

Cloud Container Engine

User Guide

Date 2024-01-23

Contents

1 Service Overview.....	1
1.1 What Is Cloud Container Engine?.....	1
1.2 Product Advantages.....	2
1.3 Application Scenarios.....	6
1.3.1 Infrastructure and Containerized Application Management.....	6
1.3.2 Auto Scaling in Seconds.....	8
1.3.3 DevOps and CI/CD.....	8
1.3.4 Hybrid Cloud Architecture.....	10
1.4 Constraints.....	12
1.5 Permissions.....	16
1.6 Pricing Details.....	23
1.7 Basic Concepts.....	24
1.7.1 Basic Concepts.....	24
1.7.2 Mappings Between CCE and Kubernetes Terms.....	31
1.7.3 CCE Turbo Cluster.....	32
1.7.4 Regions and AZs.....	34
1.8 Related Services.....	35
2 Product Bulletin.....	37
2.1 Risky Operations on Cluster Nodes.....	37
2.2 CCE Security Guide.....	38
2.3 Cluster Node OS Patch Notes.....	40
2.4 Vulnerability Notice.....	41
2.4.1 Notice on the Kubernetes Security Vulnerability (CVE-2022-3172).....	41
2.4.2 Privilege Escalation Vulnerability in Linux openvswitch Kernel Module (CVE-2022-2639).....	42
2.4.3 Notice on CRI-O Container Runtime Engine Arbitrary Code Execution Vulnerability (CVE-2022-0811).....	43
2.4.4 Notice on the Container Escape Vulnerability Caused by the Linux Kernel (CVE-2022-0492).....	44
2.4.5 Linux Kernel Integer Overflow Vulnerability (CVE-2022-0185).....	45
3 Kubernetes Basics.....	47
3.1 Overview.....	47
3.2 Container and Kubernetes.....	48
3.2.1 Containers.....	48
3.2.2 Kubernetes.....	52

3.3 Pod, Label, and Namespace.....	58
3.3.1 Pod: the Smallest Scheduling Unit in Kubernetes.....	59
3.3.2 Liveness Probe.....	63
3.3.3 Label for Managing Pods.....	66
3.3.4 Namespaces: Grouping Resources.....	68
3.4 Pod Orchestration and Scheduling.....	70
3.4.1 Deployment.....	70
3.4.2 StatefulSet.....	74
3.4.3 Job and Cron Job.....	79
3.4.4 DaemonSet.....	81
3.4.5 Affinity and Anti-Affinity Scheduling.....	83
3.5 Configuration Management.....	90
3.5.1 ConfigMap.....	90
3.5.2 Secret.....	92
3.6 Kubernetes Networking.....	93
3.6.1 Container Networking.....	94
3.6.2 Services.....	95
3.6.3 Ingresses.....	104
3.6.4 Readiness Probe.....	106
3.6.5 NetworkPolicy.....	110
3.7 Persistent Storage.....	111
3.7.1 Volume.....	112
3.7.2 PersistentVolumes, PersistentVolumeClaims, and StorageClasses.....	114
3.8 Authentication and Authorization.....	118
3.8.1 ServiceAccounts.....	118
3.8.2 RBAC.....	122
3.9 Auto Scaling.....	126
4 Getting Started.....	130
4.1 Introduction.....	130
4.2 Preparations.....	131
4.3 Creating a Kubernetes Cluster.....	134
4.4 Creating a Deployment (Nginx) from an Image.....	136
4.5 Deploying WordPress and MySQL That Depend on Each Other.....	138
4.5.1 Overview.....	138
4.5.2 Step 1: Create a MySQL Workload.....	139
4.5.3 Step 2: Create a WordPress Workload.....	140
5 High-Risk Operations and Solutions.....	145
6 Clusters.....	152
6.1 Cluster Overview.....	152
6.1.1 Basic Cluster Information.....	152
6.1.2 Kubernetes Release Notes.....	153

6.1.2.1 Kubernetes 1.27 Release Notes.....	153
6.1.2.2 Kubernetes 1.25 Release Notes.....	159
6.1.2.3 Kubernetes 1.23 Release Notes.....	163
6.1.2.4 Kubernetes 1.21 Release Notes.....	164
6.1.2.5 Kubernetes 1.19 Release Notes.....	166
6.1.2.6 Kubernetes 1.17 (EOM) Release Notes.....	168
6.1.3 Release Notes for CCE Cluster Versions.....	169
6.2 Creating a Cluster.....	174
6.2.1 CCE Turbo Clusters and CCE Clusters.....	174
6.2.2 Creating a Cluster.....	175
6.2.3 Comparing iptables and IPVS.....	178
6.3 Connecting to a Cluster.....	179
6.3.1 Connecting to a Cluster Using kubectl.....	179
6.3.2 Connecting to a Cluster Using an X.509 Certificate.....	182
6.3.3 Accessing a Cluster Using a Custom Domain Name.....	183
6.4 Upgrading a Cluster.....	184
6.4.1 Upgrade Overview.....	184
6.4.2 Before You Start.....	187
6.4.3 Performing In-place Upgrade.....	204
6.4.4 Performing Post-Upgrade Verification.....	206
6.4.4.1 Service Verification.....	206
6.4.4.2 Pod Check.....	207
6.4.4.3 Node and Container Network Check.....	207
6.4.4.4 Node Label and Taint Check.....	209
6.4.4.5 New Node Check.....	210
6.4.4.6 New Pod Check.....	210
6.4.4.7 Node Skipping Check.....	211
6.4.5 Migrating Services Across Clusters of Different Versions.....	212
6.4.6 Troubleshooting for Pre-upgrade Check Exceptions.....	213
6.4.6.1 Pre-upgrade Check.....	213
6.4.6.2 Node Restrictions.....	218
6.4.6.3 Upgrade Management.....	219
6.4.6.4 Add-ons.....	220
6.4.6.5 Helm Charts.....	220
6.4.6.6 SSH Connectivity of Master Nodes.....	221
6.4.6.7 Node Pools.....	221
6.4.6.8 Security Groups.....	221
6.4.6.9 Arm Node Restrictions.....	222
6.4.6.10 To-Be-Migrated Nodes.....	222
6.4.6.11 Discarded Kubernetes Resources.....	222
6.4.6.12 Compatibility Risks.....	223
6.4.6.13 Node CCE Agent Versions.....	228

6.4.6.14 Node CPU Usage.....	229
6.4.6.15 CRDs.....	229
6.4.6.16 Node Disks.....	230
6.4.6.17 Node DNS.....	230
6.4.6.18 Node Key Directory File Permissions.....	231
6.4.6.19 Kubelet.....	231
6.4.6.20 Node Memory.....	231
6.4.6.21 Node Clock Synchronization Server.....	232
6.4.6.22 Node OS.....	233
6.4.6.23 Node CPUs.....	233
6.4.6.24 Node Python Commands.....	233
6.4.6.25 ASM Version.....	233
6.4.6.26 Node Readiness.....	234
6.4.6.27 Node journald.....	234
6.4.6.28 containerd.sock.....	235
6.4.6.29 Internal Errors.....	235
6.4.6.30 Node Mount Points.....	235
6.4.6.31 Kubernetes Node Taints.....	236
6.4.6.32 everest Restrictions.....	237
6.4.6.33 cce-hpa-controller Restrictions.....	237
6.4.6.34 Enhanced CPU Policies.....	237
6.4.6.35 Health of Worker Node Components.....	238
6.4.6.36 Health of Master Node Components.....	238
6.4.6.37 Memory Resource Limit of Kubernetes Components.....	238
6.4.6.38 Discarded Kubernetes APIs.....	239
6.4.6.39 IPv6 Capabilities of a CCE Turbo Cluster.....	239
6.4.6.40 Node NetworkManager.....	240
6.4.6.41 Node ID File.....	240
6.4.6.42 Node Configuration Consistency.....	240
6.4.6.43 Node Configuration File.....	242
6.4.6.44 CoreDNS Configuration Consistency.....	243
6.4.6.45 sudo Commands of a Node.....	244
6.4.6.46 Key Commands of Nodes.....	245
6.4.6.47 Mounting of a Sock File on a Node.....	245
6.4.6.48 HTTPS Load Balancer Certificate Consistency.....	246
6.4.6.49 Node Mounting.....	247
6.4.6.50 Login Permissions of User paas on a Node.....	248
6.4.6.51 Private IPv4 Addresses of Load Balancers.....	249
6.4.6.52 Historical Upgrade Records.....	249
6.4.6.53 CIDR Block of the Cluster Management Plane.....	249
6.4.6.54 GPU Add-on.....	250
6.4.6.55 Nodes' System Parameter Settings.....	250

