The number of new pods required (rounded up) = Current metric value/Expected value x Number of current pods</li> <li>• <b>Threshold:</b> Enter the scaling thresholds. If the metric value is greater than the scale-in threshold and less than the scale-out threshold, no scaling is triggered. <b>This parameter is supported only in clusters of v1.15 or later.</b></li> </ul> <p>You can click <b>Add Rule</b> again to add more scaling policies.</p> <p><b>NOTE</b> When calculating the number of pods to be added or reduced, the HPA policy uses the maximum metrics values in the last 5 minutes.</p>

**Step 5** After the configuration is complete, click **Create**. If the system displays a message indicating that the request to create workload policy \*\*\* is successfully submitted, click **Back to Workload Scaling**.

**Step 6** On the **Workload Scaling** tab page, you can view the newly created HPA policy.

**Figure 17-153** Creating an HPA policy



----End

### 17.15.2.3 Creating a CustomedHPA Policy for Workload Auto Scaling

A CustomedHPA policy scales Deployments based on metrics (such as CPU usage and memory usage) or at a periodic interval (a specific time point every day, every week, every month, or every year). This type of policy is a CCE-enhanced auto scaling capability.

Supported functions:

- Scaling can be performed based on the percentage of the current number of pods.
- The minimum scaling step can be set.

- Different scaling operations can be performed based on the actual metric values.

## Prerequisites

To use a CustomedHPA policy, you must install the [cce-hpa-controller](#) add-on. If the cce-hpa-controller version is earlier than 1.2.11, the [prometheus](#) add-on must be installed. If the cce-hpa-controller version is 1.2.11 or later, the add-ons that can provide metrics API, such as metrics-server and prometheus, must be installed.

## Notes and Constraints


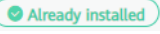
- CustomedHPA policies can be created only for clusters of v1.15 or later.
- Only one policy can be created for each workload. If you have created an HPA policy, you cannot create a CustomedHPA policy or other HPA policies for the workload. You can delete the created HPA policy and create a new one.
- For clusters earlier than v1.19.10, if an HPA policy is used to scale out a workload with EVS volumes mounted, the existing pods cannot be read or written when a new pod is scheduled to another node.

For clusters of v1.19.10 and later, if an HPA policy is used to scale out a workload with EVS volume mounted, a new pod cannot be started because EVS disks cannot be attached.

## Procedure

**Step 1** Log in to the CCE console. In the navigation pane, choose **Auto Scaling**. On the **Workload Scaling** tab page, click **Create CustomedHPA Policy**.

**Step 2** In the **Check Add-ons** step:

- If  **Not installed** is displayed next to the add-on name, click **Install**, set add-on parameters as required, and click **Install** to install the add-on.
- If  **Already installed** is displayed next to the add-on name, the add-on has been installed.

**Step 3** After the required add-ons have been installed, click **Next: Policy configuration**.

### NOTE

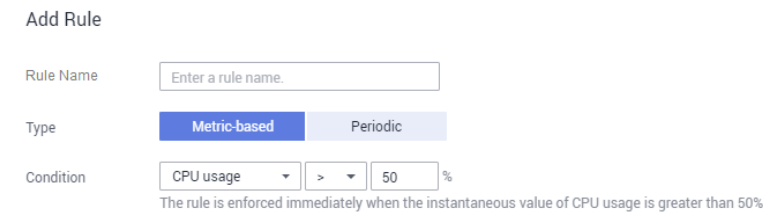
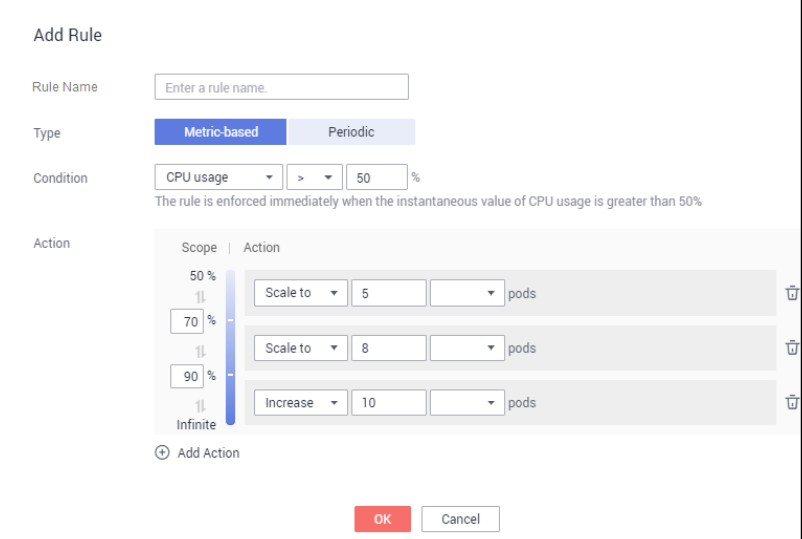
If the add-ons have been installed, after you click **Create CustomedHPA Policy**, you will directly land on the second step to configure the policy. The first step (checking the add-ons) has been completed almost instantly.

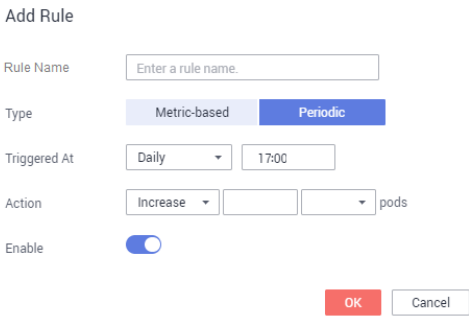
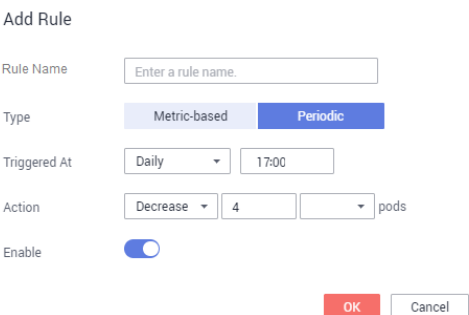
**Step 4** Set policy parameters by referring to [Table 17-141](#).

**Table 17-141** CustomedHPA policy parameters

Parameter	Description
Policy Name	Name of the policy to be created. Set this parameter as required.
Cluster Name	Cluster to which the workload belongs.

Parameter	Description
Namespace	Namespace to which the workload belongs.
Associated Workload	Workload with which the CustomedHPA policy is associated.
Pod Range	Minimum and maximum numbers of pods. When a policy is triggered, the workload pods are scaled within this range.
Cooldown Period	Enter an interval, in minutes. This parameter indicates the interval between consecutive scaling operations. The cooldown period ensures that a scaling operation is initiated only when the previous one is completed and the system is running stably.

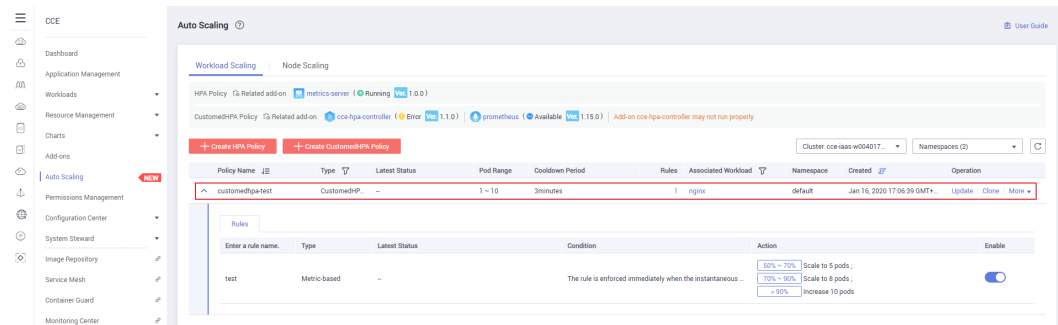
Parameter	Description
Rules	<p>Click <b>Add Rule</b>. In the dialog box displayed, set the following parameters:</p> <ul style="list-style-type: none"> <li>• <b>Name:</b> Enter a custom rule name.</li> <li>• <b>Type:</b> You can select <b>Metric-based</b> or <b>Periodic</b>.</li> </ul> <p><b>Metric-based:</b></p> <ul style="list-style-type: none"> <li>• <b>Condition:</b> Select <b>CPU usage</b> or <b>Memory usage</b>, choose &gt; or &lt;, and enter a percentage. As shown in the following figure, the rule will be executed immediately when the CPU usage is greater than 50%.</li> </ul> <p><b>NOTE</b> Usage = CPUs or memory used by pods/Requested CPUs or memory.</p> <p><b>Figure 17-154</b> Setting a trigger condition</p>  <p>The rule is enforced immediately when the instantaneous value of CPU usage is greater than 50%</p> <ul style="list-style-type: none"> <li>• <b>Action:</b> Set an action to be performed when the trigger condition is met. Multiple actions can be added. As shown below, when the CPU usage exceeds 50%, the number of pods is scaled out to 5. When the CPU usage exceeds 70%, the number of pods is scaled out to 8. When the CPU usage exceeds 90%, the number of pods is scaled out to 18 (adding 10 more pods). These rules also work for scale-in operations.</li> </ul> <p><b>Figure 17-155</b> Action</p>  <p>OK Cancel</p>

Parameter	Description
	<ul style="list-style-type: none"> <li>• <b>Enable:</b> Enable or disable the policy rule.</li> </ul> <p><b>Periodic:</b></p> <ul style="list-style-type: none"> <li>• <b>Triggered At:</b> You can select a specific time point every day, every week, every month, or every year. As shown below, the rule is triggered at 17:00 every day.</li> </ul> <p><b>Figure 17-156</b> Periodic triggering (Daily)</p>  <p><b>Figure 17-157</b> Setting an action for periodic triggering</p>  <ul style="list-style-type: none"> <li>• <b>Action:</b> Set an action to be performed when the <b>Triggered At</b> value is reached. As shown below, four pods will be reduced at 17:00 every day.</li> </ul> <p><b>Figure 17-157</b> Setting an action for periodic triggering</p> <ul style="list-style-type: none"> <li>• <b>Enable:</b> Enable or disable the policy rule.</li> </ul> <p>Click <b>OK</b>. You can view the added rule in the policy rule list and enable, disable, edit, or delete the rule.</p> <p>You can click <b>Add Rule</b> below the policy rule list to add multiple rules.</p>

**Step 5** After the configuration is complete, click **Create**. If the system displays a message indicating that the request to create workload policy \*\*\* is successfully submitted, click **Back to Workload Scaling**.

**Step 6** On the **Workload Scaling** tab page, you can view the newly created CustomedHPA policy.

**Figure 17-158** Creating a CustomedHPA policy



----End

## 17.15.2.4 Managing Workload Scaling Policies

### Scenario

After an HPA or CustomedHPA policy is created, you can update, clone, edit, and delete the policy, as well as edit the YAML file.

### Checking an HPA Policy

You can view the rules, status, and events of an HPA policy and handle exceptions based on the error information displayed.

- Step 1** Log in to the CCE console. In the navigation pane, choose **Auto Scaling**. On the **Workload Scaling** tab page, click in front of the target policy.
- Step 2** In the expanded area, you can view the **Rules**, **Status**, and **Events** tab pages. If the policy is abnormal, locate and rectify the fault based on the error information.

#### NOTE

You can also view the created HPA policy on the workload details page. Log in to the CCE console, choose **Workloads > Deployments** or **Workloads > StatefulSets** in the navigation pane, and choose **More > Scaling** in the **Operation** column. On the workload details page, click the **Scaling** tab. You can see the **Auto Scaling-HPA / CustomedHPA** pane, as well as the HPA policy you have configured on the **Auto Scaling** page.

**Table 17-142** Event types and names


Event Type	Event Name	Description
Normal	SuccessfulRescale	The scaling is performed successfully.
Abnormal	InvalidTargetRange	Invalid target range.
	InvalidSelector	Invalid selector.
	FailedGetObjectMetric	Objects fail to be obtained.
	FailedGetPodsMetric	Pods fail to be obtained.
	FailedGetResourceMetric	Resources fail to be obtained.

Event Type	Event Name	Description
	FailedGetExternalMetric	External metrics fail to be obtained.
	InvalidMetricSourceType	Invalid metric source type.
	FailedConvertHPA	HPA conversion failed.
	FailedGetScale	The scale fails to be obtained.
	FailedComputeMetricsReplicas	Failed to calculate metric-defined replicas.
	FailedGetScaleWindow	Failed to obtain ScaleWindow.
	FailedRescale	Failed to scale the service.

----End

## Viewing a CustomedHPA Policy

You can view the rules and latest status of a CustomedHPA policy and rectify faults based on the error information displayed.

- Step 1** Log in to the CCE console. In the navigation pane, choose **Auto Scaling**. On the **Workload Scaling** tab page, click  in front of the target policy.
- Step 2** In the expanded area, if the policy is abnormal on the **Rules** tab page, click **Details** in **Latest Status** and locate the fault based on the information displayed.

### NOTE

You can also view the created CustomedHPA policy on the workload details page. Log in to the CCE console, choose **Workloads > Deployments** or **Workloads > StatefulSets** in the navigation pane, and choose **More > Scaling** in the **Operation** column. On the workload details page, click the **Scaling** tab. You can see the **Auto Scaling-HPA / CustomedHPA** pane, as well as the CustomedHPA policy you have configured on the **Auto Scaling** page.

----End

## Updating an HPA or CustomedHPA Policy

An HPA policy is used as an example.

- Step 1** Log in to the CCE console. In the navigation pane, choose **Auto Scaling**. On the **Workload Scaling** tab page, click **Update** in the **Operation** column of the policy to be updated.
- Step 2** On the **Update HPA Policy** page displayed, set the policy parameters listed in [Table 17-140](#).
- Step 3** Click **Update**.

----End

## Cloning an HPA or CustomedHPA Policy

- Step 1** Log in to the CCE console. In the navigation pane, choose **Auto Scaling**. On the **Workload Scaling** tab page, click **Clone** in the **Operation** column of the target policy.
- Step 2** For example, for an HPA policy, on the **Create HPA Policy** page, you can view that parameters such as **Pod Range**, **Cooldown Period**, and **Rules** have been cloned. Add or modify other policy parameters as needed.
- Step 3** Click **Create** to complete policy cloning. On the **Workload Scaling** tab page, you can view the cloned policy in the policy list.

----End

## Editing the YAML File (HPA Policy)

- Step 1** Log in to the CCE console. In the navigation pane, choose **Auto Scaling**. On the **Workload Scaling** tab page, choose **More** > **Edit YAML** in the **Operation** column of the target policy.
- Step 2** In the **Edit YAML** dialog box displayed, edit or download the YAML file.
- Step 3** Click the close button in the upper right corner.

----End

## Viewing the YAML File (CustomedHPA Policy)

- Step 1** Log in to the CCE console. In the navigation pane, choose **Auto Scaling**. On the **Workload Scaling** tab page, choose **More** > **View YAML** in the **Operation** column of the target policy.
- Step 2** In the dialog box displayed, you can copy and download the YAML file but cannot modify it.
- Step 3** Click the close button in the upper right corner.

----End

## Deleting an HPA or CustomedHPA Policy

- Step 1** Log in to the CCE console. In the navigation pane, choose **Auto Scaling**. On the **Workload Scaling** tab page, choose **More** > **Delete** in the **Operation** column of the target policy.
- Step 2** In the **Delete HPA Policy** dialog box displayed, confirm whether to delete the HPA policy.
- Step 3** Click **Yes** to delete the policy.

----End

## 17.15.3 Scaling a Cluster/Node



### 17.15.3.1 Node Scaling Mechanisms

Kubernetes HPA is designed for pods. However, if the cluster resources are insufficient, you can only add nodes. Scaling of cluster nodes could be laborious. Now with clouds, you can add or delete nodes by simply calling APIs.

**autoscaler** is a component provided by Kubernetes for auto scaling of cluster nodes based on the pod scheduling status and resource usage.

#### Prerequisites

Before using the node scaling function, you must install the **autoscaler** add-on of v1.13.8 or later.

#### How autoscaler Works

The cluster autoscaler (CA) goes through two processes.

- **Scale-out:** The CA checks all unschedulable pods every 10 seconds and selects a node group that meets the requirements for scale-out based on the policy you set.
- **Scale-in:** The CA scans all nodes every 10 seconds. If the number of pod requests on a node is less than the user-defined percentage for scale-in, the CA simulates whether the pods on the node can be migrated to other nodes. If yes, the node will be removed after an idle time window.

As described above, if a cluster node is idle for a period of time (10 minutes by default), scale-in is triggered, and the idle node is removed.