6.4.6.56 Residual Package Versions.....	251
6.4.6.57 Node Commands.....	251
6.4.6.58 Node Swap.....	251
6.4.6.59 nginx-ingress Upgrade.....	252
6.5 Managing a Cluster.....	253
6.5.1 Cluster Configuration Management.....	253
6.5.2 Cluster Overload Control.....	259
6.5.3 Changing Cluster Scale.....	259
6.5.4 Deleting a Cluster.....	260
6.5.5 Hibernating and Waking Up a Cluster.....	261
7 Nodes.....	263
7.1 Node Overview.....	263
7.2 Container Engine.....	265
7.3 Creating a Node.....	268
7.4 Adding Nodes for Management.....	274
7.5 Logging In to a Node.....	278
7.6 Management Nodes.....	279
7.6.1 Managing Node Labels.....	279
7.6.2 Managing Node Taints.....	281
7.6.3 Resetting a Node.....	284
7.6.4 Removing a Node.....	287
7.6.5 Synchronizing Data with Cloud Servers.....	289
7.6.6 Deleting a Node.....	289
7.6.7 Stopping a Node.....	290
7.6.8 Performing Rolling Upgrade for Nodes.....	291
7.7 Node O&M.....	294
7.7.1 Node Resource Reservation Policy.....	294
7.7.2 Data Disk Space Allocation.....	297
7.7.3 Maximum Number of Pods That Can Be Created on a Node.....	301
7.7.4 Migrating Nodes from Docker to containerd.....	303
7.7.5 Node Fault Detection Policy.....	304
8 Node Pools.....	315
8.1 Node Pool Overview.....	315
8.2 Creating a Node Pool.....	318
8.3 Managing a Node Pool.....	326
8.3.1 Updating a Node Pool.....	326
8.3.2 Configuring a Node Pool.....	328
8.3.3 Copying a Node Pool.....	341
8.3.4 Synchronizing Node Pools.....	341
8.3.5 Upgrading an OS.....	342
8.3.6 Migrating a Node.....	344
8.3.7 Deleting a Node Pool.....	344

9 Workloads	346
9.1 Overview	346
9.2 Creating a Workload	350
9.2.1 Creating a Deployment	350
9.2.2 Creating a StatefulSet	357
9.2.3 Creating a DaemonSet	363
9.2.4 Creating a Job	368
9.2.5 Creating a Cron Job	374
9.3 Configuring a Container	380
9.3.1 Configuring Time Zone Synchronization	380
9.3.2 Configuring an Image Pull Policy	381
9.3.3 Using Third-Party Images	382
9.3.4 Setting Container Specifications	383
9.3.5 Setting Container Lifecycle Parameters	386
9.3.6 Setting Health Check for a Container	389
9.3.7 Setting an Environment Variable	392
9.3.8 Configuring the Workload Upgrade Policy	395
9.3.9 Scheduling Policy (Affinity/Anti-affinity)	397
9.3.10 Taints and Tolerations	407
9.3.11 Labels and Annotations	409
9.4 Accessing a Container	411
9.5 Managing Workloads and Jobs	412
10 Scheduling	417
10.1 Overview	417
10.2 CPU Scheduling	418
10.2.1 CPU Policy	418
10.3 GPU Scheduling	420
10.3.1 Default GPU Scheduling in Kubernetes	420
10.4 Volcano Scheduling	422
10.4.1 NUMA Affinity Scheduling	422
11 Network	428
11.1 Overview	428
11.2 Container Network Models	431
11.2.1 Overview	431
11.2.2 Container Tunnel Network	434
11.2.3 VPC Network	438
11.2.4 Cloud Native 2.0 Network	442
11.3 Service	445
11.3.1 Overview	445
11.3.2 ClusterIP	454
11.3.3 NodePort	457
11.3.4 LoadBalancer	461

11.3.4.1 Creating a LoadBalancer Service.....	461
11.3.4.2 Using Annotations to Configure Load Balancing.....	478
11.3.4.3 Service Using HTTP.....	491
11.3.4.4 Configuring Health Check for Multiple Ports.....	493
11.3.4.5 Configuring Timeout for a LoadBalancer Service.....	495
11.3.4.6 Enabling Passthrough Networking for LoadBalancer Services.....	496
11.3.4.7 Enabling ICMP Security Group Rules.....	499
11.3.5 Headless Service.....	499
11.4 Ingresses.....	500
11.4.1 Overview.....	500
11.4.2 ELB Ingresses.....	505
11.4.2.1 Creating an ELB Ingress on the Console.....	505
11.4.2.2 Using kubectl to Create an ELB Ingress.....	510
11.4.2.3 Configuring ELB Ingresses Using Annotations.....	521
11.4.2.4 Configuring HTTPS Certificates for ELB Ingresses.....	529
11.4.2.5 Configuring the Server Name Indication (SNI) for ELB Ingresses.....	534
11.4.2.6 ELB Ingresses Routing to Multiple Services.....	536
11.4.2.7 ELB Ingresses Using HTTP/2.....	536
11.4.2.8 Interconnecting ELB Ingresses with HTTPS Backend Services.....	538
11.4.2.9 Configuring Timeout for an ELB Ingress.....	539
11.4.3 Nginx Ingresses.....	541
11.4.3.1 Creating Nginx Ingresses on the Console.....	541
11.4.3.2 Using kubectl to Create an Nginx Ingress.....	543
11.4.3.3 Configuring HTTPS Certificates for Nginx Ingresses.....	548
11.4.3.4 Configuring URL Rewriting Rules for Nginx Ingresses.....	550
11.4.3.5 Interconnecting Nginx Ingresses with HTTPS Backend Services.....	553
11.4.3.6 Nginx Ingresses Using Consistent Hashing for Load Balancing.....	554
11.4.3.7 Configuring Nginx Ingresses Using Annotations.....	555
11.5 DNS.....	558
11.5.1 Overview.....	558
11.5.2 DNS Configuration.....	561
11.5.3 Using CoreDNS for Custom Domain Name Resolution.....	568
11.6 Container Network Settings.....	572
11.6.1 Host Network.....	572
11.6.2 Configuring QoS Rate Limiting for Inter-Pod Access.....	574
11.6.3 Container Tunnel Network Settings.....	576
11.6.3.1 Network Policies.....	576
11.6.4 Cloud Native Network 2.0 Settings.....	580
11.6.4.1 Security Group Policies.....	581
11.6.4.2 NetworkAttachmentDefinition.....	583
11.7 Cluster Network Settings.....	587
11.7.1 Switching a Node Subnet.....	587

11.7.2 Adding a Container CIDR Block for a Cluster.....	588
11.8 Configuring Intra-VPC Access.....	588
11.9 Accessing Public Networks from a Container.....	591
12 Storage.....	593
12.1 Overview.....	593
12.2 Storage Basics.....	597
12.3 Elastic Volume Service (EVS).....	602
12.3.1 Overview.....	602
12.3.2 Using an Existing EVS Disk Through a Static PV.....	603
12.3.3 Using an EVS Disk Through a Dynamic PV.....	613
12.3.4 Dynamically Mounting an EVS Disk to a StatefulSet.....	620
12.3.5 Snapshots and Backups.....	626
12.4 Scalable File Service (SFS).....	629
12.4.1 Overview.....	629
12.4.2 Using an Existing SFS File System Through a Static PV.....	629
12.4.3 Using an SFS File System Through a Dynamic PV.....	635
12.4.4 Configuring SFS Volume Mount Options.....	639
12.5 SFS Turbo File Systems.....	642
12.5.1 Overview.....	642
12.5.2 Using an Existing SFS Turbo File System Through a Static PV.....	643
12.5.3 Configuring SFS Turbo Mount Options.....	652
12.5.4 Dynamically Creating and Mounting Subdirectories of an SFS Turbo File System.....	654
12.6 Object Storage Service (OBS).....	658
12.6.1 Overview.....	658
12.6.2 Using an Existing OBS Bucket Through a Static PV.....	659
12.6.3 Using an OBS Bucket Through a Dynamic PV.....	671
12.6.4 Configuring OBS Mount Options.....	679
12.6.5 Using a Custom Access Key (AK/SK) to Mount an OBS Volume.....	682
12.7 Local Persistent Volumes (Local PVs).....	687
12.7.1 Overview.....	687
12.7.2 Importing a PV to a Storage Pool.....	688
12.7.3 Using a Local PV Through a Dynamic PV.....	689
12.7.4 Dynamically Mounting a Local PV to a StatefulSet.....	695
12.8 Ephemeral Volumes (emptyDir).....	699
12.8.1 Overview.....	699
12.8.2 Importing an EV to a Storage Pool.....	700
12.8.3 Using a Local EV.....	701
12.8.4 Using a Temporary Path.....	703
12.9 hostPath.....	705
12.10 StorageClass.....	708
13 Observability.....	715
13.1 Logging.....	715

13.1.1 Overview.....	715
13.1.2 Using ICAgent to Collect Container Logs.....	715
13.2 Monitoring.....	721
13.2.1 Monitoring Overview.....	721
13.2.2 Monitoring Custom Metrics on AOM.....	726
13.3 CTS Logs.....	730
13.3.1 CCE Operations Supported by CTS.....	731
13.3.2 Querying Real-Time Traces.....	735
14 Namespaces.....	737
14.1 Creating a Namespace.....	737
14.2 Managing Namespaces.....	739
14.3 Setting a Resource Quota.....	741
15 ConfigMaps and Secrets.....	744
15.1 Creating a ConfigMap.....	744
15.2 Using a ConfigMap.....	746
15.3 Creating a Secret.....	753
15.4 Using a Secret.....	757
15.5 Cluster Secrets.....	763
16 Auto Scaling.....	765
16.1 Overview.....	765
16.2 Scaling a Workload.....	766
16.2.1 Workload Scaling Mechanisms.....	766
16.2.2 HPA.....	768
16.2.3 Managing Workload Scaling Policies.....	770
16.3 Scaling a Node.....	772
16.3.1 Node Scaling Mechanisms.....	772
16.3.2 Creating a Node Scaling Policy.....	778
16.3.3 Managing Node Scaling Policies.....	782
16.4 Using HPA and CA for Auto Scaling of Workloads and Nodes.....	784
17 Add-ons.....	792
17.1 Overview.....	792
17.2 CoreDNS.....	795
17.3 CCE Container Storage (Everest).....	802
17.4 npd.....	805
17.5 CCE Cluster Autoscaler.....	817
17.6 NGNIX Ingress Controller.....	823
17.7 Kubernetes Metrics Server.....	826
17.8 gpu-device-plugin.....	828
17.9 Volcano Scheduler.....	832
17.10 CCE Container Storage (FlexVolume).....	852
18 Helm Chart.....	854

18.1 Overview.....	854
18.2 Deploying an Application from a Chart.....	855
18.3 Differences Between Helm v2 and Helm v3 and Adaptation Solutions.....	859
18.4 Deploying an Application Through the Helm v2 Client.....	860
18.5 Deploying an Application Through the Helm v3 Client.....	863
18.6 Converting a Release from Helm v2 to v3.....	865
19 Permissions.....	868
19.1 Permissions Overview.....	868
19.2 Cluster Permissions (IAM-based).....	870
19.3 Namespace Permissions (Kubernetes RBAC-based).....	875
19.4 Example: Designing and Configuring Permissions for Users in a Department.....	882
19.5 Permission Dependency of the CCE Console.....	885
19.6 Pod Security.....	888
19.6.1 Configuring a Pod Security Policy.....	888
19.6.2 Configuring Pod Security Admission.....	892
19.7 Service Account Token Security Improvement.....	895
20 FAQs.....	897
20.1 Common Questions.....	897
20.2 Billing.....	898
20.2.1 How Is CCE Billed/Charged?.....	898
20.2.2 Will I Be Notified When My Balance Is Insufficient?.....	899
20.2.3 Will I Be Notified When My Account Balance Changes?.....	899
20.3 Cluster.....	899
20.3.1 Cluster Creation.....	899
20.3.1.1 Why Can't I Create a CCE Cluster?.....	899
20.3.1.2 Is Management Scale of a Cluster Related to the Number of Master Nodes?.....	900
20.3.1.3 Which Resource Quotas Should I Pay Attention To When Using CCE?.....	900
20.3.2 Cluster Running.....	900
20.3.2.1 How Do I Rectify the Fault When the Cluster Status Is Unavailable?.....	900
20.3.2.2 How Do I Retrieve Data After a Cluster Is Deleted?.....	901
20.3.3 Cluster Deletion.....	902
20.3.3.1 Failed to Delete a Cluster: Residual ENIs.....	902
20.3.3.2 How Do I Clear Residual Resources After a Non-Running Cluster Is Deleted?.....	902
20.3.4 Cluster Upgrade.....	904
20.3.4.1 What Do I Do If a Cluster Add-On Fails to be Upgraded During the CCE Cluster Upgrade?.....	904
20.4 Node.....	904
20.4.1 Node Creation.....	904
20.4.1.1 How Do I Troubleshoot Problems Occurred When Adding Nodes to a CCE Cluster?.....	905
20.4.2 Node Running.....	906
20.4.2.1 What Should I Do If a Cluster Is Available But Some Nodes Are Unavailable?.....	907
20.4.2.2 How Do I Log In to a Node Using a Password and Reset the Password?.....	912
20.4.2.3 How Do I Collect Logs of Nodes in a CCE Cluster?.....	912

20.4.2.4 What Should I Do If the vdb Disk of a Node Is Damaged and the Node Cannot Be Recovered After Reset?.....	914
20.4.2.5 What Should I Do If I/O Suspension Occasionally Occurs When SCSI EVS Disks Are Used?.....	915
20.4.2.6 How Do I Fix an Abnormal Container or Node Due to No Thin Pool Disk Space?.....	916
20.4.2.7 How Do I Rectify Failures When the NVIDIA Driver Is Used to Start Containers on GPU Nodes?.....	918
20.4.3 Specification Change.....	919
20.4.3.1 How Do I Change the Node Specifications in a CCE Cluster?.....	919
20.4.3.2 What Should I Do If I Fail to Restart or Create Workloads on a Node After Modifying the Node Specifications?.....	920
20.5 Node Pool.....	920
20.5.1 What Should I Do If No Node Creation Record Is Displayed When the Node Pool Is Being Expanding?.....	920
20.6 Workload.....	921
20.6.1 Workload Abnormalities.....	921
20.6.1.1 How Do I Use Events to Fix Abnormal Workloads?.....	921
20.6.1.2 What Should I Do If Pod Scheduling Fails?.....	922
20.6.1.3 What Should I Do If a Pod Fails to Pull the Image?.....	929
20.6.1.4 What Should I Do If Container Startup Fails?.....	936
20.6.1.5 What Should I Do If a Pod Fails to Be Evicted?.....	942
20.6.1.6 What Should I Do If a Storage Volume Cannot Be Mounted or the Mounting Times Out?.....	946
20.6.1.7 What Should I Do If a Workload Remains in the Creating State?.....	947
20.6.1.8 What Should I Do If Pods in the Terminating State Cannot Be Deleted?.....	949
20.6.1.9 What Should I Do If a Workload Is Stopped Caused by Pod Deletion?.....	949
20.6.1.10 What Should I Do If an Error Occurs When Deploying a Service on the GPU Node?.....	950
20.6.1.11 What Should I Do If Sandbox-Related Errors Are Reported When the Pod Remains in the Creating State?.....	951
20.6.2 Container Configuration.....	952
20.6.2.1 When Is Pre-stop Processing Used?.....	952
20.6.2.2 How Do I Set an FQDN for Accessing a Specified Container in the Same Namespace?.....	952
20.6.2.3 What Should I Do If Health Check Probes Occasionally Fail?.....	953
20.6.2.4 How Do I Set the umask Value for a Container?.....	953
20.6.2.5 What Can I Do If an Error Is Reported When a Deployed Container Is Started After the JVM Startup Heap Memory Parameter Is Specified for ENTRYPOINT in Dockerfile?.....	954
20.6.2.6 What Is the Retry Mechanism When CCE Fails to Start a Pod?.....	954
20.6.3 Scheduling Policies.....	955
20.6.3.1 How Do I Evenly Distribute Multiple Pods to Each Node?.....	955
20.6.3.2 How Do I Prevent a Container on a Node from Being Evicted?.....	956
20.6.3.3 Why Are Pods Not Evenly Distributed to Nodes?.....	956
20.6.3.4 How Do I Evict All Pods on a Node?.....	956
20.6.4 Others.....	958
20.6.4.1 What Should I Do If a Scheduled Task Cannot Be Restarted After Being Stopped for a Period of Time?.....	958
20.6.4.2 What Is a Headless Service When I Create a StatefulSet?.....	959

20.6.4.3 What Should I Do If Error Message "Auth is empty" Is Displayed When a Private Image Is Pulled?.....	960
20.6.4.4 Why Cannot a Pod Be Scheduled to a Node?.....	960
20.6.4.5 What Is the Image Pull Policy for Containers in a CCE Cluster?.....	960
20.6.4.6 What Can I Do If a Layer Is Missing During Image Pull?.....	961
20.7 Networking.....	961
20.7.1 Network Planning.....	961
20.7.1.1 What Is the Relationship Between Clusters, VPCs, and Subnets?.....	961
20.7.1.2 Configuring Cluster Security Group Rules.....	962
20.7.2 Network Fault.....	972
20.7.2.1 How Do I Locate a Workload Networking Fault?.....	972
20.7.2.2 Why Does the Browser Return Error Code 404 When I Access a Deployed Application?.....	974
20.7.2.3 What Should I Do If a Container Fails to Access the Internet?.....	975
20.7.2.4 What Should I Do If a Node Fails to Connect to the Internet (Public Network)?.....	976
20.7.2.5 What Should I Do If an Nginx Ingress Access in the Cluster Is Abnormal After the Add-on Is Upgraded?.....	976
20.8 Storage.....	978
20.8.1 What Are the Differences Among CCE Storage Classes in Terms of Persistent Storage and Multi-node Mounting?.....	978
20.8.2 Can I Add a Node Without a Data Disk?.....	979
20.8.3 What Should I Do If the Host Cannot Be Found When Files Need to Be Uploaded to OBS During the Access to the CCE Service from a Public Network?.....	979
20.8.4 How Can I Achieve Compatibility Between ExtendPathMode and Kubernetes client-go?.....	980
20.8.5 Can CCE PVCs Detect Underlying Storage Faults?.....	983
20.9 Namespace.....	983
20.9.1 Why Cannot I Delete a Namespace Due to an APIService Object Access Failure?.....	983
20.10 Chart and Add-on.....	984
20.10.1 Why Does Add-on Installation Fail and Prompt "The release name is already exist"?.....	984
20.11 API & kubectl FAQs.....	986
20.11.1 How Can I Access a Cluster API Server?.....	986
20.11.2 Can the Resources Created Using APIs or kubectl Be Displayed on the CCE Console?.....	986
20.11.3 How Do I Download kubeconfig for Connecting to a Cluster Using kubectl?.....	986
20.11.4 How Do I Rectify the Error Reported When Running the kubectl top node Command?.....	987
20.11.5 Why Is "Error from server (Forbidden)" Displayed When I Use kubectl?.....	987
20.12 DNS FAQs.....	988
20.12.1 What Should I Do If Domain Name Resolution Fails?.....	988
20.12.2 Why Does a Container in a CCE Cluster Fail to Perform DNS Resolution?.....	990
20.12.3 How Do I Optimize the Configuration If the External Domain Name Resolution Is Slow or Times Out?.....	991
20.12.4 How Do I Configure a DNS Policy for a Container?.....	992
20.13 Image Repository FAQs.....	993
20.13.1 How Do I Upload My Images to CCE?.....	993
20.14 Permissions.....	993
20.14.1 Can I Configure Only Namespace Permissions Without Cluster Management Permissions?.....	993