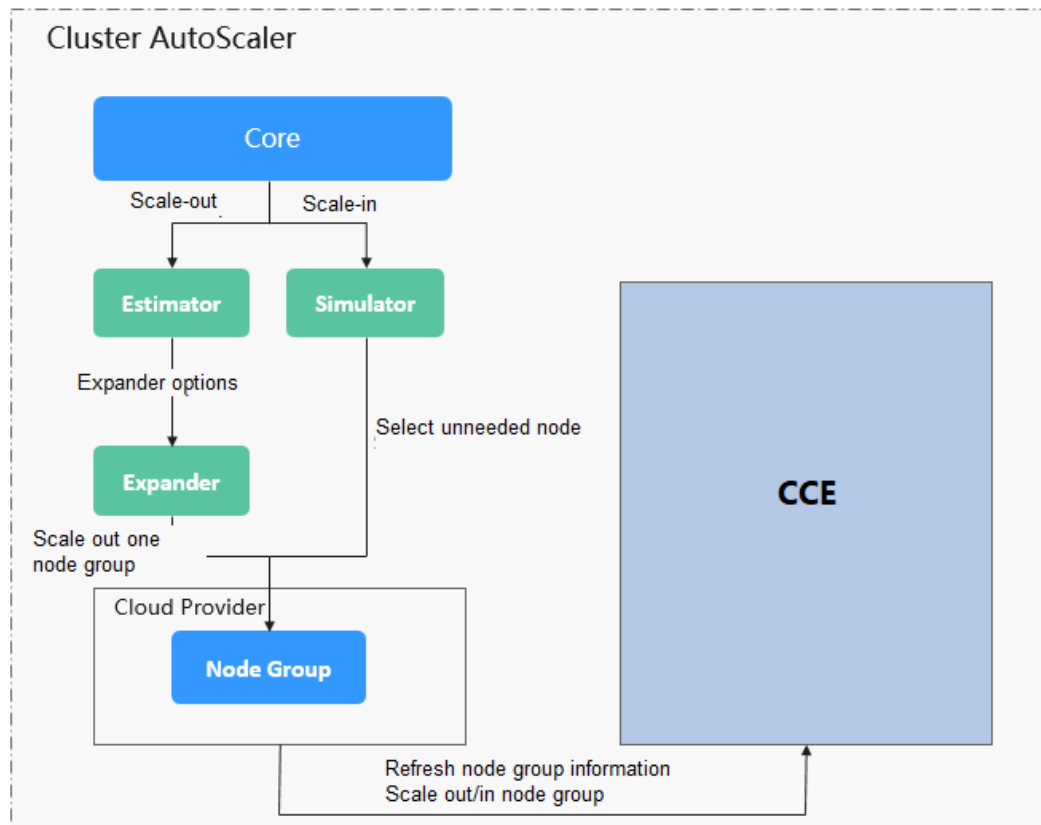
However, a node cannot be removed from a cluster if the following pods exist:

1. Pods that do not meet specific requirements set in PodDisruptionBudget
2. Pods that cannot be scheduled to other nodes due to constraints such as affinity and anti-affinity policies
3. Pods that have the "**cluster-autoscaler.kubernetes.io/safe-to-evict**": "**false**" annotation
4. Pods (except those created by kube-system DaemonSet) that exist in the kube-system namespace on the node
5. Pods that are not created by the controller (Deployment/ReplicaSet/job/StatefulSet)

#### autoscaler Architecture

**Figure 17-159** shows the autoscaler architecture and its core modules:

Figure 17-159 autoscaler architecture



### Description

- **Estimator:** Evaluates the number of nodes to be added to each node pool to host unschedulable pods.
- **Simulator:** Finds the nodes that meet the scale-in conditions in the scale-in scenario.
- **Expander:** Selects an optimal node from the node pool picked out by the Estimator based on the user-defined policy in the scale-out scenario. Currently, the Expander has the following policies:
  - **Random:** Selects a node pool randomly. If you have not specified a policy, **Random** is set by default.
  - **most-Pods:** Selects the node pool that can host the largest number of unschedulable pods after the scale-out. If multiple node pools meet the requirement, a random node pool will be selected.
  - **least-waste:** Selects the node pool that has the least CPU or memory resource waste after scale-out.
  - **price:** Selects the node pool in which the to-be-added nodes cost least for scale-out.
  - **priority:** Selects the node pool with the highest weight. The weights are user-defined.

Currently, CCE supports all policies except **price**. By default, CCE add-ons use the **least-waste** policy.

### 17.15.3.2 Creating a Node Scaling Policy

CCE provides auto scaling through the **autoscaler** add-on. Nodes with different specifications can be automatically added across AZs on demand.

If a node scaling policy and the configuration in the autoscaler add-on take effect at the same time, for example, there are pods that cannot be scheduled and the value of a metric reaches the threshold at the same time, scale-out is performed first for the unschedulable pods.

- If the scale-out succeeds for the unschedulable pods, the system skips the metric-based rule logic and enters the next loop.
- If the scale-out fails for the unschedulable pods, the metric-based rule is executed.


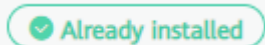
#### Prerequisites

Before using the node scaling function, you must install the **autoscaler** add-on of v1.13.8 or later.

#### Procedure

**Step 1** Log in to the CCE console. In the navigation pane, choose **Auto Scaling**. On the **Node Scaling** tab page, click **Create Node Scaling Policy**.

**Step 2** In the **Check Add-ons** step:

- If  **Not installed** is displayed next to the add-on name, click **Install**, set add-on parameters as required, and click **Install** to install the add-on.
- If  **Already installed** is displayed next to the add-on name, the add-on has been installed.

**Step 3** After the required add-ons have been installed, click **Next: Policy configuration**.

#### NOTE

If the add-ons have been installed, after you click **Create Node Scaling Policy**, you will directly land on the second step to configure the policy. The first step (checking the add-ons) has been completed almost instantly.

**Step 4** On the **Create Node Scaling Policy** page, set the following policy parameters.

- **Policy Name:** name of the policy to be created, which can be customized.
- **Associated Node Pool:** Click **Add Node Pool** and select the node pool to be associated. You can associate multiple node pools to use the same scaling policy.

 NOTE

Priority is now supported for node pools. CCE will select a node pool for auto scaling based on the following policies:

1. CCE uses algorithms to determine whether a node pool meets the conditions to allow scheduling of a pod in pending state, including whether the node resources are greater than requested by the pod, and whether the nodeSelect, nodeAffinity, and taints meet the conditions. In addition, the node pools that fail to be scaled (due to insufficient resources or other reasons) and are still in the 15-minute cool-down interval are filtered.
2. If multiple node pools meet the scaling requirements, the system checks the priority of each node pool and selects the node pool with the highest priority for scaling. The value ranges from 0 to 100 and the default priority is 0. The value 100 indicates the highest priority, and the value 0 indicates the lowest priority.
3. If multiple node pools have the same priority or no priority is configured for them, the system selects the node pool that will consume the least resources based on the configured VM specification.
4. If the VM specifications of multiple node pools are the same but the node pools are deployed in different AZs, the system randomly selects a node pool to trigger scaling.
5. If the resources of the preferred node pool are insufficient, the system automatically selects next node pool based on the priority.

For details about the node pool priority, see [Autoscaler](#).

- **Execution Rules:** Click **Add Rule**. In the dialog box displayed, set the following parameters:

**Name:** Enter a rule name.

**Type:** You can select **Metric-based** or **Periodic**. The differences between the two types are as follows:

– **Metric-based:**

- **Condition:** Select **CPU allocation** or **Memory allocation** and enter a value. The value must be greater than the scale-in percentage configured in the autoscaler add-on.

 NOTE

- Resource allocation (%) = Resources requested by pods in the node pool/Resources allocatable to pods in the node pool
- **If multiple rules meet the conditions, the rules are executed in either of the following modes:**
  - If rules based on the **CPU allocation rate** and **memory allocation rate** are configured and two or more rules meet the scale-out conditions, the rule that will add the most nodes will be executed.
  - If a rule based on the **CPU allocation rate** and a **periodic rule** are configured and they both meet the scale-out conditions, one of them will be executed randomly. The rule executed first (rule A) changes the node pool to the scaling state. As a result, the other rule (rule B) cannot be executed. After rule A is executed and the node pool status becomes normal, rule B will not be executed.
- If rules based on the **CPU allocation rate** and **memory allocation rate** are configured, the policy detection period varies with the processing logic of each loop of the autoscaler add-on. Scale-out is triggered once the conditions are met, but it is constrained by other factors such as the cool-down interval and node pool status.

- **Action:** Set an action to be performed when the trigger condition is met.

**Figure 17-160** Setting an action for metric-based triggering

Add Rule ×

Rule Name

Type Metric-based Periodic

Condition Memory a... > 40 %

Resource allocation (%) = Resources requested by pods in the node pool/Resources allocatable to pods in the node pool  
The entered percentage must be greater than the scale-in percentage configured on the autoscaler add-on.

Action Add 5 nodes

OK Cancel

– **Periodic:**

- **Triggered At:** You can select a specific time point every day, every week, every month, or every year.

**Figure 17-161** Setting a trigger time

Add Rule ×

Rule Name

Type Metric-based Periodic

Triggered At Daily 15:00

Action Add 5 nodes

OK Cancel

- **Action:** Set an action to be performed when the **Triggered At** value is reached.

**Figure 17-162** Setting an action for periodic triggering

You can click **Add Rule** again to add more node scaling policies. You can add a maximum of one CPU usage-based rule and one memory usage-based rule. The total number of rules cannot exceed 10.

**Step 5** After the configuration is complete, click **Create**. If the system displays a message indicating that the request to create a node scaling policy is submitted successfully, click **Back to Node Scaling Policy List**.

**Step 6** On the **Node Scaling** tab page, you can view the created node scaling policy.

**Figure 17-163** Node scaling policy

Name	Status	Created	Associated Node Pool	Execution Rules	Scaling Records	Operation
node-test	Enabled	Jan 16, 2020 15:33:50 GMT+08:00	1	1	0	Delete Edit More

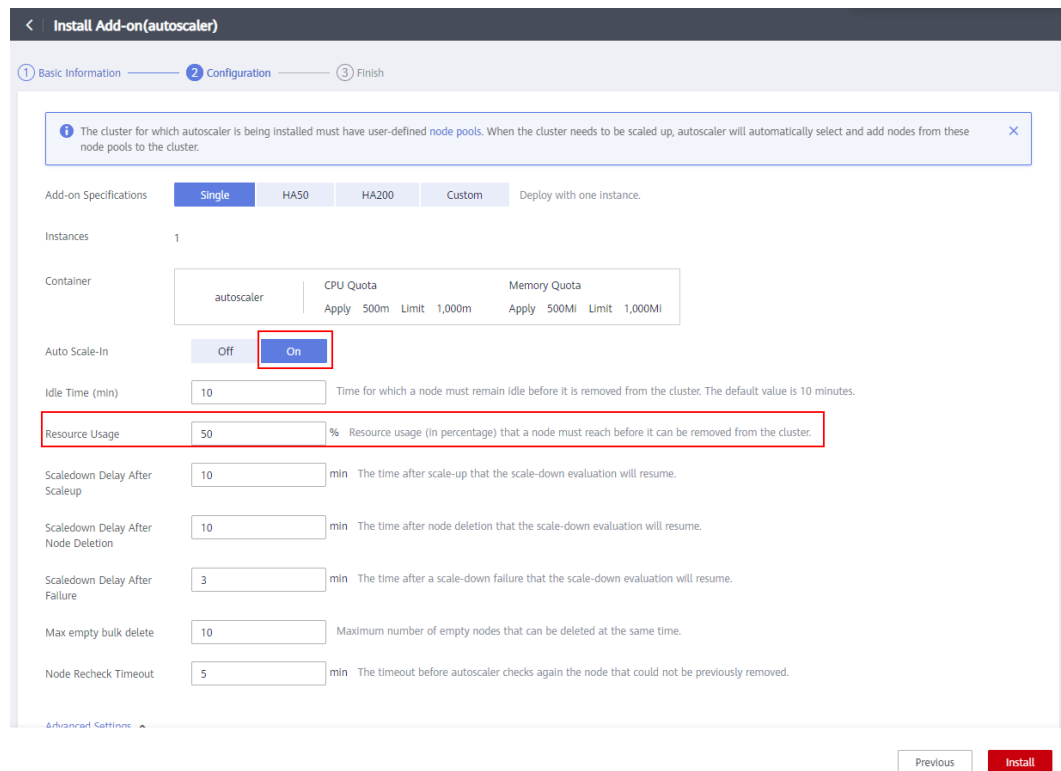
----End

## Constraints on Scale-in

CCE cannot trigger scale-in by using node scaling policies. You can set a scale-in policy when installing the **autoscaler add-on**.

Node scale-in can be triggered only by the resource allocation rate. When CPU and memory allocation rates in a cluster are lower than the specified thresholds (set when the autoscaler add-on is installed or modified), scale-in is triggered for nodes in the node pool (this function can be disabled).

**Figure 17-164** Auto scale-in configuration



## Example YAML File

The following is a YAML example of a node scaling policy:

```

apiVersion: autoscaling.cce.io/v1alpha1
kind: HorizontalNodeAutoscaler
metadata:
  creationTimestamp: "2020-02-13T12:47:49Z"
  generation: 1
  name: xxxx
  namespace: kube-system
  resourceVersion: "11433270"
  selfLink: /apis/autoscaling.cce.io/v1alpha1/namespaces/kube-system/horizontalnodeautoscalers/xxxx
  uid: c2bd1e1d-60aa-47b5-938c-6bf3fadbe91f
spec:
  disable: false
  rules:
  - action:
    type: ScaleUp
    unit: Node
    value: 1
    cronTrigger:
      schedule: 47 20 * * *
    disable: false
    ruleName: cronrule
    type: Cron
  - action:
    type: ScaleUp
    unit: Node
    value: 2
    disable: false
    metricTrigger:
      metricName: Cpu
      metricOperation: '>'
      metricValue: "40"
    unit: Percent
  
```

```
ruleName: metricrule
type: Metric
targetNodepoolIds:
- 7d48eca7-3419-11ea-bc29-0255ac1001a8
```

**Table 17-143** Key parameters

Parameter	Type	Description
spec.disable	Bool	Whether to enable the scaling policy. This parameter takes effect for all rules in the policy.
spec.rules	Array	All rules in a scaling policy.
spec.rules[x].ruleName	String	Rule name.
spec.rules[x].type	String	Rule type. Currently, <b>Cron</b> and <b>Metric</b> are supported.
spec.rules[x].disable	Bool	Rule switch. Currently, only <b>false</b> is supported.
spec.rules[x].action.type	String	Rule action type. Currently, only <b>ScaleUp</b> is supported.
spec.rules[x].action.unit	String	Rule action unit. Currently, only <b>Node</b> is supported.
spec.rules[x].action.value	Integer	Rule action value.
spec.rules[x].cronTrigger	/	Optional. This parameter is valid only in periodic rules.
spec.rules[x].cronTrigger.schedule	String	Cron expression of a periodic rule.
spec.rules[x].metricTrigger	/	Optional. This parameter is valid only in metric-based rules.
spec.rules[x].metricTrigger.metricName	String	Metric of a metric-based rule. Currently, <b>Cpu</b> and <b>Memory</b> are supported.
spec.rules[x].metricTrigger.metricOperation	String	Comparison operator of a metric-based rule. Currently, only <b>&gt;</b> is supported.
spec.rules[x].metricTrigger.metricValue	String	Metric threshold of a metric-based rule. The value can be any integer from 1 to 100 and must be a character string.
spec.rules[x].metricTrigger.Unit	String	Unit of the metric-based rule threshold. Currently, only <b>%</b> is supported.



Parameter	Type	Description
spec.targetNodepoolIds	Array	All node pools associated with the scaling policy.
spec.targetNodepoolIds[x]	String	ID of the node pool associated with the scaling policy.


### 17.15.3.3 Managing Node Scaling Policies

#### Scenario

After a node scaling policy is created, you can delete, edit, disable, enable, or clone the policy.

#### Viewing a Node Scaling Policy

You can view the associated node pool, rules, and scaling history of a node scaling policy and rectify faults according to the error information displayed.

- Step 1** Log in to the CCE console. In the navigation pane, choose **Auto Scaling**. On the **Node Scaling** tab page, click  in front of the policy to be viewed.
- Step 2** In the expanded area, the **Associated Node Pool**, **Execution Rules**, and **Scaling Records** tab pages are displayed. If the policy is abnormal, locate and rectify the fault based on the error information.

 **NOTE**

You can also enable or disable auto scaling in **Node Pools**. Log in to the CCE console. In the navigation pane, choose **Resource Management > Node Pools**, and click **Edit** in the upper right corner of the node pool to be operated. In the **Edit Node Pool** dialog box displayed, you can enable **Autoscaler** and set the limits of the number of nodes.

----End

#### Deleting a Node Scaling Policy

- Step 1** Log in to the CCE console. In the navigation pane, choose **Auto Scaling**. On the **Node Scaling** tab page, click **Delete** in the **Operation** column of the policy to be deleted.
- Step 2** In the **Delete Node Policy** dialog box displayed, confirm whether to delete the policy.
- Step 3** Enter **DELETE** in the text box.
- Step 4** Click **OK** to delete the policy.