20.14.2 Can I Use CCE APIs If the Cluster Management Permissions Are Not Configured?.....	994
20.14.3 Can I Use kubectl If the Cluster Management Permissions Are Not Configured?.....	994
20.15 Reference.....	994
20.15.1 How Do I Expand the Storage Capacity of a Container?.....	995
20.15.2 How Can Container IP Addresses Survive a Container Restart?.....	996
21 Best Practices.....	997
21.1 Checklist for Deploying Containerized Applications in the Cloud.....	997
21.2 Containerization.....	1001
21.2.1 Containerizing an Enterprise Application (ERP).....	1001
21.2.1.1 Solution Overview.....	1001
21.2.1.2 Procedure.....	1005
21.2.1.2.1 Containerizing an Entire Application.....	1005
21.2.1.2.2 Containerization Process.....	1007
21.2.1.2.3 Analyzing the Application.....	1008
21.2.1.2.4 Preparing the Application Runtime.....	1010
21.2.1.2.5 Compiling a Startup Script.....	1012
21.2.1.2.6 Compiling the Dockerfile.....	1013
21.2.1.2.7 Building and Uploading an Image.....	1014
21.2.1.2.8 Creating a Container Workload.....	1015
21.3 Disaster Recovery.....	1019
21.3.1 Implementing High Availability for Applications in CCE.....	1019
21.4 Security.....	1021
21.4.1 Cluster Security.....	1021
21.4.2 Node Security.....	1025
21.4.3 Container Security.....	1026
21.4.4 Secret Security.....	1028
21.5 Auto Scaling.....	1030
21.5.1 Using HPA and CA for Auto Scaling of Workloads and Nodes.....	1030
21.6 Monitoring.....	1037
21.6.1 Using Prometheus for Multi-cluster Monitoring.....	1038
21.7 Cluster.....	1042
21.7.1 Connecting to Multiple Clusters Using kubectl.....	1042
21.7.2 Selecting a Data Disk for the Node.....	1047
21.8 Networking.....	1052
21.8.1 Planning CIDR Blocks for a Cluster.....	1052
21.8.2 Selecting a Network Model.....	1059
21.8.3 Implementing Sticky Session Through Load Balancing.....	1064
21.8.4 Obtaining the Client Source IP Address for a Container.....	1067
21.9 Storage.....	1069
21.9.1 Expanding the Storage Space.....	1069
21.9.2 Mounting an Object Storage Bucket of a Third-Party Tenant.....	1075
21.9.3 Dynamically Creating and Mounting Subdirectories of an SFS Turbo File System.....	1079

21.9.4 How Do I Change the Storage Class Used by a Cluster of v1.15 from FlexVolume to CSI Everest?	1083
21.9.5 Custom Storage Classes.....	1095
21.9.6 Enabling Automatic Topology for EVS Disks When Nodes Are Deployed in Different AZs (csi-disk-topology).....	1101
21.10 Container.....	1106
21.10.1 Properly Allocating Container Computing Resources.....	1107
21.10.2 Modifying Kernel Parameters Using a Privileged Container.....	1108
21.10.3 Using Init Containers to Initialize an Application.....	1110
21.10.4 Using hostAliases to Configure /etc/hosts in a Pod.....	1112
21.10.5 Configuring Core Dumps.....	1114
21.11 Permission.....	1115
21.11.1 Configuring kubeconfig for Fine-Grained Management on Cluster Resources.....	1116
21.12 Release.....	1119
21.12.1 Overview.....	1119
21.12.2 Using Services to Implement Simple Grayscale Release and Blue-Green Deployment.....	1121
22 Migrating Data from CCE 1.0 to CCE 2.0.....	1128
22.1 Differences Between CCE 1.0 and CCE 2.0.....	1128
22.2 Migrating Images.....	1129
22.3 Migrating Clusters.....	1130
22.4 Migrating Applications.....	1141
22.4.1 Applications Created Through APIs or kubectl.....	1142
22.4.2 Applications Created Through Component Templates.....	1142
22.4.3 Applications Created Through App Designer.....	1149

1 Service Overview

1.1 What Is Cloud Container Engine?

Cloud Container Engine (CCE) is a scalable, enterprise-class hosted Kubernetes service. With CCE, you can easily deploy, manage, and scale containerized applications in the cloud.

Why CCE?

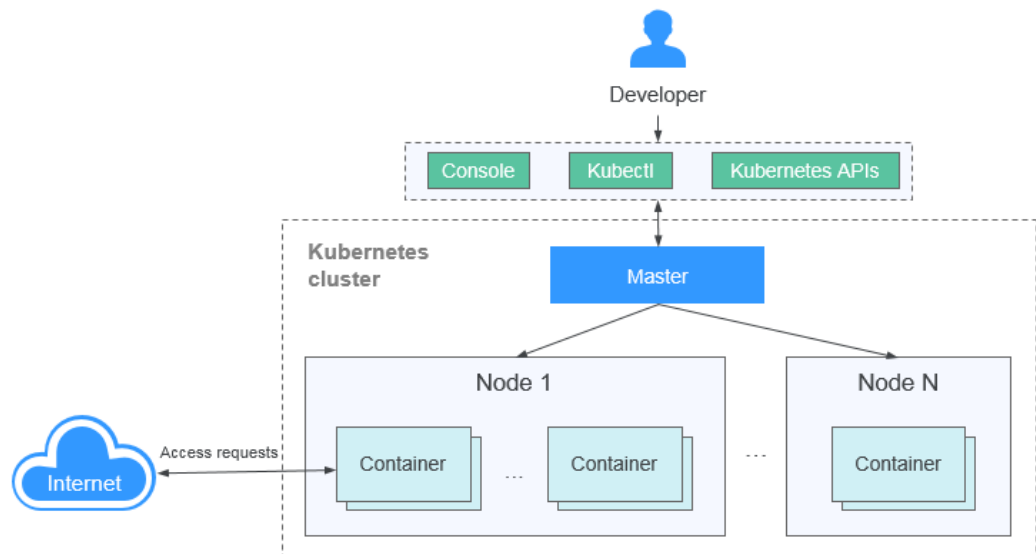
CCE is a one-stop platform integrating compute, networking, storage, and many other services. Supporting multi-AZ and multi-region disaster recovery, CCE ensures high availability of [Kubernetes](#) clusters.

For more information, see [Product Advantages](#) and [Application Scenarios](#).

Accessing CCE

You can use CCE via the CCE console, kubectl, or Kubernetes APIs. [Figure 1-1](#) shows the process.

Figure 1-1 Accessing CCE



1.2 Product Advantages

Why CCE?

CCE is a container service built on Docker and Kubernetes. A wealth of features enables you to run container clusters at scale. CCE eases containerization thanks to its reliability, performance, and open source engagement.

Easy to Use

- Creating a Kubernetes cluster is as easy as a few clicks on the web console. You can deploy and manage VMs and BMSs together.
- CCE automates deployment and O&M of containerized applications throughout their lifecycle.
- You can resize clusters and workloads by setting auto scaling policies. In-the-moment load spikes are no longer headaches.
- The console walks you through the steps to upgrade Kubernetes clusters.
- CCE supports turnkey Helm charts.

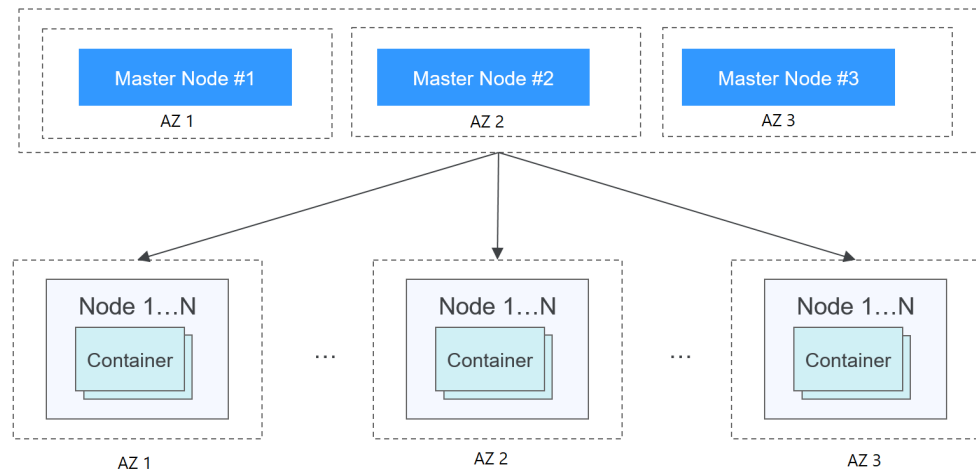
High Performance

- CCE runs on mature IaaS services and heterogeneous compute resources. You can launch containers at scale.
- AI computing is 3x to 5x better with NUMA BMSs and high-speed InfiniBand network cards.

Highly Available and Secure

- HA: Three master nodes in different AZs for your cluster control plane. Multi-active DR for your nodes and workloads. All these ensure service continuity when one of the nodes is down or an AZ gets hit by natural disasters.

Figure 1-2 High-availability setup of clusters



- **Secure:** Integrating IAM and Kubernetes RBAC, CCE clusters are under your full control. You can set different RBAC permissions for IAM users on the console.

Open and Compatible

- CCE runs on Docker that automates container deployment, discovery, scheduling, and scaling.
- CCE is compatible with native Kubernetes APIs and kubectl. Updates from Kubernetes and Docker communities are regularly incorporated into CCE.

Comparative Analysis of CCE and On-Premises Kubernetes Cluster Management Systems

Table 1-1 CCE clusters versus on-premises Kubernetes clusters

Area of Focus	On-Premises Cluster	CCE
Ease of use	You have to handle all the complexity in deploying and managing Kubernetes clusters. Cluster upgrades are often a heavy burden to O&M personnel.	<p>Easy to manage and use clusters</p> <p>You can create and upgrade a Kubernetes container cluster in a few clicks without setting up Docker or Kubernetes environments. CCE automates deployment and O&M of containerized applications throughout their lifecycle.</p> <p>CCE supports turnkey Helm charts.</p> <p>Using CCE is as simple as choosing a cluster and the workloads that you want to run in the cluster. CCE takes care of cluster management and you focus on app development.</p>

Area of Focus	On-Premises Cluster	CCE
Scalability	You have to assess service loads and cluster health before resizing a cluster.	Managed scaling service CCE auto scales clusters and workloads according to resource metrics and scaling policies.
Reliability	Only one master node is available in a cluster. Once this node is down, the entire cluster is down, as well as all the applications in it.	High availability Enabling HA when creating a cluster will create three master nodes for the control plane. Single points of failure (SPOFs) will not shut down your cluster.
Efficiency	You have to either build an image repository or turn to a third-party one. Images are pulled in serial.	Rapid deployment with images CCE connects to SWR to pull images in parallel. Faster pulls, faster container build.
Cost	Heavy upfront investment in installing, managing, and scaling cluster management infrastructure	Cost effective You only pay for master nodes and the resources used to run and manage applications.

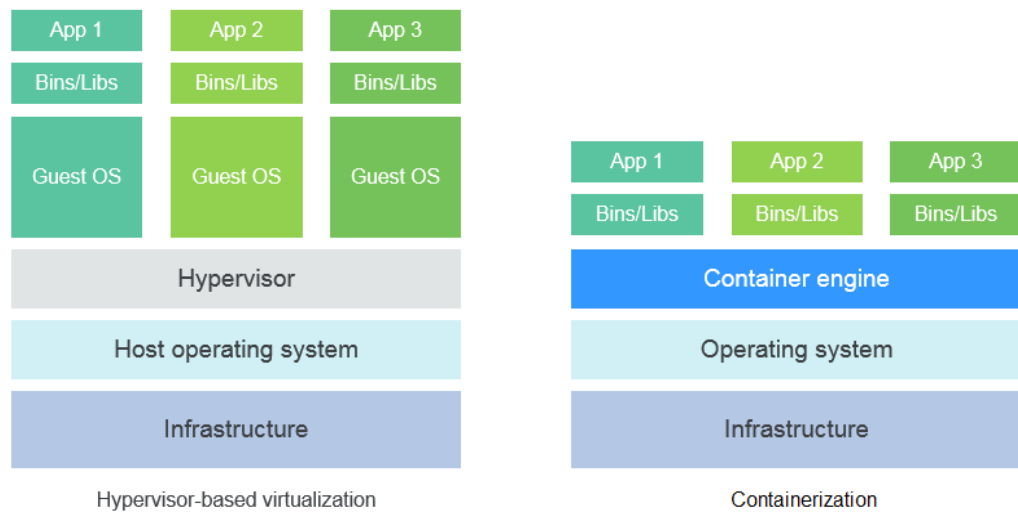
Why Containers?

Docker is written in the Go language designed by Google. It provides operating-system-level virtualization. Linux Control Groups (cgroups), namespaces, and UnionFS (for example, AUFS) isolate each software process. A Docker container packages everything needed to run a software process. Containers are independent from each other and from the host.

Docker has moved forward to enhance container isolation. Containers have their own file systems. They cannot see each other's processes or network interfaces. This simplifies container creation and management.

VMs use a hypervisor to virtualize and allocate hardware resources (such as memory, CPU, network, and disk) of a host machine. A complete operating system runs on a VM. Each VM needs to run its own system processes. On the contrary, a container does not require hardware resource virtualization. It runs an application process directly in the the host machine OS kernel. No resource overheads are incurred by running system processes. Therefore, Docker is lighter and faster than VMs.

Figure 1-3 Comparison between Docker containers and VMs



To sum up, Docker containers have many advantages over VMs.

Resource use

Containers have no overheads for virtualizing hardware and running a complete OS. They are faster than VMs in execution and file storage, while having no memory loss.

Start speed

It takes several minutes to start an application on a VM. Docker containers run on the host kernel without needing an independent OS. Apps in containers can start in seconds or even milliseconds. Development, testing, and deployment can be much faster.

Consistent environment

Different development, testing, and production environments sometimes prevent bug discovery before rollout. A Docker container image includes everything needed to run an application. You can deploy the same copy of configurations in different environments.

Continuous delivery and deployment

"Deploy once, run everywhere" would be great for DevOps personnel.

Docker supports CI/CD by allowing you to customize container images. You compile Dockerfiles to build container images and use CI systems for testing. The Ops team can deploy images into production environments and use CD systems for auto deployment.

The use of Dockerfiles makes the DevOps process visible to everyone in a DevOps team. Developers can better understand both user needs and the O&M headaches faced by the Ops team. The Ops team can also have some knowledge of the must-met conditions to run the application. The knowledge is helpful when the Ops personnel deploy container images in production.

Portability

Docker ensures environmental consistency across development, testing, and production. Portable Docker containers work the same, regardless of their running environments, including physical machines, VMs, or even laptops. Apps are now free to migrate and run anywhere.

Application update

Docker images consist of layers. Each layer is only stored once and different images can contain the exact same layers. When transferring such images, those same layers get transferred only once. This makes distribution efficient. Updating a containerized application is also simple. Either edit the top-most writable layer in the final image or add layers to the base image. Docker joins hands with many open source projects to maintain a variety of high-quality official images. You can directly use them in the production environment or easily build new images based on them.

Table 1-2 Containers versus traditional VMs

Feature	Containers	VMs
Start speed	In seconds	In minutes
Disk capacity	MiB	GiB
Performance	Near-native performance	Weak
Per-machine capacity	Thousands of containers	Tens of VMs

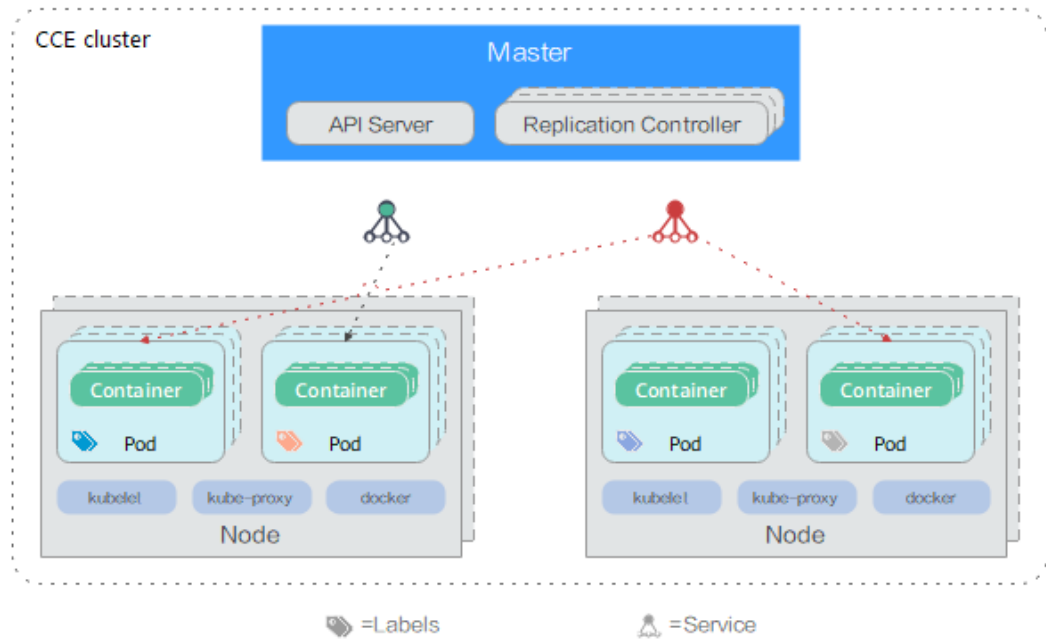
1.3 Application Scenarios

1.3.1 Infrastructure and Containerized Application Management

Application Scenario

In CCE, you can run clusters with x86 and Arm nodes. Create and manage Kubernetes clusters. Deploy containerized applications in them. All done in CCE.