----End

#### Editing a Node Scaling Policy

- Step 1** Log in to the CCE console. In the navigation pane, choose **Auto Scaling**. On the **Node Scaling** tab page, click **Edit** in the **Operation** column of the policy.

**Step 2** On the **Edit Node Scaling Policy** page displayed, modify policy parameter values listed in [Table 17-143](#).

**Step 3** After the configuration is complete, click **OK**.

----End

## Cloning a Node Scaling Policy

**Step 1** Log in to the CCE console. In the navigation pane, choose **Auto Scaling**. On the **Node Scaling** tab page, click **More** > **Clone** in the **Operation** column of the policy.

**Step 2** On the **Create Node Scaling Policy** page displayed, certain parameters have been cloned. Add or modify other policy parameters based on service requirements.

**Step 3** Click **Create Now** to clone the policy. The cloned policy is displayed in the policy list on the **Node Scaling** tab page.

----End

## Enabling or Disabling a Node Scaling Policy

**Step 1** Log in to the CCE console. In the navigation pane, choose **Auto Scaling**. On the **Node Scaling** tab page, click **More** > **Disable** or **Enable** in the **Operation** column of the policy.

**Step 2** In the dialog box displayed, confirm whether to disable or enable the node policy.

**Step 3** Click **Yes**. The policy status is displayed in the node scaling list.

----End

## 17.15.4 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.

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. Pod scaling can effectively resolve this issue. If the resource usage on the node increases to a certain extent, newly added pods cannot be scheduled to this node. In this case, CCE will add nodes accordingly.

### Solution

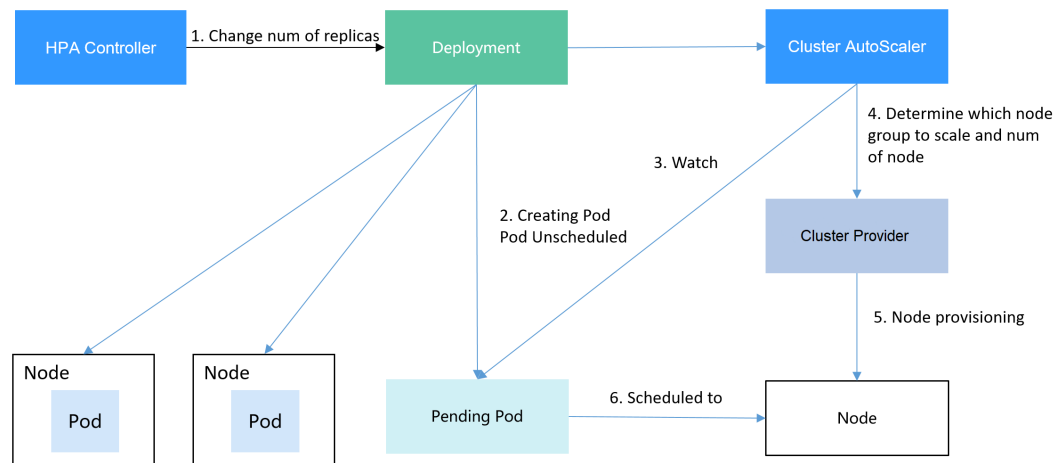
Two major auto scaling policies are HPA (Horizontal Pod Autoscaling) and CA (Cluster AutoScaling). HPA is for workload auto scaling and CA is for node auto scaling.

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 17-165](#), 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.

**Figure 17-165** 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 cores of vCPUs and 4 GiB of memory, or a higher specification, 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.

**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  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 {Image repository address}/group/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 {Image repository address}/group/hpa-example: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 defaults for other parameters.

**Step 3** Locate the row containing the newly created node pool and click **Auto Scaling** in the upper right corner.

If the CCE Cluster Autoscaler add-on is not installed in the cluster, install it first.

- **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

**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:
            limits: # The value of limits must be the same as that of requests to prevent flapping
                    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.

```
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 hpa-example 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
```

## 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**

 **NOTE**

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.

Observe the scaling process of the workload.

```
# kubectl get hpa hpa-policy --watch
NAME          REFERENCE          TARGETS  MINPODS  MAXPODS  REPLICAS  AGE
hpa-policy    Deployment/hpa-example  0%/50%   1         100       1          4m
hpa-policy    Deployment/hpa-example  190%/50%  1         100       1          4m23s
hpa-policy    Deployment/hpa-example  190%/50%  1         100       4          4m31s
hpa-policy    Deployment/hpa-example  200%/50%  1         100       4          5m16s
hpa-policy    Deployment/hpa-example  200%/50%  1         100       4          6m16s
hpa-policy    Deployment/hpa-example  85%/50%   1         100       4          7m16s
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    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
# 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
```

Check the number of nodes. The following output shows that two nodes are added.

```
# kubectl get node
NAME          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 get hpa hpa-policy --watch
NAME          REFERENCE          TARGETS  MINPODS  MAXPODS  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 hpa-
```



```
example-79dd795485 to 3
Normal ScalingReplicaSet 72s deployment-controller Scaled down replica set hpa-
example-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).

----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.

## 17.16 Permissions Management

### 17.16.1 Permissions Overview

CCE permissions management allows you to assign permissions to IAM users and user groups under your tenant accounts. CCE combines the advantages of Identity and Access Management (IAM) and Kubernetes Role-based Access Control (RBAC) authorization to provide a variety of authorization methods, including IAM fine-grained authorization, IAM token authorization, cluster-scoped authorization, and namespace-wide authorization.

If you need to perform refined permissions management on CCE clusters and related resources, for example, to control the access of employees in different departments to cloud resources, you can perform multi-dimensional permissions management on CCE.

This section describes the CCE permissions management mechanism and related concepts. If your account has met your service requirements, you can skip the configurations in this chapter.

## CCE Permissions Management

CCE permissions are described as follows:

- **Cluster-level permissions:** Cluster-level permissions management evolves out of the system policy authorization feature of IAM. IAM users in the same user group have the same permissions. On IAM, you can configure system policies

to describe which IAM user groups can perform which operations on cluster resources. For example, you can grant user group A to create and delete cluster X, add a node, or install an add-on, while granting user group B to view information about cluster X.

Cluster-level permissions involve CCE non-Kubernetes APIs and support fine-grained IAM policies and enterprise project management capabilities.

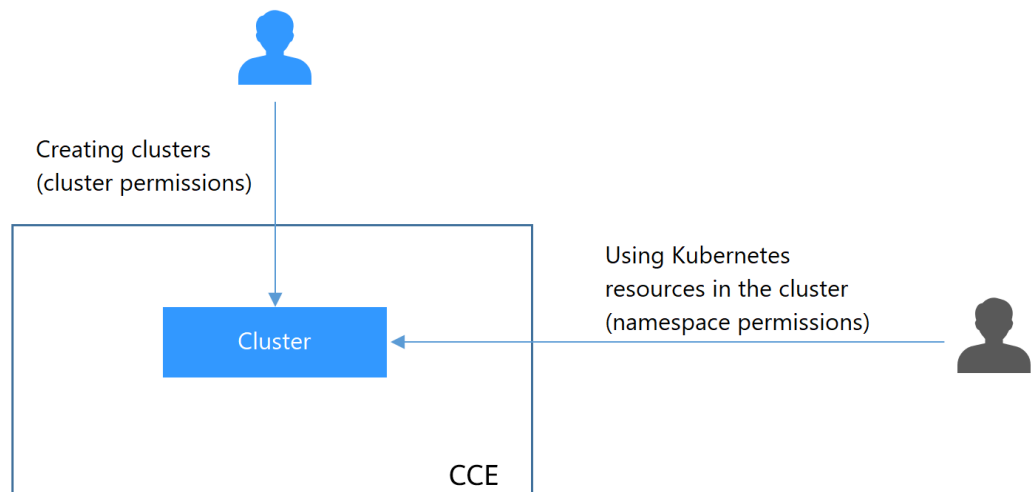
- Namespace-level permissions:** You can regulate users' or user groups' access to Kubernetes resources in a single namespace based on their Kubernetes RBAC roles. CCE has also been enhanced based on open-source capabilities. It supports RBAC authorization based on IAM user or user group, and RBAC authentication on access to APIs using IAM tokens.

Namespace-level permissions involve CCE Kubernetes APIs and are enhanced based on the Kubernetes RBAC capabilities. Namespace-level permissions can be granted to IAM users or user groups for authentication and authorization, but are independent of fine-grained IAM policies.

Starting from version 1.11.7-r2, CCE clusters allow you to configure namespace permissions. Clusters earlier than v1.11.7-r2 have all namespace permissions by default.

In general, you configure CCE permissions in two scenarios. The first is creating and managing clusters and related resources, such as nodes. The second is creating and using Kubernetes resources in the cluster, such as workloads and Services.

**Figure 17-166** Illustration on CCE permissions



These permissions allow you to manage resource users at a finer granularity.

## Cluster Permissions (IAM-based) and Namespace Permissions (Kubernetes RBAC-based)

Users with different cluster permissions (assigned using IAM) have different namespace permissions (assigned using Kubernetes RBAC). [Table 17-144](#) lists the namespace permissions of different users.

**Table 17-144** Differences in namespace permissions

User	Permission
User with the Tenant Administrator permissions	All namespace permissions
IAM user with the CCE Administrator role	All namespace permissions
IAM user with the CCE FullAccess or CCE ReadOnlyAccess role	Requires Kubernetes RBAC authorization.
IAM user with the Tenant Guest role	Requires Kubernetes RBAC authorization.

## kubectl Permissions

You can use [kubectl](#) to access Kubernetes resources in a cluster.

When you access a cluster using kubectl, CCE uses the kubeconfig.json file generated on the cluster for authentication. This file contains user information, based on which CCE determines which Kubernetes resources can be accessed by kubectl. The permissions recorded in a kubeconfig.json file vary from user to user. The permissions that a user has are listed in [Table 17-144](#).

### 17.16.2 Cluster Permissions (IAM-based)

CCE cluster permissions are assigned based on IAM **system policies** and **custom policies**. You can use user groups to assign permissions to IAM users.

---

 **CAUTION**

**Cluster permissions** are configured only for cluster-related resources (such as clusters and nodes). You must also configure [namespace permissions](#) to operate Kubernetes resources (such as workloads and Services).

---

#### Prerequisites

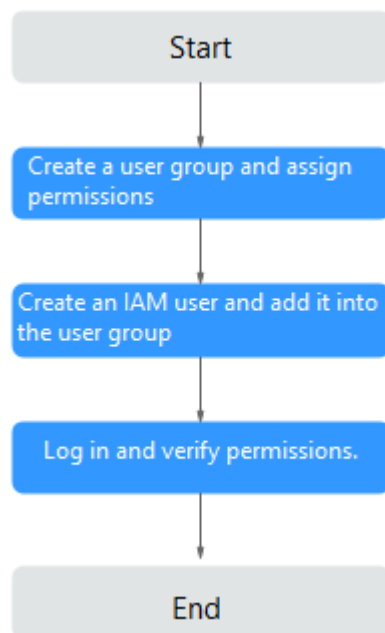
- A user with the Security Administrator role has all IAM permissions except role switching. Only these users can view user groups and their permissions on the **Permissions Management** page on the CCE console.

#### Configuration

On the CCE console, when you choose **Permissions Management > Cluster-Level Permissions** to create a user group, you will be directed to the IAM console to complete the process. After the user group is created and its permissions are configured, you can view the information on the **Cluster-Level Permissions** tab page. This section describes the operations in IAM.

## Process Flow

**Figure 17-167** Process of assigning CCE permissions



1. Create a user group and assign permissions to it.

Create a user group on the IAM console, and assign CCE permissions, for example, the **CCE ReadOnlyAccess** policy to the group.

### NOTE

CCE is deployed by region. On the IAM console, select **Region-specific projects** when assigning CCE permissions.

2. Create a user and add it to a user group.

Create a user on the IAM console and add the user to the group created in **1**.

3. Log in and verify permissions.

Log in to the management console as the user you created, and verify that the user has the assigned permissions.

- Log in to the management console and switch to the CCE console. Click **Buy Cluster** in the upper right corner. If you fail to do so (assuming that only the CCE ReadOnlyAccess role is assigned), the permission policy takes effect.
- Switch to the console of any other service. If a message appears indicating that you do not have the required permissions to access the service, the **CCE ReadOnlyAccess** policy takes effect.

## System-defined Roles

Roles are a type of coarse-grained authorization mechanism that defines service-level permissions based on user responsibilities. This mechanism provides only a limited number of service-level roles for authorization. However, roles are not an ideal choice for fine-grained authorization and secure access control.

The preset system role for CCE in IAM is **CCE Administrator**. When assigning this role to a user group, you must also assign other roles and policies on which this role depends, such as **Tenant Guest**, **Server Administrator**, and **ELB Administrator**.

## System-defined Policies

The system policies preset for CCE in IAM are **CCE FullAccess** and **CCE ReadOnlyAccess**.

- **CCE FullAccess**: common operation permissions on CCE cluster resources, excluding the namespace-level permissions for the clusters (with Kubernetes RBAC enabled) and the privileged administrator operations, such as agency configuration and cluster certificate generation
- **CCE ReadOnlyAccess**: permissions to view CCE cluster resources, excluding the namespace-level permissions of the clusters (with Kubernetes RBAC enabled)

## Custom Policies

Custom policies can be created as a supplement to the system-defined policies of CCE.

You can create custom policies in either of the following ways:

- Visual editor: Select cloud services, actions, resources, and request conditions. This does not require knowledge of policy syntax.
- JSON: Edit JSON policies from scratch or based on an existing policy.

This section provides examples of common custom CCE policies.

### Example Custom Policies:

- Example 1: Creating a cluster named **test**

```
{
  "Version": "1.1",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "cce:cluster:create"
      ]
    }
  ]
}
```

- Example 2: Denying node deletion

A policy with only "Deny" permissions must be used in conjunction with other policies to take effect. If the permissions assigned to a user contain both "Allow" and "Deny", the "Deny" permissions take precedence over the "Allow" permissions.

The following method can be used if you need to assign permissions of the **CCEFullAccess** policy to a user but you want to prevent the user from deleting nodes (**cce:node:delete**). Create a custom policy for denying node deletion, and attach both policies to the group to which the user belongs. Then, the user can perform all operations on CCE except deleting nodes. The following is an example of a deny policy:

```
{
  "Version": "1.1",
  "Statement": [
    {
      "Effect": "Deny",
      "Action": [
        "cce:node:delete"
      ]
    }
  ]
}
```

- Example 3: Defining permissions for multiple services in a policy

A custom policy can contain the actions of multiple services that are of the global or project-level type. The following is an example policy containing actions of multiple services:

```
{
  "Version": "1.1",
  "Statement": [
    {
      "Action": [
        "ecs:cloudServers:resize",
        "ecs:cloudServers:delete",
        "ecs:cloudServers:delete",
        "ims:images:list",
        "ims:serverImages:create"
      ],
      "Effect": "Allow"
    }
  ]
}
```

## CCE Cluster Permissions and Enterprise Projects

CCE supports resource management and permission allocation by cluster and enterprise project.

Note that:

- IAM projects are based on physical isolation of resources, whereas enterprise projects provide global logical groups of resources, which better meet the actual requirements of enterprises. In addition, IAM policies can be managed based on enterprise projects. Therefore, you are advised to use enterprise projects for permissions management.
- When there are both IAM projects and enterprise projects, IAM preferentially matches the IAM project policies.
- When creating a cluster or node using purchased cloud resources, ensure that IAM users have been granted the required permissions in the enterprise project to use these resources. Otherwise, the cluster or node may fail to be created.

## CCE Cluster Permissions and IAM RBAC

CCE is compatible with IAM system roles for permissions management. You are advised to use fine-grained policies provided by IAM to simplify permissions management.

CCE supports the following roles:

- Basic IAM roles:

- te\_admin (Tenant Administrator): Users with this role can call all APIs of all services except IAM.
- readonly (Tenant Guest): Users with this role can call APIs with the read-only permissions of all services except IAM.
- Custom CCE administrator role: CCE Administrator

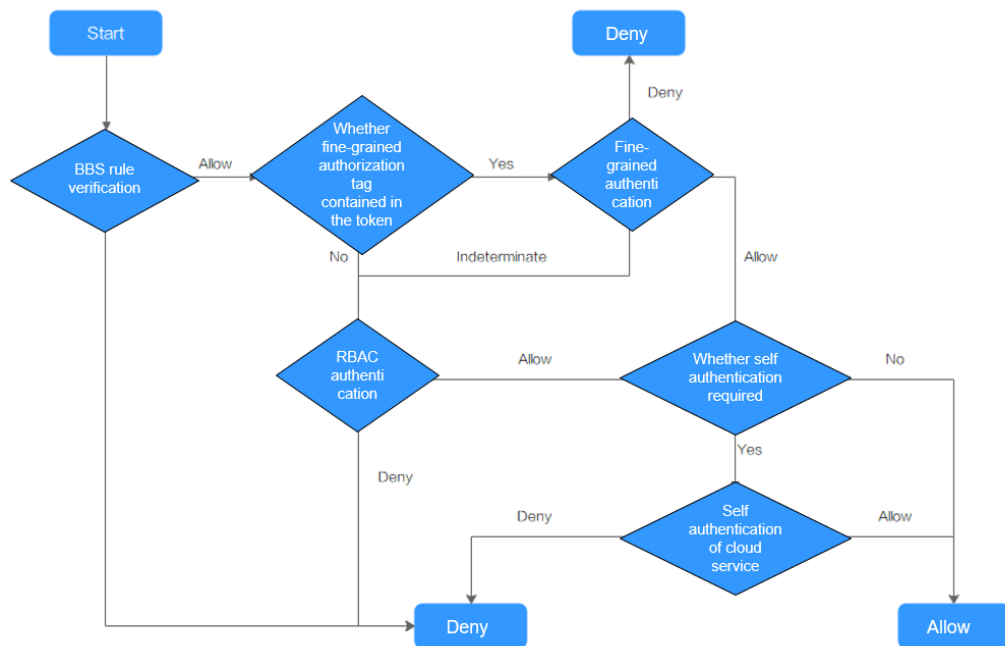
**NOTE**

- Tenant Administrator and Tenant Guest are special IAM system roles. After any system or custom policy is configured, Tenant Administrator and Tenant Guest take effect as system policies to achieve compatibility with IAM RBAC and ABAC scenarios.
- If a user has the Tenant Administrator or CCE Administrator system role, the user has the cluster-admin permissions in Kubernetes RBAC and the permissions cannot be removed after the cluster is created.

If the user is the cluster creator, the cluster-admin permissions in Kubernetes RBAC are granted to the user by default. The permissions can be manually removed after the cluster is created.

- Method 1: Choose **Permissions Management > Namespace-Level Permissions > Delete** at the same role as cluster-creator on the CCE console.
- Method 2: Delete **ClusterRoleBinding: cluster-creator** through the API or kubectl.

When RBAC and IAM policies co-exist, the backend authentication logic for open APIs or console operations on CCE is as follows:



**CAUTION**

Certain CCE APIs involve namespace-level permissions or key operations and therefore, they require special permissions:

Using clusterCert to obtain the cluster kubeconfig: cceadm/teadmin

## 17.16.3 Namespace Permissions (Kubernetes RBAC-based)

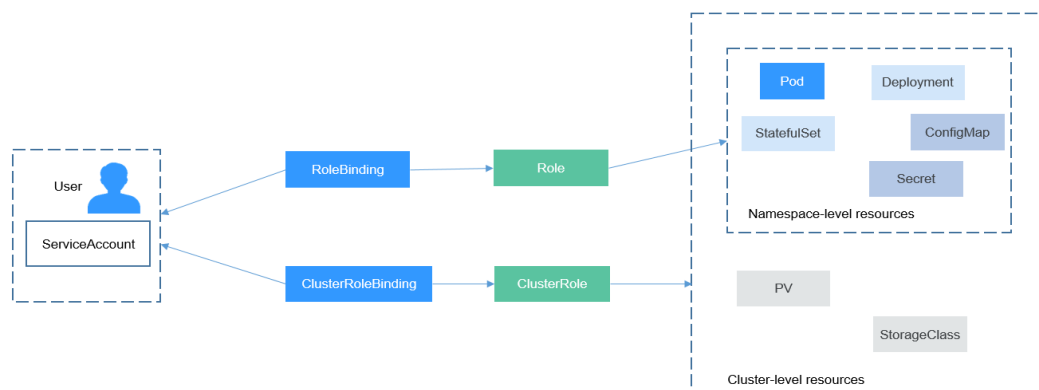
### Namespace Permissions (Kubernetes RBAC-based)

You can regulate users' or user groups' access to Kubernetes resources in a single namespace based on their Kubernetes RBAC roles. The RBAC API declares four kinds of Kubernetes objects: Role, ClusterRole, RoleBinding, and ClusterRoleBinding, which are described as follows:

- Role: defines a set of rules for accessing Kubernetes resources in a namespace.
- RoleBinding: defines the relationship between users and roles.
- ClusterRole: defines a set of rules for accessing Kubernetes resources in a cluster (including all namespaces).
- ClusterRoleBinding: defines the relationship between users and cluster roles.

Role and ClusterRole specify actions that can be performed on specific resources. RoleBinding and ClusterRoleBinding bind roles to specific users, user groups, or ServiceAccounts. Illustration:

**Figure 17-168** Role binding



On the CCE console, you can assign permissions to a user or user group to access resources in one or multiple namespaces. By default, the CCE console provides the following ClusterRoles:

- view: read-only permission on most resources in all or selected namespaces.
- edit: read and write permissions on most resources in all or selected namespaces. If this ClusterRole is configured for all namespaces, its capability is the same as the O&M permission.
- admin: read and write permissions on most resources in all namespaces, and read-only permission on nodes, storage volumes, namespaces, and quota management.
- cluster-admin: read and write permissions on all resources in all namespaces.



## Cluster Permissions (IAM-based) and Namespace Permissions (Kubernetes RBAC-based)

Users with different cluster permissions (assigned using IAM) have different namespace permissions (assigned using Kubernetes RBAC). [Table 17-145](#) lists the namespace permissions of different users.

**Table 17-145** Differences in namespace permissions

User	Permission
User with the Tenant Administrator permissions	All namespace permissions
IAM user with the CCE Administrator role	All namespace permissions
IAM user with the CCE FullAccess or CCE ReadOnlyAccess role	Requires Kubernetes RBAC authorization.
IAM user with the Tenant Guest role	Requires Kubernetes RBAC authorization.

### Prerequisites

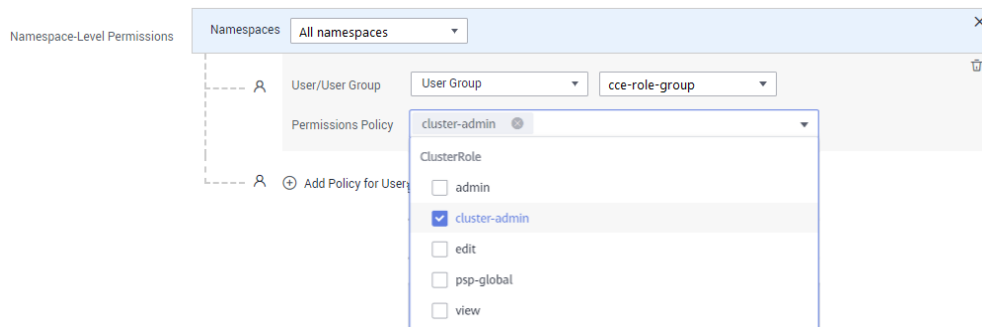
- Kubernetes RBAC authorization can be used for clusters of v1.11.7-r2 and later.
- After you create a cluster of v1.11.7-r2 or later, CCE automatically assigns the cluster-admin permission to you, which means you have full control on all resources in all namespaces in the cluster.
- A user with the Security Administrator role has all IAM permissions except role switching. Only these users can assign permissions on the **Permissions Management** page on the CCE console.

### Configuring Namespace Permissions (on the Console)

You can regulate users' or user groups' access to Kubernetes resources in a single namespace based on their Kubernetes RBAC roles.

- Step 1** Log in to the CCE console. In the navigation pane, choose **Permissions Management**.
- Step 2** On the displayed page, click the **Namespace-Level Permissions** tab. In the upper right corner of the namespace permissions list, select the cluster that contains the namespace whose access will be managed, and click **Add Permissions**.
- Step 3** Confirm the cluster name and select the namespace to assign permissions for. For example, select **All namespaces**, the target user or user group, and select the permissions.

**Figure 17-169** Configuring namespace permissions



**Step 4** Click **Create**.

----End

## Using kubectl to Configure Namespace Permissions

### NOTE

When you access a cluster using kubectl, CCE uses the kubeconfig.json file generated on the cluster for authentication. This file contains user information, based on which CCE determines which Kubernetes resources can be accessed by kubectl. The permissions recorded in a kubeconfig.json file vary from user to user. The permissions that a user has are listed in [Cluster Permissions \(IAM-based\) and Namespace Permissions \(Kubernetes RBAC-based\)](#).

In addition to cluster-admin, admin, edit, and view, you can define Roles and RoleBindings to configure the permissions to add, delete, modify, and query resources, such as pods, Deployments, and Services, in the namespace.

The procedure for creating a Role is very simple. To be specific, specify a namespace and then define rules. The rules in the following example are to allow GET and LIST operations on pods in the default namespace.

```
kind: Role
apiVersion: rbac.authorization.k8s.io/v1
metadata:
  namespace: default          # Namespace
  name: role-example
rules:
- apiGroups: [""]
  resources: ["pods"]         # The pod can be accessed.
  verbs: ["get", "list"]     # The GET and LIST operations can be performed.
```

- **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**.

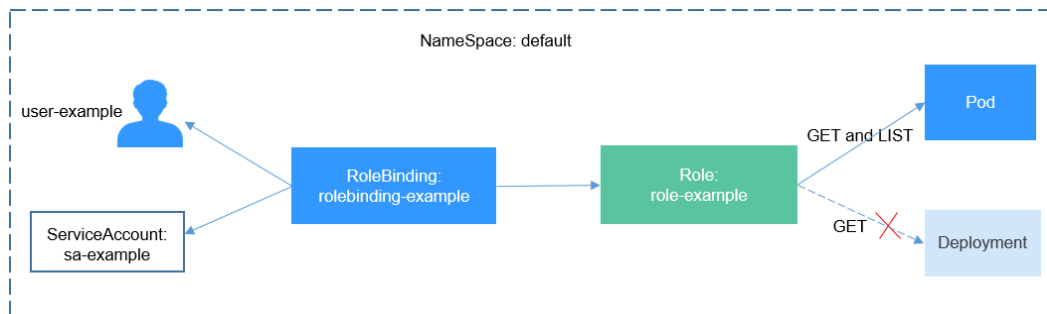
For details, see [Using RBAC Authorization](#).

After creating a Role, you can bind the Role to a specific user, which is called RoleBinding. The following shows an example:

```
kind: RoleBinding
apiVersion: rbac.authorization.k8s.io/v1
metadata:
  name: RoleBinding-example
  namespace: default
  annotations:
    CCE.com/IAM: 'true'
roleRef:
  kind: Role
  name: role-example
  apiGroup: rbac.authorization.k8s.io
subjects:
- kind: User
  name: 0c97ac3cb280f4d91fa7c0096739e1f8 # User ID of the user-example
  apiGroup: rbac.authorization.k8s.io
```

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.

**Figure 17-170** Binding a role to a user



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
```

Use the IAM user user-example to connect to the cluster and obtain the pod information. The following is an example of the returned pod information.

```
# kubectl get pod
NAME                                READY STATUS RESTARTS AGE
deployment-389584-2-6f6bd4c574-2n9rk 1/1   Running 0       4d7h
deployment-389584-2-6f6bd4c574-7s5qw 1/1   Running 0       4d7h
deployment-3895841-746b97b455-86g77 1/1   Running 0       4d7h
deployment-3895841-746b97b455-twvpm 1/1   Running 0       4d7h
nginx-658dff48ff-7rkph                1/1   Running 0       4d9h
nginx-658dff48ff-njdjh                1/1   Running 0       4d9h
# 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 required permissions. Try querying the pods in namespace kube-system. The output shows **user-example** does not have the required permissions, 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.

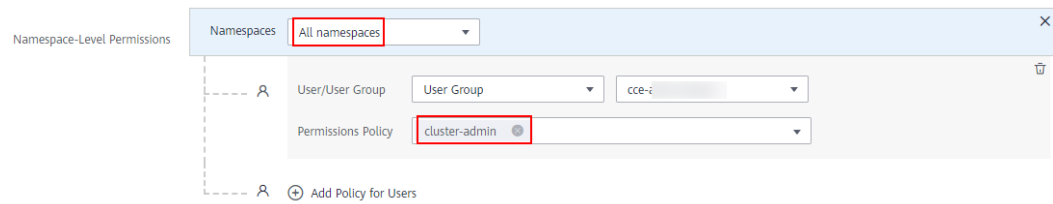
```
# kubectl get deploy
Error from server (Forbidden): deployments.apps is forbidden: User "0c97ac3cb280f4d91fa7c0096739e1f8"
```

```
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"
```

### Example: Assigning All Cluster Permissions (cluster-admin)

You can use the cluster-admin role to assign all permissions on a cluster. This role contains the permissions for cluster resources (such as PVs and StorageClasses).

Figure 17-171 Assigning all cluster permissions (cluster-admin)



In the following example kubectl output, a ClusterRoleBinding has been created and binds the cluster-admin role to the user group **cce-role-group**.

```
# kubectl get clusterrolebinding
NAME                                     ROLE                                AGE
clusterrole_cluster-admin_group0c96fad22880f32a3f84c009862af6f7  ClusterRole/cluster-admin          61s

# kubectl get clusterrolebinding clusterrole_cluster-admin_group0c96fad22880f32a3f84c009862af6f7 -oyaml
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  annotations:
    CCE.com/IAM: "true"
  creationTimestamp: "2021-06-23T09:15:22Z"
  name: clusterrole_cluster-admin_group0c96fad22880f32a3f84c009862af6f7
  resourceVersion: "36659058"
  selfLink: /apis/rbac.authorization.k8s.io/v1/clusterrolebindings/clusterrole_cluster-admin_group0c96fad22880f32a3f84c009862af6f7
  uid: d6cd43e9-b4ca-4b56-bc52-e36346fc1320
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: cluster-admin
subjects:
- apiGroup: rbac.authorization.k8s.io
  kind: Group
  name: 0c96fad22880f32a3f84c009862af6f7
```

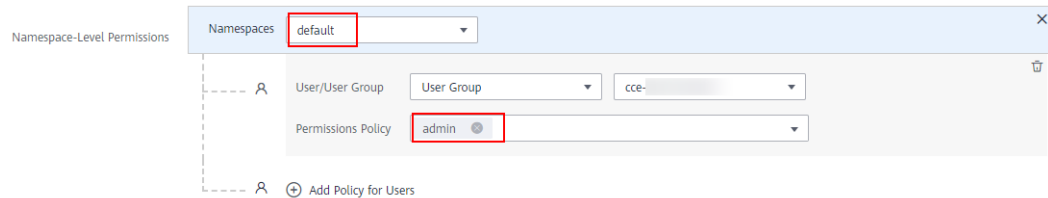
Connect to the cluster as an authorized user. If the PVs and StorageClasses can be queried, the permission configuration takes effect.

```
# kubectl get pv
No resources found
# kubectl get sc
NAME             PROVISIONER             RECLAIMPOLICY  VOLUMEBINDINGMODE
ALLOWVOLUMEEXPANSION  AGE
csi-disk         everest-csi-provisioner  Delete         Immediate         true             75d
csi-disk-topology everest-csi-provisioner  Delete         WaitForFirstConsumer  true             75d
csi-nas          everest-csi-provisioner  Delete         Immediate         true             75d
csi-obs          everest-csi-provisioner  Delete         Immediate         false            75d
csi-sfsturbo     everest-csi-provisioner  Delete         Immediate         true             75d
```

## Example: Assigning All Namespace Permissions (admin)

The admin role contains all permissions on a namespace. You can assign permissions to users to access one or multiple namespaces.

**Figure 17-172** Assigning all namespace permissions (admin)



In the following example kubectl output, a RoleBinding has been created, the admin role is bound to the user group **cce-role-group**, and the target namespace is the default namespace.

```
# kubectl get rolebinding
NAME                                ROLE                                AGE
clusterrole_admin_group0c96fad22880f32a3f84c009862af6f7  ClusterRole/admin 18s
# kubectl get rolebinding clusterrole_admin_group0c96fad22880f32a3f84c009862af6f7 -oyaml
apiVersion: rbac.authorization.k8s.io/v1
kind: RoleBinding
metadata:
  annotations:
    CCE.com/IAM: "true"
  creationTimestamp: "2021-06-24T01:30:08Z"
  name: clusterrole_admin_group0c96fad22880f32a3f84c009862af6f7
  namespace: default
  resourceVersion: "36963685"
  selfLink: /apis/rbac.authorization.k8s.io/v1/namespaces/default/rolebindings/clusterrole_admin_group0c96fad22880f32a3f84c009862af6f7
  uid: 6c6f46a6-8584-47da-83f5-9eef1f7b75d6
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: admin
subjects:
- apiGroup: rbac.authorization.k8s.io
  kind: Group
  name: 0c96fad22880f32a3f84c009862af6f7
```

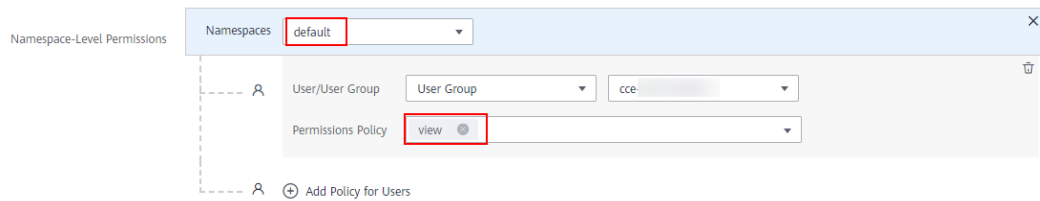
Connect to a cluster as an authorized user. In this example, you can create and query resources in the default namespace, but cannot query resources in the kube-system namespace or cluster resources.

```
# kubectl get pod
NAME                                READY  STATUS   RESTARTS  AGE
test-568d96f4f8-brdrp 1/1    Running  0          33m
test-568d96f4f8-cgjqp 1/1    Running  0          33m
# kubectl get pod -nkube-system
Error from server (Forbidden): pods is forbidden: User "0c97ac3cb280f4d91fa7c0096739e1f8" cannot list resource "pods" in API group "" in the namespace "kube-system"
# kubectl get pv
Error from server (Forbidden): persistentvolumes is forbidden: User "0c97ac3cb280f4d91fa7c0096739e1f8" cannot list resource "persistentvolumes" in API group "" at the cluster scope
```

## Example: Assigning Read-Only Namespace Permissions (view)

The view role has the read-only permissions on a namespace. You can assign permissions to users to view one or multiple namespaces.

**Figure 17-173** Assigning read-only namespace permissions (view)



In the following example `kubectl` output, a `RoleBinding` has been created, the `view` role is bound to the user group **cce-role-group**, and the target namespace is the default namespace.

```
# kubectl get rolebinding
NAME                                ROLE          AGE
clusterrole_view_group0c96fad22880f32a3f84c009862af6f7  ClusterRole/view  7s

# kubectl get rolebinding clusterrole_view_group0c96fad22880f32a3f84c009862af6f7 -oyaml
apiVersion: rbac.authorization.k8s.io/v1
kind: RoleBinding
metadata:
  annotations:
    CCE.com/IAM: "true"
  creationTimestamp: "2021-06-24T01:36:53Z"
  name: clusterrole_view_group0c96fad22880f32a3f84c009862af6f7
  namespace: default
  resourceVersion: "36965800"
  selfLink: /apis/rbac.authorization.k8s.io/v1/namespaces/default/rolebindings/clusterrole_view_group0c96fad22880f32a3f84c009862af6f7
  uid: b86e2507-e735-494c-be55-c41a0c4ef0dd
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: view
subjects:
- apiGroup: rbac.authorization.k8s.io
  kind: Group
  name: 0c96fad22880f32a3f84c009862af6f7
```

Connect to the cluster as an authorized user. In this example, you can query resources in the default namespace but cannot create resources.