Figure 1-4 CCE cluster



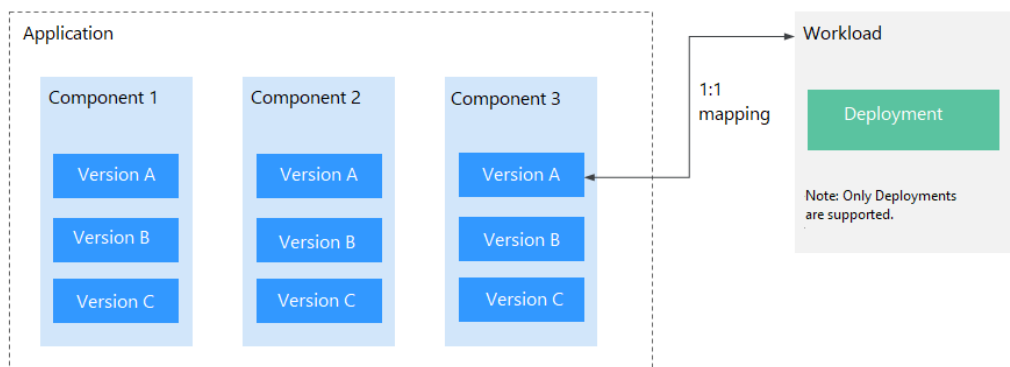
Benefits

Containerization requires less resources to deploy application. Services are not interrupted during upgrades.

Advantages

- Multiple types of workloads
Runs Deployments, StatefulSets, DaemonSets, jobs, and cron jobs to meet different needs.
- Application upgrade
Upgrades your apps in replace or rolling mode (by proportion or by number of pods), or rolls back the upgrades.
- Auto scaling
Auto scales your nodes and workloads according to the policies you set.

Figure 1-5 Workload



1.3.2 Auto Scaling in Seconds

Application Scenarios

- Shopping apps and websites, especially during promotions
- Live streaming, where service loads often fluctuate
- Games, where many players may go online in certain time periods

Benefits

CCE auto adjusts capacity to cope with service surges according to the policies you set. CCE adds or reduces cloud servers and containers to scale your cluster and workloads. Your applications will always have the right resources at the right time.

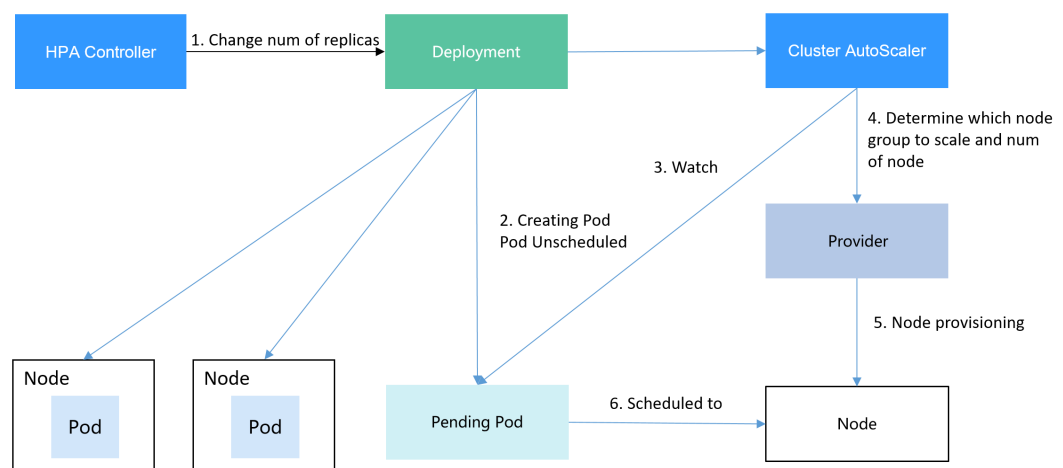
Advantages

- Flexible
Allows diverse types of scaling policies and scales containers within seconds once triggered.
- Highly available
Monitors pod running and replaces unhealthy pods with new ones.
- Lower costs
Bills you only for the scaled cloud servers as you use.

Related Services

HPA (Horizontal Pod Autoscaling) + CA (Cluster AutoScaling)

Figure 1-6 How auto scaling works



1.3.3 DevOps and CI/CD

Application Scenario

You may receive a lot feedback and requirements for your apps or services. You may want to boost user experience with new features. Continuous integration (CI)

and delivery (CD) can help. CI/CD automates builds, tests, and merges, making app delivery faster.

Benefits

CCE works with SWR to support DevOps and CI/CD. A pipeline automates coding, image build, grayscale release, and deployment based on code sources. Existing CI/CD systems can connect to CCE to containerize legacy applications.

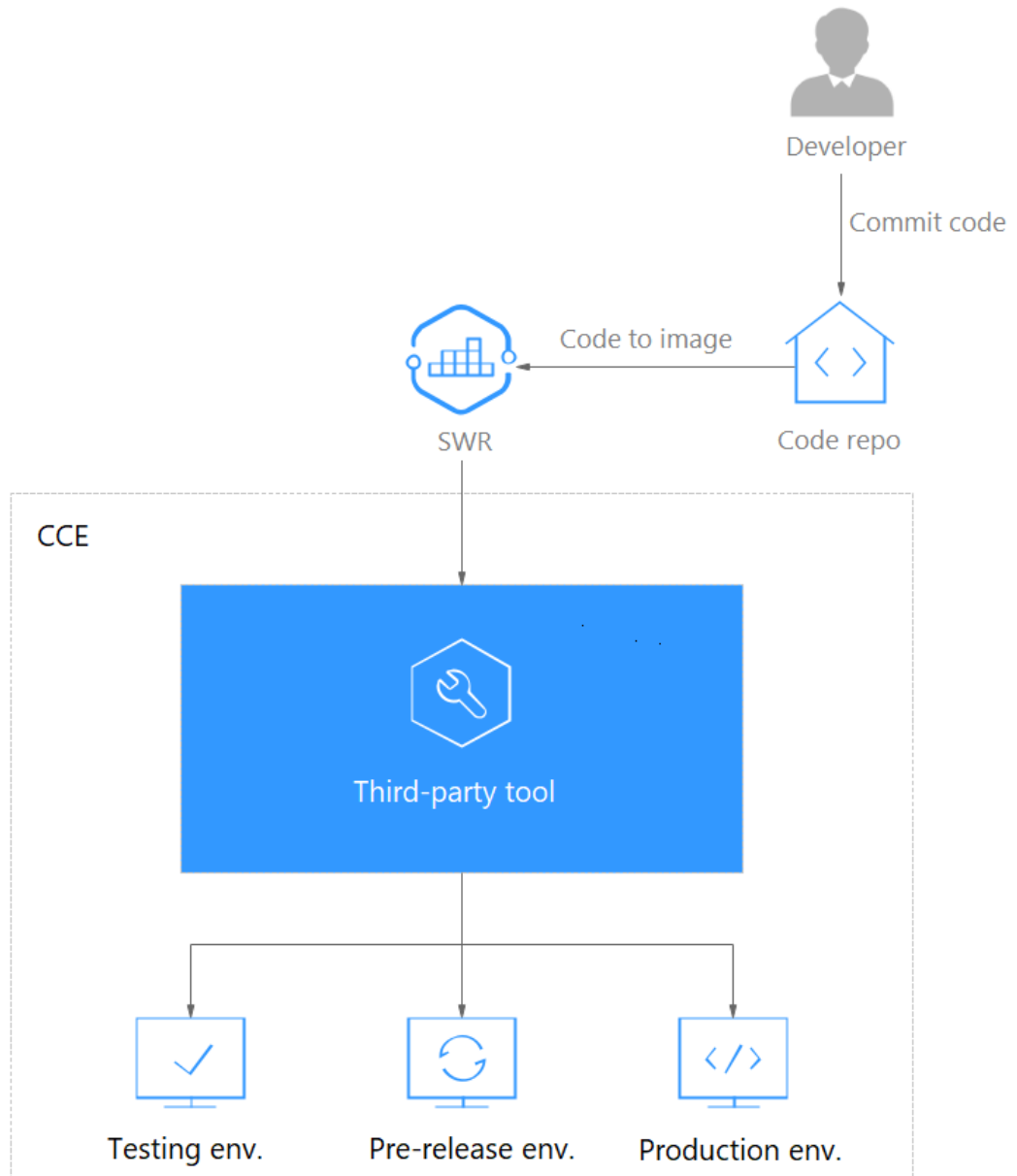
Advantages

- **Efficient process**
Reduces scripting workload by more than 80% through streamlined processes.
- **Flexible integration**
Provides various APIs to integrate with existing CI/CD systems for in-depth customization.
- **High performance**
Enables flexible scheduling with a containerized architecture.

Related Services

Software Repository for Container (SWR), Object Storage Service (OBS), Virtual Private Network (VPN)

Figure 1-7 How DevOps works



1.3.4 Hybrid Cloud Architecture

Application Scenarios

- Multi-cloud deployment and disaster recovery
Running apps in containers on different clouds can ensure high availability. When a cloud is down, other clouds respond and serve.
- Traffic distribution and auto scaling
Large organizations often span cloud facilities in different regions. They need to communicate and auto scale — start small and then scale as system load grows. CCE takes care of these for you, cutting the costs of maintaining facilities.

- Migration to the cloud and database hosting
Industries like finance and security have a top concern on data protection. They want to run critical systems in local IDCs while moving others to the cloud. They also expect one unified dashboard to manage all systems.
- Environment decoupling
To ensure IP security, you can decouple development from production. Set up one on the cloud and the other in the local IDC.

Benefits

Your apps and data can flow free on and off the cloud. Resource scheduling and DR are much easier, thanks to environment-independent containers. CCE provides container services with interconnected networks and unified management.

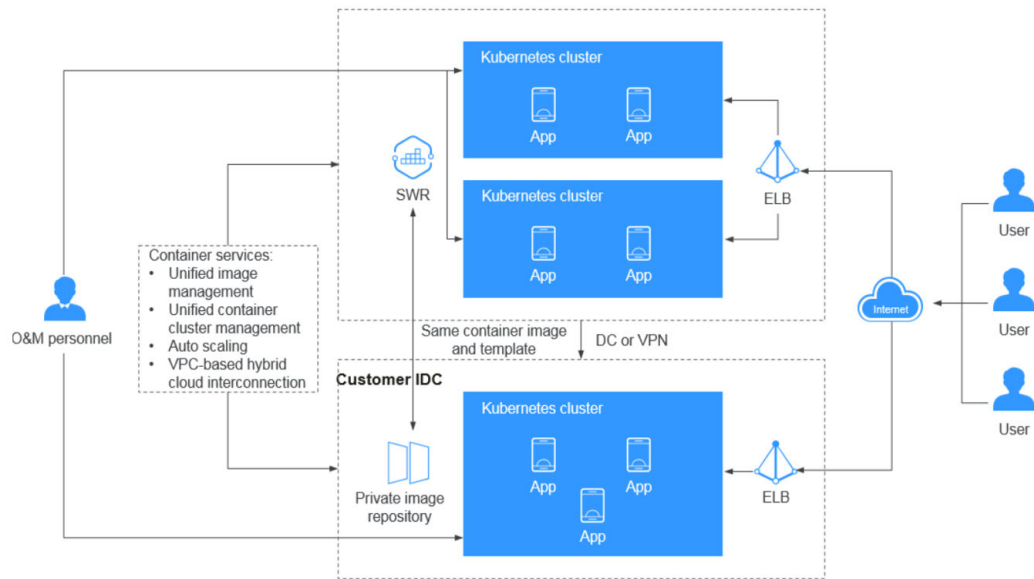
Advantages

- On-cloud DR
Multicloud prevents systems from outages. When a cloud is faulty, CCE auto diverts traffic to other clouds to ensure service continuity.
- Automatic traffic distribution
CCE reduces access latency by processing user requests on the nodes nearest to the users. Overloaded apps in local IDCs can be burst to the cloud backed by auto scaling.
- Decoupling and sharing
CCE decouples data, environments, and compute capacity. Sensitive data vs general data. Development vs production. Compute-intensive services vs general services. Apps running on-premises can burst to the cloud. Your resources on and off the cloud can be better used.
- Lower costs
Cloud resource pools, backed by auto scaling, can respond to load spikes in time. Manual operations are no longer needed and you can save big.

Related Services

Elastic Cloud Server (ECS), Direct Connect (DC), Virtual Private Network (VPN), SoftWare Repository for Container (SWR)

Figure 1-8 How hybrid cloud works



1.4 Constraints

This section describes the notes and constraints on using CCE.

Clusters and Nodes

- After a cluster is created, the following items cannot be changed:
 - Number of master nodes: For example, a non-HA cluster (with one master node) cannot be changed to an HA cluster (with three master nodes).
 - AZ where a master node is deployed
 - Network configuration of the cluster, such as the VPC, subnet, container CIDR block, Service CIDR block, IPv6 settings, and kube-proxy (forwarding) settings.
 - Network model: For example, a container tunnel network cannot be changed to a VPC network.
- CCE underlying resources such as ECS nodes are limited by quota and their inventory. It is possible that only some nodes are created during cluster creation, cluster scaling, or auto scaling.
- ECS node specifications: CPU \geq 2 cores, memory \geq 4 GiB
- To access a CCE cluster through a VPN, ensure that the VPN CIDR block does not conflict with the VPC CIDR block where the cluster resides and the container CIDR block.

Networks

- By default, a NodePort Service is accessed within a VPC. To access a NodePort Service through the Internet, bind an EIP to the node in the cluster beforehand.
- 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.
- Constraints on network policies:
 - 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: Only the following OSs and cluster versions support egress rules.

OS	Cluster Version	Verified Kernel Version
CentOS	v1.23 or later	3.10.0-1062.18.1.el7.x86_64 3.10.0-1127.19.1.el7.x86_64 3.10.0-1160.25.1.el7.x86_64
EulerOS 2.5	v1.23 or later	3.10.0-862.14.1.5.h591.eulerosv2r7.x86_64 3.10.0-862.14.1.5.h687.eulerosv2r7.x86_64
EulerOS 2.9	v1.23 or later	4.18.0-147.5.1.6.h541.eulerosv2r9.x86_64 4.18.0-147.5.1.6.h766.eulerosv2r9.x86_64

- Network isolation is not supported for IPv6 addresses.
- If upgrade to a cluster version that supports egress rules is performed in in-place mode, you cannot use egress rules because the node OS is not upgraded. In this case, reset the node.

Storage Volumes

- Constraints on EVS volumes:
 - 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, create 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 disks attached, 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 disks attached, a new pod cannot be started because EVS disks cannot be attached.
- Constraints on SFS volumes:
 - Multiple PVs can use the same SFS or SFS Turbo file system with the following restrictions:

- If multiple PVCs/PVs use the same underlying SFS or SFS Turbo file system, when you attempt to mount these PVCs/PVs to the same pod, all PVCs cannot be mounted to the pod and the pod startup fails. This is because the **volumeHandle** values of these PVs are the same.
- 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.
- Constraints on OBS volumes:
 - When parallel file systems and object buckets are used, the group and permission of the mount point cannot be modified.
 - CCE allows you to use OBS parallel file systems by calling the OBS SDK or mounting a PVC through the **obsfs** tool provided by OBS. Each time an OBS parallel file system is mounted, an obsfs resident process is generated. The following shows an example.

Figure 1-9 obsfs resident process

```
tree@k8s-master-112b-pfr-58064 -[~] ps -aux | grep obsfs
root      7923  0.0  0.3 33520 4048  ?        Ss   11:09   0:00 /usr/bin/obsfs --pv-e77bf8d-3387-4814-9a23-fbaa55693cf1 --mnt/pas/kubernetes/ubicle/pods/4099580-36a-410-1694-103a188401e/pas/kubernetes_1--csi/pvc-e77bf8d-3387-4814-9a23-fbaa55693cf1/mount --url=https://obs.csi.cce.com --mount.kubernetes --id=443 --endpoint=north-7 --passwd_file=/opt/everest-host-connector/36179375369250038_obsfsycrred/pvc-e77bf8d-3387-4814-9a23-fbaa55693cf1 --a
New other -> memory -> lib_writes -> ext_writes33897 -> ext_backgounds30 -> use_top -> ext_check_certificate -> done
```

Reserve 1 GiB of memory for each obsfs process. For example, for a node with 4 vCPUs and 8 GiB of memory, an obsfs parallel file system should be mounted to **no more than eight** pods.

NOTE

An obsfs resident process runs on a node. If the consumed memory exceeds the upper limit of the node, the node malfunctions. On a node with 4 vCPUs and 8 GiB of memory, if more than 100 pods are mounted to a parallel file system, the node will be unavailable. Control the number of pods mounted to a parallel file system on a single node.

- Constraints on local PVs:
 - 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, removing, resetting, or scaling in a node will cause the PVC/PV data of the local PV associated with the node to be lost, which cannot be restored or 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.

- A local PV cannot be mounted to multiple workloads or jobs at the same time.
- Constraints on local EVs:
 - 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.
- Constraints on snapshots and backups:
 - 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), data encryption, 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 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`.
 - Snapshot data of encrypted disks is stored encrypted, and that of non-encrypted disks is stored non-encrypted.

Add-ons

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 add-on resources on the backend. Otherwise, add-on exceptions or other unexpected problems may occur.

CCE Cluster Resources

There are resource quotas for your CCE clusters in each region.

Item	Constraints on Common Users
Total number of clusters in a region	50
Number of nodes in a cluster (cluster management scale)	A maximum of 50, 200, 1000, or 2000 nodes can be selected.
Maximum number of pods on a node	256

Item	Constraints on Common Users
Maximum number of pods managed by a cluster	100,000 pods

Dependent Underlying Cloud Resources

Category	Item	Constraints on Common Users
Compute	Pods	1000
	Cores	8000
	RAM capacity (MB)	16,384,000
Networking	VPCs per account	5
	Subnets per account	100
	Security groups per account	100
	Security group rules per account	5000
	Routes per route table	100
	Routes per VPC	100
	VPC peering connections per region	50
	Network ACLs per account	200
	Layer 2 connection gateways per account	5
Load balancing	Elastic load balancers	50
	Load balancer listeners	100
	Load balancer certificates	120
	Load balancer forwarding policies	500
	Load balancer backend host group	500
	Load balancer backend server	500

1.5 Permissions

CCE 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) to provide

a variety of authorization methods, including IAM fine-grained/token authorization and cluster-/namespace-scoped authorization.

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**, such as workloads, jobs, and Services, 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. For details, see [Using RBAC Authorization](#).

⚠ CAUTION

- **Cluster-level 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, jobs, and Services).
- After you create a cluster of v1.11.7-r2 or later, CCE automatically assigns the cluster-admin permissions of all namespaces in the cluster to you, which means you have full control on the cluster and all resources in all namespaces.