```
# kubectl get pod
NAME                                READY STATUS RESTARTS AGE
test-568d96f4f8-brdrp 1/1 Running 0 40m
test-568d96f4f8-cgjqp 1/1 Running 0 40m
# kubectl run -i --tty --image tutum/dnsutils dnsutils --restart=Never --rm /bin/sh
Error from server (Forbidden): pods is forbidden: User "0c97ac3cb280f4d91fa7c0096739e1f8" cannot create resource "pods" in API group "" in the namespace "default"
```

## Example: Assigning Permissions for a Specific Kubernetes Resource Object

You can assign permissions on a specific Kubernetes resource object, such as `pod`, `Deployment`, and `Service`. For details, see [Using kubectl to Configure Namespace Permissions](#).

## 17.16.4 Example: Designing and Configuring Permissions for Users in a Department

### Overview

The conventional distributed task scheduling mode is being replaced by Kubernetes. CCE allows you to easily deploy, manage, and scale containerized applications in the cloud by providing support for you to use Kubernetes.

To help enterprise administrators manage resource permissions in clusters, CCE provides multi-dimensional, fine-grained permission policies and management measures. CCE permissions are described as follows:

- **Cluster-level permissions:** allowing a user group to perform operations on clusters, nodes, node pools, charts, and add-ons. These permissions are assigned based on IAM system policies.
- **Namespace-level permissions:** allowing a user or user group to perform operations on Kubernetes resources, such as workloads, networking, storage, and namespaces. These permissions are assigned based on Kubernetes RBAC.

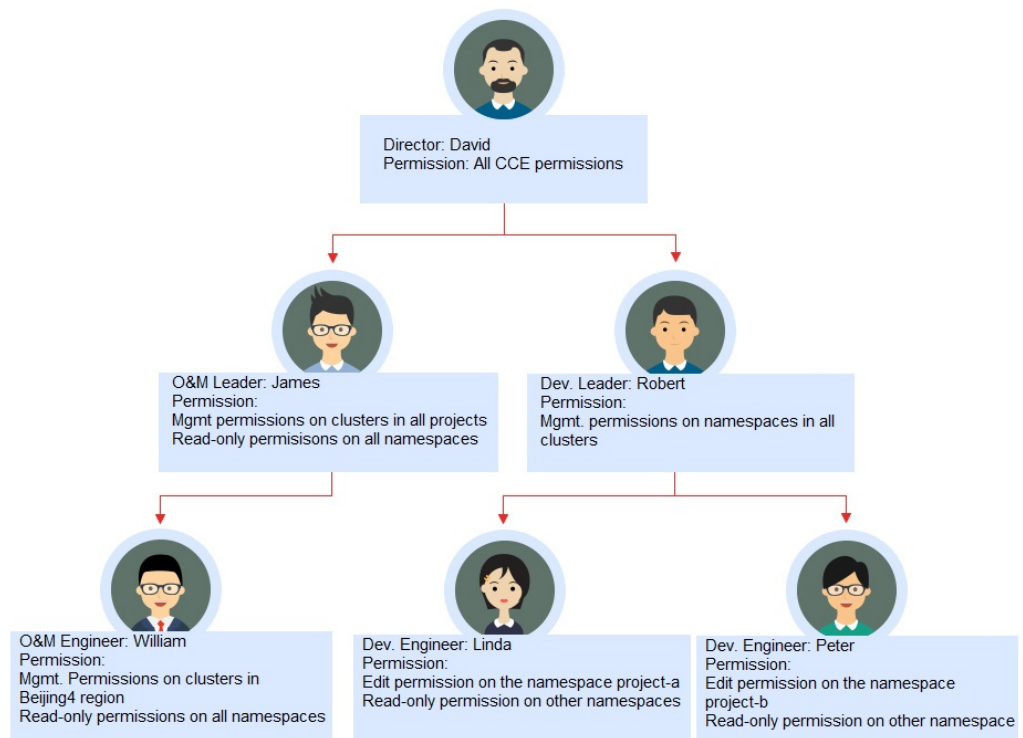
Cluster permissions and namespace permissions are independent of each other but must be used together. The permissions set for a user group apply to all users in the user group. When multiple permissions are added to a user or user group, they take effect at the same time (the union set is used).

### Permission Design

The following uses company X as an example.

Generally, a company has multiple departments or projects, and each department has multiple members. Therefore, you need to design how permissions are to be assigned to different groups and projects, and set a user name for each member to facilitate subsequent user group and permissions configuration.

The following figure shows the organizational structure of a department in a company and the permissions to be assigned to each member:



## Director: David

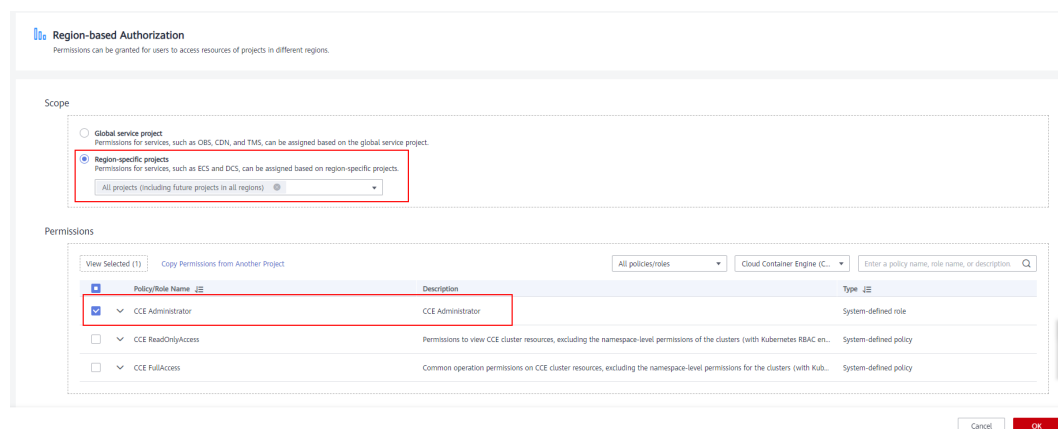
David is a department director of company X. To assign him all CCE permissions (both cluster and namespace permissions), you need to create the **cce-admin** user group for David on the IAM console and assign the CCE Administrator role.

### NOTE

**CCE Administrator:** This role has all CCE permissions. You do not need to assign other permissions.

**CCE FullAccess and CCE ReadOnlyAccess:** These policies are related to cluster management permissions and configured only for cluster-related resources (such as clusters and nodes). You must also configure namespace permissions to perform operations on Kubernetes resources (such as workloads and Services).

**Figure 17-174** Assigning permissions to the user group to which David belongs



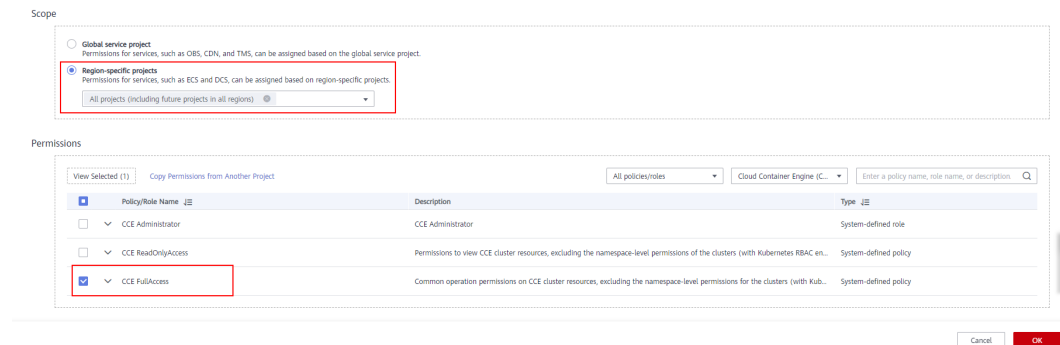


## O&M Leader: James

James is the O&M team leader of the department. He needs the cluster permissions for all projects and the read-only permissions for all namespaces.

To assign the permissions, create a user group named **cce-sre** on the IAM console and add James to this user group. Then, assign CCE FullAccess to the user group **cce-sre** to allow it to perform operations on clusters in all projects.

**Figure 17-175** Assigning permissions to the user group to which James belongs



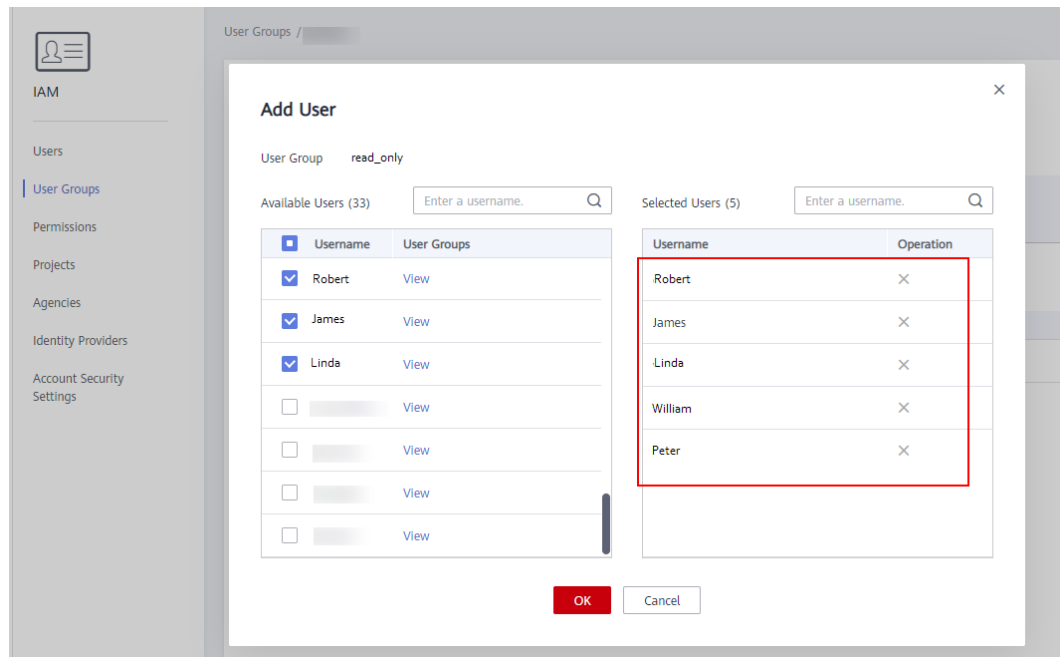
## Assigning Read-only Permissions on All Clusters and Namespaces to All Team Leaders and Engineers

You can create a read-only user group named **read\_only** on the IAM console and add users to the user group.

- Although the development engineers Linda and Peter do not require cluster management permissions, they still need to view data on the CCE console. Therefore, the read-only cluster permission is required.
- For the O&M engineer William, assign the read-only permission on clusters to him in this step.
- The O&M team leader James already has the management permissions on all clusters. You can add him to the **read\_only** user group to assign the read-only permission on clusters to him.

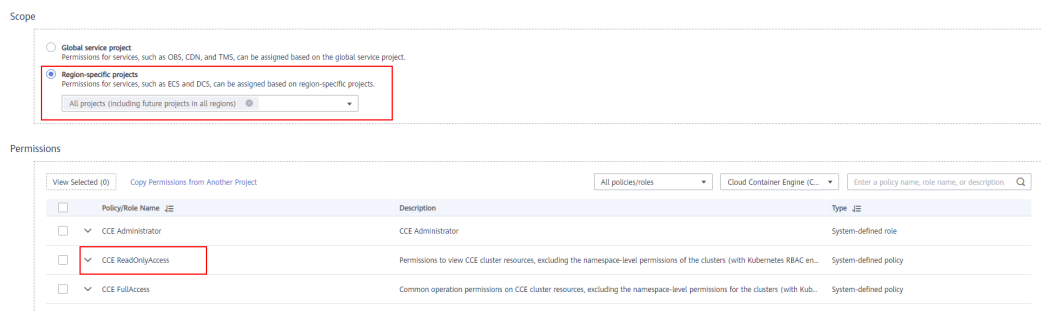
As shown in the following figure, users James, Robert, William, Linda, and Peter are added to the **read\_only** user group.

**Figure 17-176** Adding users to the read\_only user group



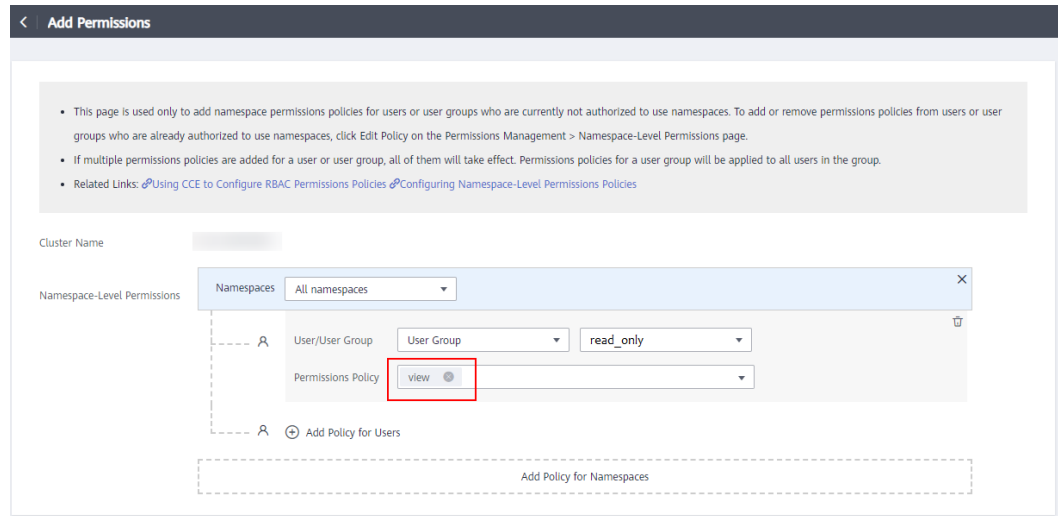
Assign the read-only permission on clusters to the user group **read\_only**.

**Figure 17-177** Assigning the read-only permission on clusters to the user group



Return to the CCE console, and add the read-only permission on namespaces to the user group **read\_only** to which the five users belong. Choose **Permissions Management > Namespace-Level Permissions** on the CCE console, and assign the view policy to the user group **read\_only** for each cluster.

**Figure 17-178** Assigning the read-only permission on namespaces to the user group



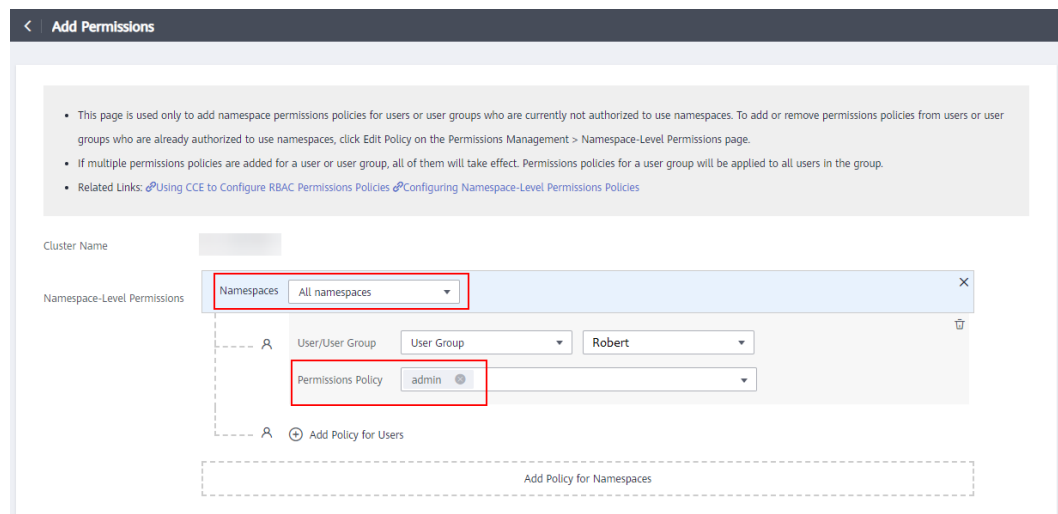
After the setting is complete, James has the cluster management permissions for all projects and the read-only permissions on all namespaces, and the Robert, William, Linda, and Peter have the read-only permission on all clusters and namespaces.

## Development Team Leader: Robert

In the previous steps, Robert has been assigned the read-only permission on all clusters and namespaces. Now, assign the management permissions on all namespaces to Robert.

Log in to the CCE console and choose **Permissions Management > Namespace-Level Permissions** to assign the admin policy to Robert for each cluster.

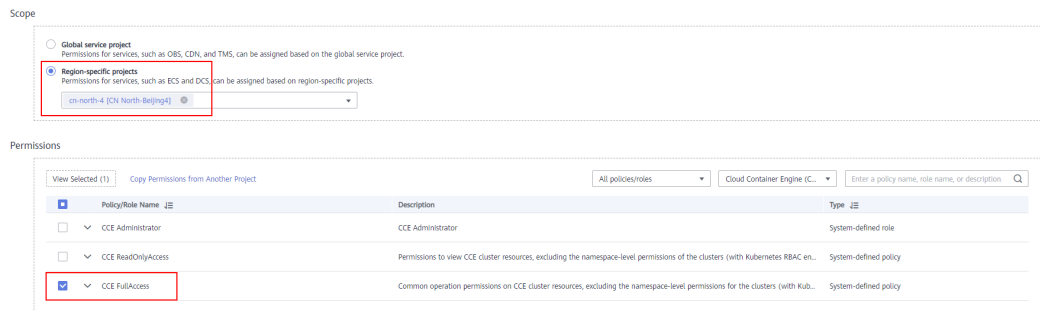
**Figure 17-179** Assigning the management permissions on namespaces to Robert



## O&M Engineer: William

In the previous steps, William has been assigned the read-only permission on all clusters and namespaces. He also requires the cluster management permissions in region Beijing4. Therefore, you can log in to the IAM console, create a user group named **cce-sre-b4** and assign CCE FullAccess to William for region Beijing4.

**Figure 17-180** Assigning the cluster management permissions for Beijing4 region to the user group to which WILLIAM belongs



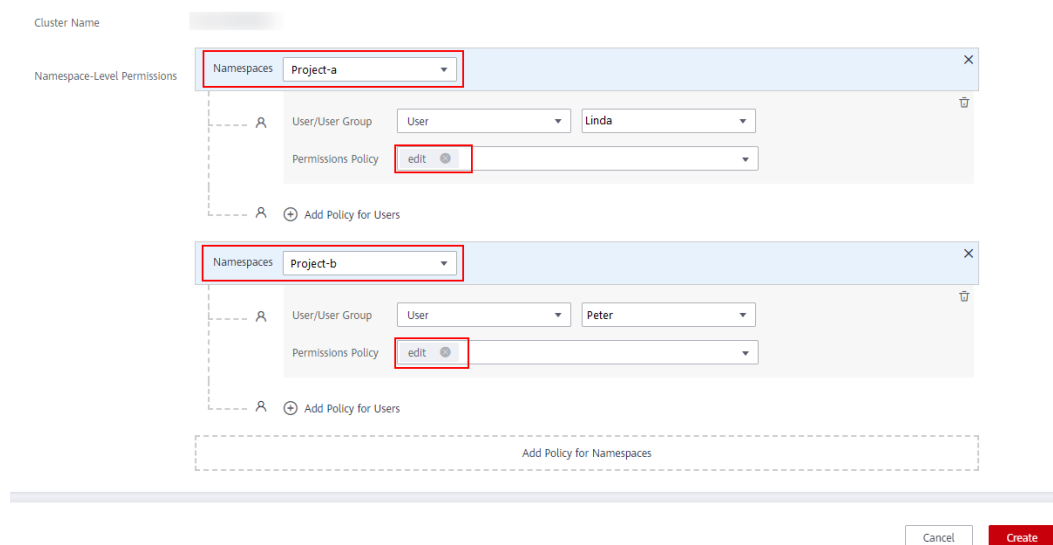
Now, William has the cluster management permissions for Beijing4 region and the read-only permission on all namespaces.

## Development Engineers: Linda and Peter

In the previous steps, Linda and Peter have been assigned the read-only permission on clusters and namespaces. Therefore, you only need to assign the edit policy to them by choosing **Permissions Management > Namespace-Level Permissions** on the CCE console.

**Figure 17-181** Assigning the edit policy on namespaces

- This page is used only to add namespace permissions policies for users or user groups who are currently not authorized to use namespaces. To add or remove permissions policies from users or user groups who are already authorized to use namespaces, click Edit Policy on the Permissions Management > Namespace-Level Permissions page.
- If multiple permissions policies are added for a user or user group, all of them will take effect. Permissions policies for a user group will be applied to all users in the group.
- Related Links: [Using CCE to Configure RBAC Permissions Policies](#) [Configuring Namespace-Level Permissions Policies](#)



By now, all the required permissions are assigned to the department members.

## FAQs

- 1. Can I configure only namespace permissions without cluster management permissions?**

No. The permissions you have to perform operations on the console are determined by the IAM system policy. Therefore, if the cluster management permissions are not configured, you do not have the permissions to access the CCE console.
- 2. Can I use CCE APIs if the cluster management permissions are not configured?**

No. API calls must be authenticated by IAM using tokens.
- 3. Can I use kubectl if the cluster management permissions are not configured?**

Yes. The prerequisite is that the kubectl configuration file has been downloaded from the console. Therefore, if you have configured cluster permissions, after you download the kubectl configuration file from the CCE console and then delete the cluster permissions (reserving the namespace permissions), you can still use kubectl to perform operations on Kubernetes clusters.

## 17.16.5 Permission Dependency of the CCE Console

Some CCE permissions policies depend on the policies of other cloud services. To view or use other cloud resources on the CCE console, you need to enable the system policy access control feature of IAM and assign dependency policies for the other cloud services.

- Dependency policies are assigned based on the CCE FullAccess or CCE ReadOnlyAccess policy you configure. For details, see [Cluster Permissions \(IAM-based\)](#).
- Only users and user groups with namespace permissions can gain the view access to resources in clusters of v1.11.7-r2 and later.
  - If a user is granted the view access to all namespaces of a cluster, the user can view all namespaced resources (except secrets) in the cluster. To view secrets in the cluster, the user must gain the **admin** or **edit** role in all namespaces of the cluster.
  - HPA policies take effect only after the cluster-admin permissions are configured for the namespace.
  - The **view** role within a single namespace allows users to view resources only in the specified namespace.

## Dependency Policy Configuration

To grant an IAM user the permissions to view or use resources of other cloud services on the CCE console, you must first grant the CCE Administrator, CCE FullAccess, or CCE ReadOnlyAccess policy to the user group to which the user belongs and then grant the dependency policies listed in [Table 17-146](#) to the user. These dependency policies will allow the IAM user to access resources of other cloud services.

 NOTE

**Enterprise projects** can group and manage resources across different projects of an enterprise. Resources are thereby isolated. IAM allows you to implement fine-grained authorization. It is strongly recommended that you use IAM for permissions management.

If you use an enterprise project to set permissions for IAM users, the following restrictions apply:

- Storage resources are not available to enterprise projects. Therefore, when an enterprise project user queries the storage resources of clusters in an enterprise project other than **Default** on the CCE console, all buttons on the page are in gray and resources are not displayed.
- On the CCE console, enterprise projects cannot call the API used to obtain AOM monitoring data for cluster monitoring. Therefore, IAM users in these enterprise projects cannot query monitoring data.
- On the CCE console, enterprise projects cannot call the API to query the key pair created during node creation. Therefore, IAM users in these enterprise projects cannot use the key pair login mode. Only the password login mode is supported.
- After assigning the **CCE FullAccess** permission to an enterprise project, you need to configure the **ecs:availabilityZones:list** action on the IAM console so that the enterprise project users can create nodes. Otherwise, the system will prompt that they do not have the permission.

CCE supports fine-grained permissions configuration, but has the following restrictions:

- AOM does not support resource-level monitoring. After operation permissions on specific resources are configured using IAM's fine-grained cluster resource management function, IAM users can view cluster monitoring information on the **Dashboard** page of the CCE console, but cannot view the data on non-fine-grained metrics.

**Table 17-146** Dependency policies

Console Function	Dependent Services	Roles or Policies Required
Dashboard	Application Operations Management (AOM)	<ul style="list-style-type: none"> <li>• An IAM user with CCE Administrator assigned can use this function only after AOM FullAccess policy is assigned.</li> <li>• IAM users with IAM ReadOnlyAccess, CCE FullAccess, or CCE ReadOnlyAccess assigned can directly use this function.</li> </ul>

Console Function	Dependent Services	Roles or Policies Required
Workload management	Elastic Load Balance (ELB) Application Performance Management (APM) Application Operations Management (AOM) NAT Gateway Object Storage Service (OBS) Scalable File Service (SFS)	<p>Except in the following cases, the user does not require any additional role to create workloads.</p> <ul style="list-style-type: none"> <li>To create a Service using ELB, you must have ELB FullAccess or ELB Administrator plus VPC Administrator assigned.</li> <li>To use a Java probe, you must have AOM FullAccess and APM FullAccess assigned.</li> <li>To create a Service using NAT Gateway, you must have NAT Gateway Administrator assigned.</li> <li>To use OBS, you must have OBS Administrator globally assigned.</li> </ul> <p><b>NOTE</b> Because of the cache, it takes about 13 minutes for the RBAC policy to take effect after being granted to users, user groups, and enterprise projects. After an OBS-related system policy is granted, it takes about 5 minutes for the policy to take effect.</p> <ul style="list-style-type: none"> <li>To use SFS, you must have SFS FullAccess assigned.</li> </ul>
Cluster management	Application Operations Management (AOM) Billing Center (BSS)	<ul style="list-style-type: none"> <li>Auto scale-out or scale-up requires the AOM FullAccess policy.</li> <li>Changing the billing mode to yearly/monthly requires the BSS Administrator role.</li> </ul>
Node management	Elastic Cloud Server (ECS)	If the permission assigned to an IAM user is CCE Administrator, creating or deleting a node requires the ECS FullAccess or ECS Administrator policy and the VPC Administrator policy.
Network management	Elastic Load Balance (ELB) NAT Gateway	<p>Except in the following cases, the user does not require any additional role to create a Service.</p> <ul style="list-style-type: none"> <li>To create a Service using ELB, you must have ELB FullAccess or ELB Administrator plus VPC Administrator assigned.</li> <li>To create a Service using NAT Gateway, you must have NAT Administrator assigned.</li> </ul>

Console Function	Dependent Services	Roles or Policies Required
Storage management	Object Storage Service (OBS) Scalable File Service (SFS)	<ul style="list-style-type: none"> <li>To use OBS, you must have OBS Administrator globally assigned.</li> </ul> <p><b>NOTE</b> Because of the cache, it takes about 13 minutes for the RBAC policy to take effect after being granted to users, user groups, and enterprise projects. After an OBS-related system policy is granted, it takes about 5 minutes for the policy to take effect.</p> <ul style="list-style-type: none"> <li>To use SFS, you must have SFS FullAccess assigned.</li> <li>Using SFS Turbo requires the SFS Turbo Admin role.</li> </ul> <p>The CCE Administrator role is required for importing storage devices.</p>
Namespace management	/	/
Chart management	/	IAM users with CCE Administrator assigned can use this function.
Add-on management	/	IAM users with CCE Administrator, CCE FullAccess, or CCE ReadOnlyAccess assigned can use this function.
Permissions management	/	<ul style="list-style-type: none"> <li>IAM users with CCE Administrator or global Security Administrator assigned can use this function.</li> <li>IAM users with IAM ReadOnlyAccess plus CCE FullAccess or CCE ReadOnlyAccess assigned can use this function.</li> </ul>
Configuration center	/	<ul style="list-style-type: none"> <li>Creating ConfigMaps does not require any additional policy.</li> <li>Viewing secrets requires that the cluster-admin, admin, or edit permission be configured for the namespace. The DEW KeypairFullAccess or DEW KeypairReadOnlyAccess policy must be assigned for dependent services.</li> </ul>
Help center	/	/



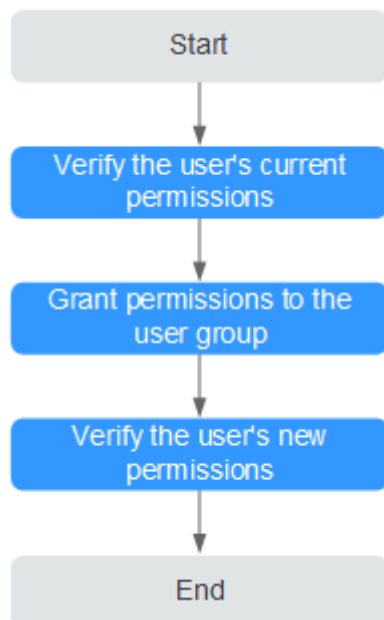
Console Function	Dependent Services	Roles or Policies Required
Switching to other related services	Software Repository for Container (SWR) Application Operations Management (AOM) Cloud Eye (for viewing metrics of nodes and storage systems)	The CCE console provides links to other related services. To view or use these services, an IAM user must be assigned required permissions for the services.

### Example

This example describes how to authorize the user James to view monitoring information on the **Dashboard** page of the CCE console.

Assume that the IAM user James belongs to the user group Developers and Developers is already granted the CCE Administrator role. According to [Table 17-146](#), to view monitoring information on the **Dashboard** page of the CCE console, the AOM FullAccess policy is additionally required.

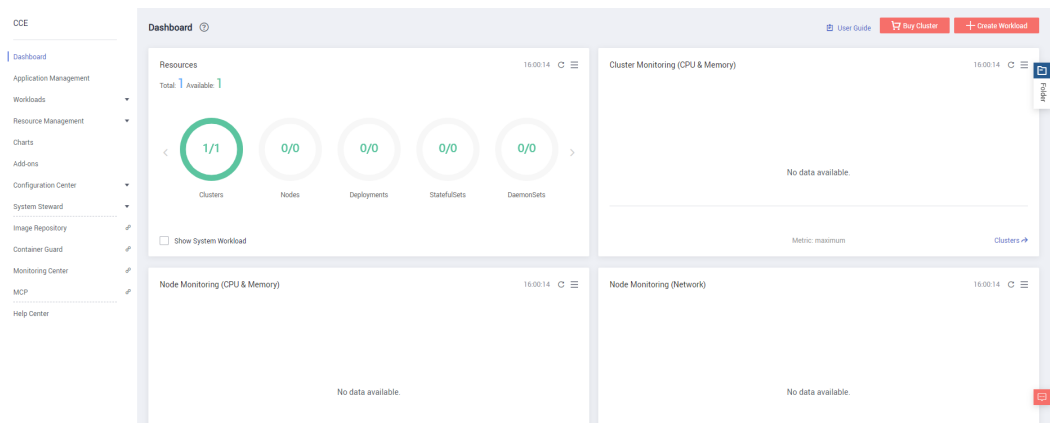
**Figure 17-182** Process for granting dependency policies



### Step 1: Verify User's Current Permissions

- Step 1** Log in to the CCE console as the IAM user **James**.
- Step 2** In the navigation pane, click **Dashboard**. The monitoring information on the **Dashboard** page is not available to the user James.

**Figure 17-183** Viewing monitoring information (no data available)



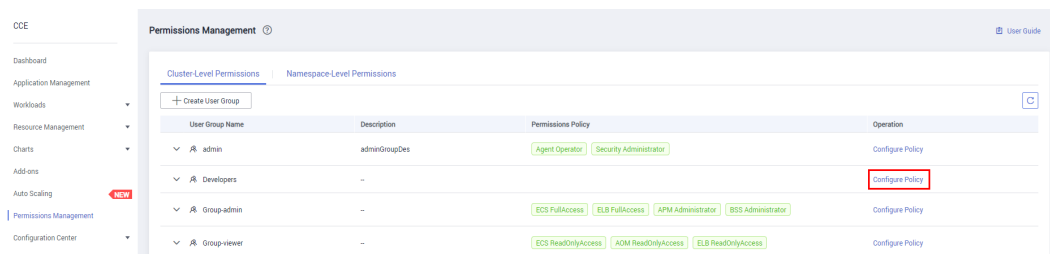
----End

## Step 2: Grant Permissions to the User Group

**Step 1** Log out as IAM user James and log in to the CCE console using your account.

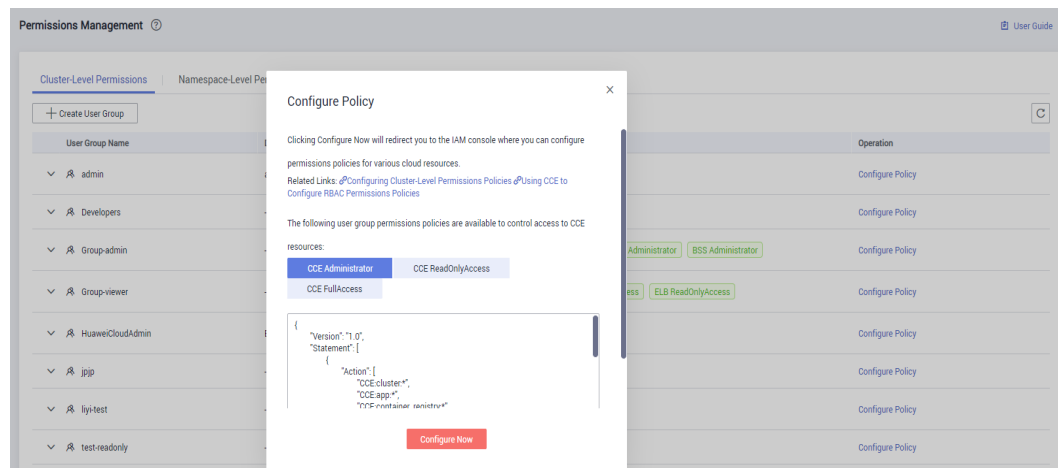
**Step 2** In the navigation pane, choose **Permissions Management**. Click the **Cluster-Level Permissions** tab. In the same row as the user group Developers to which the IAM user James belongs, click **Configure Policy**.

**Figure 17-184** Granting permissions to a user group



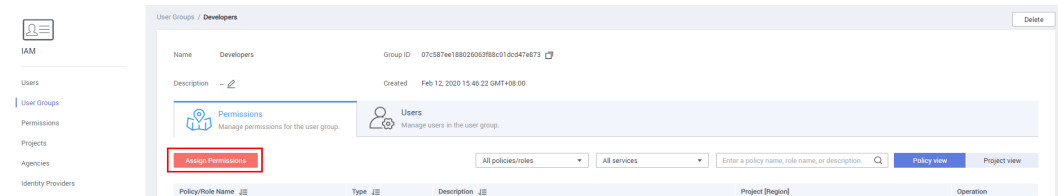
**Step 3** In the **Configure Policy** dialog box, click **Configure Now**.

**Figure 17-185** Configuring a permissions policy



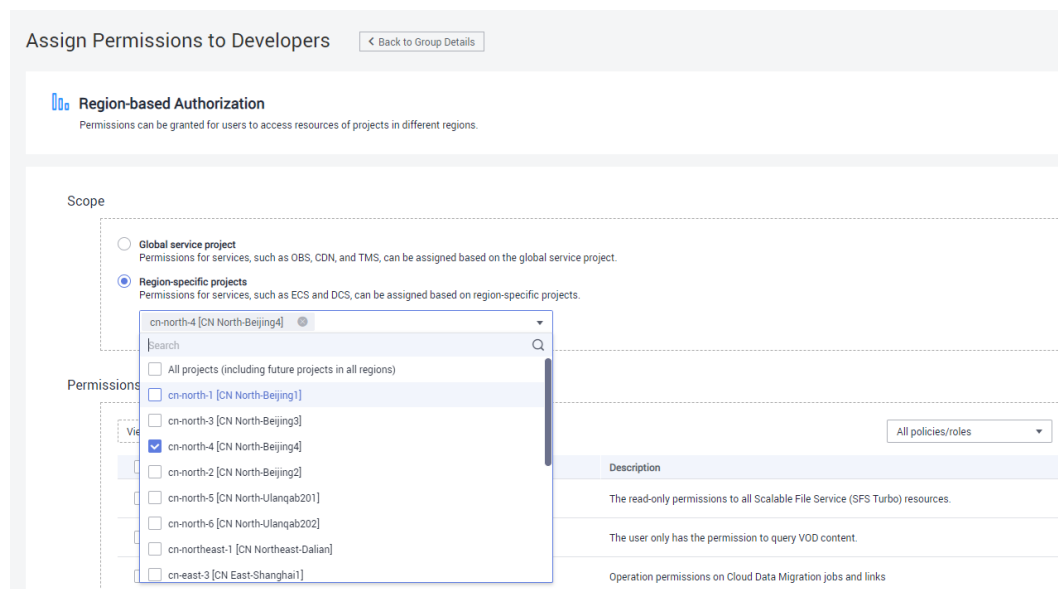
**Step 4** In the navigation pane of the IAM console, choose **User Groups**. In the user group list, click the user group name **Developers** to show user group details. On the **Permissions** tab page, click **Assign Permissions**.

**Figure 17-186** Granting permissions to a user group



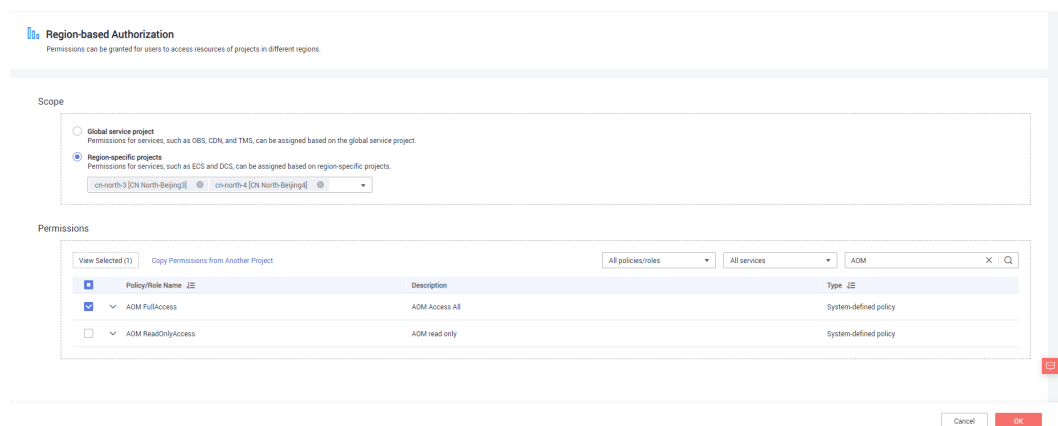
**Step 5** On the page displayed, select the application scope. In this case, select **Region-specific projects** and then one or more projects in the drop-down list.

**Figure 17-187** Selecting a project type for authorization



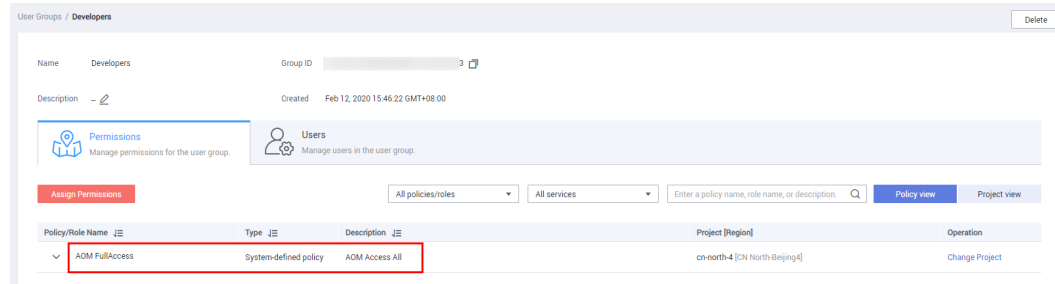
**Step 6** Search for AOM in the search box above the permissions list, select the **AOM FullAccess** policy, and click **OK**.


**Figure 17-188** Adding an AOM FullAccess policy



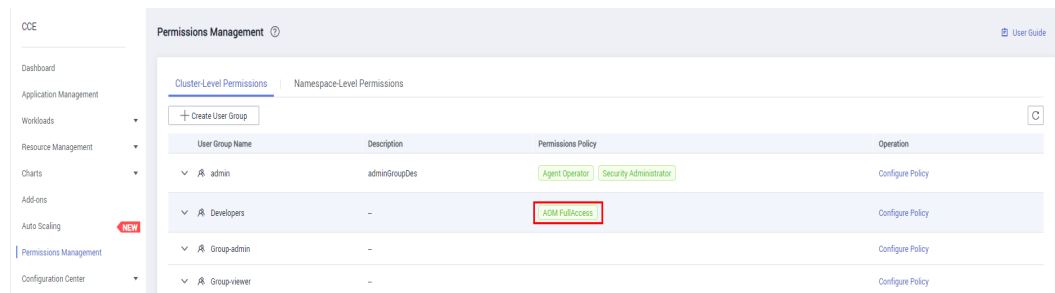
**Step 7** Verify that the AOM FullAccess policy is displayed on the **Permissions** tab page.

**Figure 17-189** Verifying whether a permissions policy is successfully added



**Step 8** In the navigation pane of the CCE console, choose **Permissions Management**. Click the **Cluster-Level Permissions** tab. In the upper right corner of the page, click the  icon to refresh the page. The AOM FullAccess policy is displayed in the policy list.

**Figure 17-190** Success in adding a policy



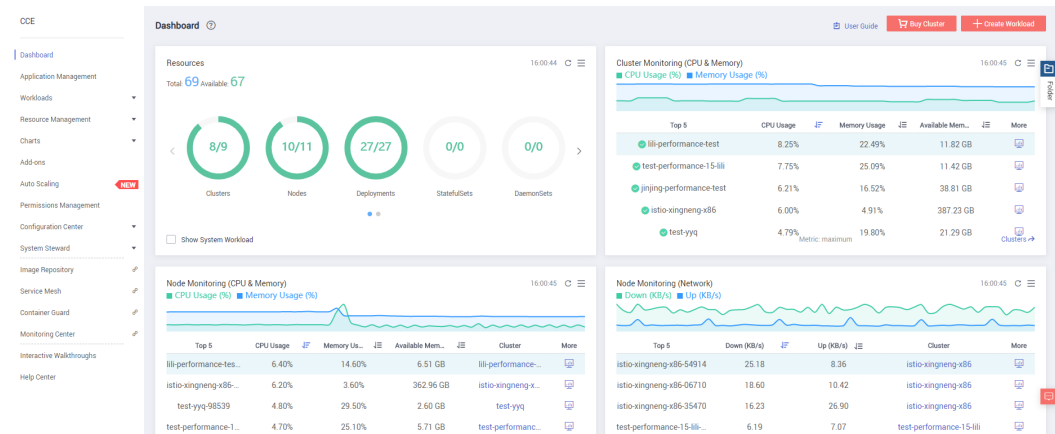
----End

### Step 3: Verify User's New Permissions

**Step 1** Log in to the CCE console as the IAM user **James**.

**Step 2** In the navigation pane, choose **Dashboard**. The monitoring information on the **Dashboard** page is now available to James.

**Figure 17-191** Monitoring information on the Dashboard page



**Step 3** Verify that the **Dashboard** page also displays information about dependent services to James.

----End

## 17.16.6 Pod Security

### 17.16.6.1 Configuring a Pod Security Policy

A pod security policy (PSP) is a cluster-level resource that controls sensitive security aspects of the pod specification. The **PodSecurityPolicy** object in Kubernetes defines a group of conditions that a pod must comply with to be accepted by the system, as well as the default values of related fields.

By default, the PSP access control component is enabled for clusters of v1.17.17 and a global default PSP named **psp-global** is created. You can modify the default policy (but not delete it). You can also create a PSP and bind it to the RBAC configuration.

#### NOTE

- In addition to the global default PSP, the system configures independent PSPs for system components in namespace kube-system. Modifying the psp-global configuration does not affect pod creation in namespace kube-system.
- PodSecurityPolicy was deprecated in Kubernetes v1.21, and removed from Kubernetes in v1.25. You can use pod security admission as a substitute for PodSecurityPolicy. For details, see [Configuring Pod Security Admission](#).

## Modifying the Global Default PSP

Before modifying the global default PSP, ensure that a CCE cluster has been created and connected by using kubectl.

**Step 1** Run the following command:

```
kubectl edit psp psp-global
```

**Step 2** Modify the required parameters, as shown in [Table 17-147](#).

**Table 17-147** PSP configuration

Item	Description
privileged	Starts the privileged container.
hostPID hostIPC	Uses the host namespace.
hostNetwork hostPorts	Uses the host network and port.
volumes	Specifies the type of the mounted volume that can be used.

Item	Description
allowedHostPaths	Specifies the host path to which a hostPath volume can be mounted. The <b>pathPrefix</b> field specifies the host path prefix group to which a hostPath volume can be mounted.
allowedFlexVolumes	Specifies the FlexVolume driver that can be used.
fsGroup	Configures the supplemental group ID used by the mounted volume in the pod.
readOnlyRootFilesystem	Pods can only be started using a read-only root file system.
runAsUser runAsGroup supplementalGroups	Specifies the user ID, primary group ID, and supplemental group ID for starting containers in a pod.
allowPrivilegeEscalation defaultAllowPrivilegeEscalation	Specifies whether <b>allowPrivilegeEscalation</b> can be set to <b>true</b> in a pod. This configuration controls the use of Setuid and whether programs can use additional privileged system calls.
defaultAddCapabilities requiredDropCapabilities allowedCapabilities	Controls the Linux capabilities used in pods.
seLinux	Controls the configuration of seLinux used in pods.
allowedProcMountTypes	Controls the ProcMountTypes that can be used by pods.
annotations	Configures AppArmor or Seccomp used by containers in a pod.
forbiddenSysctls allowedUnsafeSysctls	Controls the configuration of Sysctl used by containers in a pod.

----End

## Example of Enabling Unsafe Sysctls in Pod Security Policy

You can configure allowed-unsafe-sysctls for a node pool. For CCE clusters of **v1.17.17** and later versions, add configurations in **allowedUnsafeSysctls** of the pod security policy to make the configuration take effect. For details, see [Table 17-147](#).

In addition to modifying the global pod security policy, you can add new pod security policies. For example, enable the **net.core.somaxconn** unsafe sysctls. The following is an example of adding a pod security policy:

```
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:
- '*'
---
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

```

## Restoring the Original PSP

If you have modified the default pod security policy and want to restore the original pod security policy, perform the following operations.

- Step 1** Create a policy description file named **policy.yaml**. **policy.yaml** is an example file name. You can rename it as required.

### vi policy.yaml

The content of the description file is as follows:

```
apiVersion: policy/v1beta1
kind: PodSecurityPolicy
metadata:
  name: psp-global
  annotations:
    seccomp.security.alpha.kubernetes.io/allowedProfileNames: '*'
spec:
  privileged: true
  allowPrivilegeEscalation: true
  allowedCapabilities:
    - '*'
  volumes:
    - '*'
  hostNetwork: true
  hostPorts:
    - min: 0
      max: 65535
  hostIPC: true
  hostPID: true
  runAsUser:
    rule: 'RunAsAny'
  seLinux:
    rule: 'RunAsAny'
  supplementalGroups:
    rule: 'RunAsAny'
  fsGroup:
    rule: 'RunAsAny'
---
kind: ClusterRole
apiVersion: rbac.authorization.k8s.io/v1
metadata:
  name: psp-global
rules:
  - apiGroups:
    - '*'
    resources:
    - podsecuritypolicies
    resourceNameNames:
    - psp-global
    verbs:
    - use
---
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  name: psp-global
roleRef:
  kind: ClusterRole
  name: psp-global
  apiGroup: rbac.authorization.k8s.io
subjects:
  - kind: Group
    name: system:authenticated
    apiGroup: rbac.authorization.k8s.io
```

**Step 2** Run the following command:

```
kubectl apply -f policy.yaml
```

----End

### 17.16.6.2 Configuring Pod Security Admission

Before using [pod security admission](#), understand Kubernetes [Pod Security Standards](#). These standards define different isolation levels for pods. They let you define how you want to restrict the behavior of pods in a clear, consistent fashion.



Kubernetes offers a built-in pod security admission controller to enforce the pod security standards. Pod security restrictions are applied at the namespace level when pods are created.

The pod security standard defines three security policy levels:

**Table 17-148** Pod security policy levels

Level	Description
privileged	Unrestricted policy, providing the widest possible level of permissions, typically aimed at system- and infrastructure-level workloads managed by privileged, trusted users, such as CNIs and storage drivers.
baseline	Minimally restrictive policy that prevents known privilege escalations, typically targeted at non-critical workloads. This policy disables capabilities such as hostNetwork and hostPID.
restricted	Heavily restricted policy, following current Pod hardening best practices.

**Pod security admission** is applied at the namespace level. The controller restricts the security context and other parameters in the pod or container in the namespace. The privileged policy does not verify the **securityContext** field of the pod and container. The baseline and restricted policies have different requirements on **securityContext**. For details, see [Pod Security Standards](#).

Setting security context: [Configure a Security Context for a Pod or Container](#)

## Pod Security Admission Labels

Kubernetes defines three types of labels for pod security admission (see [Table 17-149](#)). You can set these labels in a namespace to define the pod security standard level to be used. However, do not change the pod security standard level in system namespaces such as kube-system. Otherwise, pods in the system namespace may be faulty.

**Table 17-149** Pod security admission labels

Mode	Target Object	Description
enforce	Pods	Policy violations will cause the pod to be rejected.
audit	Workloads (such as Deployment and job)	Policy violations will trigger the addition of an audit annotation to the event recorded in the audit log, but are otherwise allowed.
warn	Workloads (such as Deployment and job)	Policy violations will trigger a user-facing warning, but are otherwise allowed.

 NOTE

Pods are often created indirectly, by creating a workload object such as a Deployment or job. To help catch violations early, both the audit and warning modes are applied to the workload resources. However, the enforce mode is applied only to the resulting pod objects.

## Enforcing Pod Security Admission with Namespace Labels

You can label namespaces to enforce pod security standards. Assume that a namespace is configured as follows:

```
apiVersion: v1
kind: Namespace
metadata:
  name: my-baseline-namespace
  labels:
    pod-security.kubernetes.io/enforce: privileged
    pod-security.kubernetes.io/enforce-version: v1.25
    pod-security.kubernetes.io/audit: baseline
    pod-security.kubernetes.io/audit-version: v1.25
    pod-security.kubernetes.io/warn: restricted
    pod-security.kubernetes.io/warn-version: v1.25

# The label can be in either of the following formats:
# pod-security.kubernetes.io/<MODE>: <LEVEL>
# pod-security.kubernetes.io/<MODE>-version: <VERSION>
# The audit and warn modes inform you of which security behaviors are violated by the load.
```

Namespace labels indicate which policy level to apply for the mode. For each mode, there are two labels that determine the policy used:

- `pod-security.kubernetes.io/<MODE>: <LEVEL>`
  - `<MODE>`: must be **enforce**, **audit**, or **warn**. For details about the modes, see [Table 17-149](#).
  - `<LEVEL>`: must be **privileged**, **baseline**, or **restricted**. For details about the levels, see [Table 17-148](#).
- `pod-security.kubernetes.io/<MODE>-version: <VERSION>`

Optional, which pins the policy to a given Kubernetes version.

  - `<MODE>`: must be **enforce**, **audit**, or **warn**. For details about the modes, see [Table 17-149](#).
  - `<VERSION>`: Kubernetes version number. For example, v1.25. You can also use **latest**.

If pods are deployed in the preceding namespace, the following security restrictions apply:

1. The verification in the enforce mode is skipped (enforce mode + privileged level).
2. Restrictions related to the baseline policy are verified (audit mode + baseline level). That is, if the pod or container violates the policy, the corresponding event is recorded into the audit log.
3. Restrictions related to the restricted policy are verified (warn mode + restricted level). That is, if the pod or container violates the policy, the user will receive an alarm when creating the pod.

## Migrating from Pod Security Policy to Pod Security Admission

If you use pod security policies in a cluster earlier than v1.25 and need to replace them with pod security admission in a cluster of v1.25 or later, follow the guide in [Migrate from PodSecurityPolicy to the Built-In PodSecurity Admission Controller](#).

### NOTICE

1. Pod security admission supports only three isolation modes, less flexible than pod security policies. If you require more control over specific constraints, you will need to use a Validating Admission Webhook to enforce those policies.
2. Pod security admission is a non-mutating admission controller, meaning it will not modify pods before validating them. If you were relying on this aspect of PSP, you will need to either modify the security context in your workloads, or use a Mutating Admission Webhook to make those changes.
3. PSP lets you bind different policies to different service accounts. This approach has many pitfalls and is not recommended, but if you require this feature anyway you will need to use a third-party webhook instead.
4. Do not apply pod security admission to namespaces where CCE components, such as kube-system, kube-public, and kube-node-lease, are deployed. Otherwise, CCE components and add-on functions will be abnormal.

## Documentation

- [Pod Security Admission](#)
- [Mapping PodSecurityPolicies to Pod Security Standards](#)
- [Enforce Pod Security Standards with Namespace Labels](#)
- [Enforce Pod Security Standards by Configuring the Built-in Admission Controller](#)

## 17.16.7 Service Account Token Security Improvement

In clusters earlier than v1.21, a token is obtained by mounting the secret of the service account to a pod. Tokens obtained this way are permanent. This approach is no longer recommended starting from version 1.21. Service accounts will stop auto creating secrets in clusters from version 1.25.

In clusters of version 1.21 or later, you can use the [TokenRequest](#) API to obtain the token and use the projected volume to mount the token to the pod. Such tokens are valid for a fixed period (one hour by default). Before expiration, kubelet refreshes the token to ensure that the pod always uses a valid token. When the mounting pod is deleted, the token automatically becomes invalid. This approach is implemented by the [BoundServiceAccountTokenVolume](#) feature to improve the token security of the service account. Kubernetes clusters of v1.21 and later enable this approach by default.

For smooth transition, the community extends the token validity period to one year by default. After one year, the token becomes invalid, and clients that do not support certificate reloading cannot access the API server. It is recommended that

clients of earlier versions be upgraded as soon as possible. Otherwise, service faults may occur.

If you use a Kubernetes client of a to-be-outdated version, the certificate reloading may fail. Versions of officially supported Kubernetes client libraries able to reload tokens are as follows:

- Go:  $\geq$  v0.15.7
- Python:  $\geq$  v12.0.0
- Java:  $\geq$  v9.0.0
- Javascript:  $\geq$  v0.10.3
- Ruby: master branch
- Haskell: v0.3.0.0
- C#:  $\geq$  7.0.5

For details, visit <https://github.com/kubernetes/enhancements/tree/master/keps/sig-auth/1205-bound-service-account-tokens>.

#### NOTE

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

## Diagnosis

Perform the following steps to check your CCE clusters of v1.21 or later:

1. Use kubectl to access the cluster and run the **kubectl get --raw "/metrics" | grep stale** command to obtain the metrics. Check the metric named **serviceaccount\_stale\_tokens\_total**.

If the value is greater than 0, some workloads in the cluster may be using an earlier client-go version. In this case, check whether this problem occurs in your deployed applications. If yes, upgrade client-go to the version specified by the community as soon as possible. The version must be at least two major versions of the CCE cluster. For example, if your cluster version is 1.23, the Kubernetes dependency library version must be at least 1.19.

```
[root@ ~]# kubectl get --raw "/metrics" | grep stale
# HELP serviceaccount_stale_tokens_total [ALPHA] Cumulative stale projected service account tokens used
# TYPE serviceaccount_stale_tokens_total counter
serviceaccount_stale_tokens_total 52
```

## 17.17 Cloud Trace Service (CTS)

### 17.17.1 CCE Operations Supported by CTS

Cloud Trace Service (CTS) records operations on cloud service resources, allowing users to query, audit, and backtrack the resource operation requests initiated from the management console or open APIs as well as responses to the requests.

**Table 17-150** CCE operations supported by CTS

Operation	Resource Type	Event Name
Creating an agency	Cluster	createUserAgencies
Creating a cluster	Cluster	createCluster
Creating a yearly/ monthly-billed cluster	Cluster	createCluster/createPeriodicCluster
Updating the description of a cluster	Cluster	updateCluster
Upgrading a cluster	Cluster	clusterUpgrade
Deleting a cluster	Cluster	claimCluster/deleteCluster
Downloading a cluster certificate	Cluster	getClusterCertByUID
Binding and unbinding an EIP	Cluster	operateMasterEIP
Waking up a cluster and resetting node management (V2)	Cluster	operateCluster
Hibernating a cluster (V3)	Cluster	hibernateCluster
Waking up a cluster (V3)	Cluster	awakeCluster
Changing the specifications of a pay-per-use cluster	Cluster	resizeCluster
Changing the specifications of a yearly/monthly- billed cluster	Cluster	resizePeriodCluster
Modifying configurations of a cluster	Cluster	updateConfiguration
Creating a node pool	Node pool	createNodePool
Updating a node pool	Node pool	updateNodePool
Deleting a node pool	Node pool	claimNodePool
Migrating a node pool	Node pool	migrateNodepool

Operation	Resource Type	Event Name
Modifying node pool configurations	Node pool	updateConfiguration
Creating a node	Node	createNode
Deleting all the nodes from a specified cluster	Node	deleteAllHosts
Deleting a single node	Node	deleteOneHost/claimOneHost
Updating the description of a node	Node	updateNode
Creating an add-on instance	Add-on instance	createAddonInstance
Deleting an add-on instance	Add-on instance	deleteAddonInstance
Uploading a chart	Chart	uploadChart
Updating a chart	Chart	updateChart
Deleting a chart	Chart	deleteChart
Creating a release	Release	createRelease
Upgrading a release	Release	updateRelease
Deleting a release	Release	deleteRelease
Creating a ConfigMap	Kubernetes resource	createConfigmaps
Creating a DaemonSet	Kubernetes resource	createDaemonsets
Creating a Deployment	Kubernetes resource	createDeployments
Creating an event	Kubernetes resource	createEvents
Creating an Ingress	Kubernetes resource	createIngresses
Creating a job	Kubernetes resource	createJobs
Creating a namespace	Kubernetes resource	createNamespaces
Creating a node	Kubernetes resource	createNodes

Operation	Resource Type	Event Name
Creating a PersistentVolume-Claim	Kubernetes resource	createPersistentvolumeclaims
Creating a pod	Kubernetes resource	createPods
Creating a replica set	Kubernetes resource	createReplicasets
Creating a resource quota	Kubernetes resource	createResourcequotas
Creating a secret	Kubernetes resource	createSecrets
Creating a service	Kubernetes resource	createServices
Creating a StatefulSet	Kubernetes resource	createStatefulsets
Creating a volume	Kubernetes resource	createVolumes
Deleting a ConfigMap	Kubernetes resource	deleteConfigmaps
Deleting a DaemonSet	Kubernetes resource	deleteDaemonsets
Deleting a Deployment	Kubernetes resource	deleteDeployments
Deleting an event	Kubernetes resource	deleteEvents
Deleting an Ingress	Kubernetes resource	deleteIngresses
Deleting a job	Kubernetes resource	deleteJobs
Deleting a namespace	Kubernetes resource	deleteNamespaces
Deleting a node	Kubernetes resource	deleteNodes
Deleting a Pod	Kubernetes resource	deletePods
Deleting a replica set	Kubernetes resource	deleteReplicasets

Operation	Resource Type	Event Name
Deleting a resource quota	Kubernetes resource	deleteResourcequotas
Deleting a secret	Kubernetes resource	deleteSecrets
Deleting a service	Kubernetes resource	deleteServices
Deleting a StatefulSet	Kubernetes resource	deleteStatefulsets
Deleting volumes	Kubernetes resource	deleteVolumes
Replacing a specified ConfigMap	Kubernetes resource	updateConfigmaps
Replacing a specified DaemonSet	Kubernetes resource	updateDaemonsets
Replacing a specified Deployment	Kubernetes resource	updateDeployments
Replacing a specified event	Kubernetes resource	updateEvents
Replacing a specified ingress	Kubernetes resource	updateIngresses
Replacing a specified job	Kubernetes resource	updateJobs
Replacing a specified namespace	Kubernetes resource	updateNamespaces
Replacing a specified node	Kubernetes resource	updateNodes
Replacing a specified PersistentVolume-Claim	Kubernetes resource	updatePersistentvolumeclaims
Replacing a specified pod	Kubernetes resource	updatePods
Replacing a specified replica set	Kubernetes resource	updateReplicasets
Replacing a specified resource quota	Kubernetes resource	updateResourcequotas
Replacing a specified secret	Kubernetes resource	updateSecrets




Operation	Resource Type	Event Name
Replacing a specified service	Kubernetes resource	updateServices
Replacing a specified StatefulSet	Kubernetes resource	updateStatefulsets
Replacing the specified status	Kubernetes resource	updateStatus
Uploading a chart	Kubernetes resource	uploadChart
Updating a component template	Kubernetes resource	updateChart
Deleting a chart	Kubernetes resource	deleteChart
Creating a template application	Kubernetes resource	createRelease
Updating a template application	Kubernetes resource	updateRelease
Deleting a template application	Kubernetes resource	deleteRelease

## 17.17.2 Querying CTS Logs

### Scenario

After you enable CTS, the system starts recording operations on CCE resources. Operation records of the last 7 days can be viewed on the CTS management console.

### Procedure

- Step 1** Log in to the management console.
- Step 2** Click  in the upper left corner and select a region.
- Step 3** Choose **Service List** from the main menu. Choose **Management & Deployment > Cloud Trace Service**.
- Step 4** In the navigation pane of the CTS console, choose **Cloud Trace Service > Trace List**.
- Step 5** On the **Trace List** page, query operation records based on the search criteria. Currently, the trace list supports trace query based on the combination of the following search criteria:
  - **Trace Source, Resource Type, and Search By**


Select the search criteria from the drop-down lists. Select **CCE** from the **Trace Source** drop-down list.

If you select **Trace name** from the **Search By** drop-down list, specify the trace name.

If you select **Resource ID** from the **Search By** drop-down list, select or enter a specific resource ID.

If you select **Resource name** from the **Search By** drop-down list, select or enter a specific resource name.

- **Operator:** Select a specific operator (at user level rather than account level).
- **Trace Status:** Set this parameter to any of the following values: **All trace statuses, normal, warning, and incident.**
- **Time range:** You can query traces generated during any time range in the last seven days.

**Step 6** Click  on the left of a trace to expand its details.

**Figure 17-192** Expanding trace details

Trace Name	Resource Ty...	Trace So...	Resource ID ?	Resource Na...	Trace Stat...	Operator ?	Operation Time	Operation
operateClus...	clusters-acti...	CCE			<span style="color: green;">●</span> normal	xuchun...	Oct 11, 2018 09:24:56 GMT...	<a href="#">View Trace</a>
Trace ID	7660c4e0-ccf4-11e8-9fe5-286ed488cbe3		Source IP Address	58.213.108.72				
Trace Type	ConsoleAction		Generated	Oct 11, 2018 09:24:56 GMT+08:00				

**Step 7** Click **View Trace** in the **Operation** column. The trace details are displayed.

**Figure 17-193** Viewing event details

```
{
  "service_type": "CCE",
  "user": {
    "domain": {
      "name": "xuchun@501",
      "id": "cc9c0076188843ea938743dd166c7fef"
    },
    "id": "69d8a0dc65e2436eb973093b49bb5ecb",
    "name": "xuchun@501"
  },
  "time": "2018/10/11 09:24:56 GMT+08:00",
  "code": 200,
  "resource_type": "clusters-action",
  "source_ip": "58.213.108.72",
  "trace_name": "operateCluster",
  "trace_type": "ConsoleAction",
  "message": "UserName: xuchun@501; Operation : POST /api/v2/projects/6244f46518ab48d4ba84b3bcb93aba67/clusters/3cc",
  "record_time": "2018/10/11 09:24:56 GMT+08:00",
  "trace_id": "7660c4e0-ccf4-11e8-9fe5-286ed488cbe3",
  "trace_status": "normal"
}
```

----End