Cluster-level Permissions (Assigned by Using IAM System Policies)

By default, new IAM users do not have permissions assigned. Add a user to one or more groups, and attach permissions policies or roles to these groups. Users inherit permissions from the groups to which they are added and can perform specified operations on cloud services based on the permissions.

CCE is a project-level service deployed and accessed in specific physical regions. To assign AOM permissions to a user group, specify the scope as region-specific projects and select projects for the permissions to take effect. If **All projects** is selected, the permissions will take effect for the user group in all region-specific projects. When accessing CCE, the users need to switch to a region where they have been authorized to use the CCE service.

You can grant users permissions by using roles and policies.

- **Roles:** A type of coarse-grained authorization mechanism that defines permissions related to user responsibilities. This mechanism provides only a

limited number of service-level roles for authorization. When using roles to assign permissions, assign other roles on which the permissions depend to take effect. However, roles are not an ideal choice for fine-grained authorization and secure access control.

- **Policies:** A type of fine-grained authorization mechanism that defines permissions required to perform operations on specific cloud resources under certain conditions. This mechanism allows for more flexible policy-based authorization, meeting requirements for secure access control. For example, you can assign users only the permissions for managing a certain type of clusters and nodes.

Table 1-3 lists all the system permissions supported by CCE.

Table 1-3 System permissions supported by CCE

Role/ Policy Name	Description	Type	Dependencies
CCE Administrator	Read and write permissions for CCE clusters and all resources (including workloads, nodes, jobs, and Services) in the clusters	Role	Users granted permissions of this policy must also be granted permissions of the following policies: Global service project: OBS Buckets Viewer and OBS Administrator Region-specific projects: Tenant Guest, Server Administrator, ELB Administrator, SFS Administrator, SWR Admin, and APM FullAccess NOTE Users with both CCE Administrator and NAT Gateway Administrator policies can use NAT Gateway functions for clusters.
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	Policy	None.

Role/ Policy Name	Description	Type	Dependencies
CCE ReadOnly Access	Permissions to view CCE cluster resources, excluding the namespace-level permissions of the clusters (with Kubernetes RBAC enabled)	Policy	None.

Table 1-4 Common operations supported by CCE system policies

Operation	CCE ReadOnlyAcce ss	CCE FullAccess	CCE Administrator
Creating a cluster	x	√	√
Deleting a cluster	x	√	√
Updating a cluster, for example, updating cluster node scheduling parameters and providing RBAC support to clusters	x	√	√
Upgrading a cluster	x	√	√
Waking up a cluster	x	√	√
Hibernating a cluster	x	√	√
Listing all clusters	√	√	√
Querying cluster details	√	√	√
Adding a node	x	√	√
Deleting one or more nodes	x	√	√
Updating a cluster node, for example, updating the node name	x	√	√
Querying node details	√	√	√
Listing all nodes	√	√	√
Listing all jobs	√	√	√

Operation	CCE ReadOnlyAccess	CCE FullAccess	CCE Administrator
Deleting one or more cluster jobs	x	√	√
Querying job details	√	√	√
Creating a storage volume	x	√	√
Deleting a storage volume	x	√	√
Performing operations on all Kubernetes resources	√ (Kubernetes RBAC required)	√ (Kubernetes RBAC required)	√
Performing all operations on an Elastic Cloud Server (ECS)	x	√	√
Performing all operations on Elastic Volume Service (EVS) disks EVS disks can be attached to cloud servers and scaled to a higher capacity whenever needed.	x	√	√
Performing all operations on VPC A cluster must run in a VPC. When creating a namespace, create or associate a VPC for the namespace so that all containers in the namespace will run in the VPC.	x	√	√
Viewing details of all resources on an ECS In CCE, a node is an ECS with multiple EVS disks.	√	√	√
Listing all resources on an ECS	√	√	√

Operation	CCE ReadOnlyAccess	CCE FullAccess	CCE Administrator
Viewing details about all EVS disk resources EVS disks can be attached to cloud servers and scaled to a higher capacity whenever needed.	√	√	√
Listing all EVS resources	√	√	√
Viewing details about all VPC resources A cluster must run in a VPC. When creating a namespace, create or associate a VPC for the namespace so that all containers in the namespace will run in the VPC.	√	√	√
Listing all VPC resources	√	√	√
Viewing details about all Elastic Load Balance (ELB) resources	x	x	√
Listing all ELB resources	x	x	√
Viewing Scalable File Service (SFS) resource details	√	√	√
Listing all SFS resources	√	√	√
Viewing Application Operations Management (AOM) resource details	√	√	√
Listing AOM resources	√	√	√
Performing all operations on AOM auto scaling rules	√	√	√

Namespace-level Permissions (Assigned by Using Kubernetes RBAC)

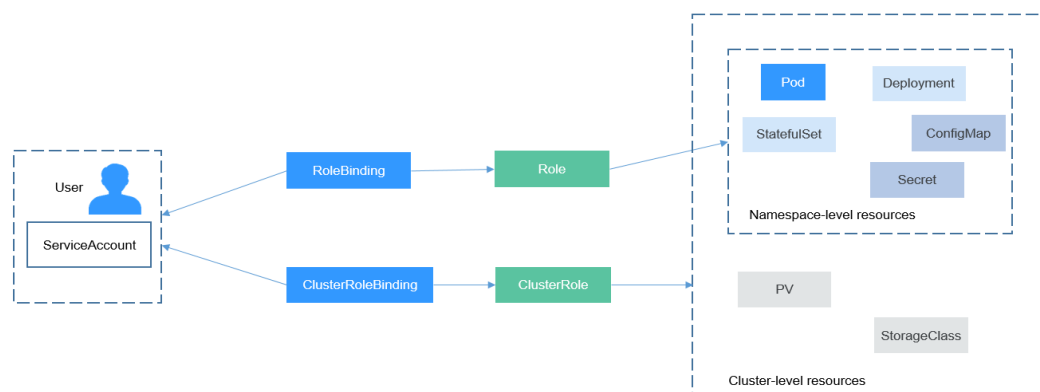
You can regulate users' or user groups' access to Kubernetes resources in a single namespace based on their Kubernetes RBAC roles. The RBAC API declares four

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. See the following figure.

Figure 1-10 Role binding



On the CCE console, you can assign permissions to a user or user group to access resources in one or all 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.
- drainage-editor: drain a node.
- drainage-viewer: view the nodal drainage status but cannot drain a node.

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.

1.6 Pricing Details

Billing Items

Cloud Container Engine (CCE) is free of charge. You only pay for the resources (such as nodes) created when you are using CCE. There are two types of billing items:

1. **Clusters:** The cluster fee is the cost of resources used by master nodes. The fee varies with the cluster type and cluster size. Cluster types include VM cluster and BMS cluster (the number of master nodes determines whether a cluster is highly available). Cluster size (also called management scale) indicates the maximum number of nodes allowed in a cluster.

 **NOTE**

The management scale indicates the number of ECSs or BMSs in a cluster.

2. **IaaS resource costs** are incurred from the IaaS resources you use to run worker nodes in your cluster. These resources, created either manually or automatically, include ECSs, EVS disks, EIPs, bandwidth, and load balancers.

Billing Modes

CCE is billed on a pay-per-use basis.

- **Pay-per-use:** It is a pay-after-use mode. Billing starts when a resource is provisioned and stops when the resource is deleted. You can use cloud resources as required and stop paying for them when you no longer need them. There is no upfront payment for excess capacity.

 **NOTE**

The following are pricing principles in the case of CCE cluster hibernation or node shutdown. Note that there are many types of cluster nodes and ECS is used as an example.

- **Cluster hibernation:** For a hibernated cluster, you will not be billed for master nodes, but the resources used by or bound to the cluster, such as the EVS disks, EIPs, and bandwidth, will be billed in their own ways such as pay-per-use.
- **Node shutdown:** Hibernating a cluster will not stop worker nodes in the cluster. You can select **Stop all nodes in the cluster** to do so. You can also stop a node on the ECS console after hibernating a cluster.

NOTICE

- Clusters follow a tiered pricing plan. Pricing for each tier varies with cluster size and type.
 - Once a pay-per-use resource becomes in arrears, you are offered a grace period and a retention period during which you can top up your account or renew resources. Within the grace period, you can still access and use your cloud service.
-

1.7 Basic Concepts

1.7.1 Basic Concepts

CCE provides highly scalable, high-performance, enterprise-class Kubernetes clusters and supports Docker containers. With CCE, you can easily deploy, manage, and scale containerized applications in the cloud.

The graphical CCE console enables E2E user experiences. In addition, CCE supports native Kubernetes APIs and kubectl. Before using CCE, you are advised to understand related basic concepts.

Cluster

A cluster is a group of one or more cloud servers (also known as nodes) in the same subnet. It has all the cloud resources (including VPCs and compute resources) required for running containers.

Node

A node is a cloud server (virtual or physical machine) running an instance of the Docker Engine. Containers are deployed, run, and managed on nodes. The node agent (kubelet) runs on each node to manage container instances on the node. The number of nodes in a cluster can be scaled.

Node Pool

A node pool contains one node or a group of nodes with identical configuration in a cluster.

Virtual Private Cloud (VPC)

A VPC is a logically isolated virtual network that facilitates secure internal network management and configurations. Resources in the same VPC can communicate with each other, but those in different VPCs cannot communicate with each other by default. VPCs provide the same network functions as physical networks and also advanced network services, such as elastic IP addresses and security groups.

Security Group

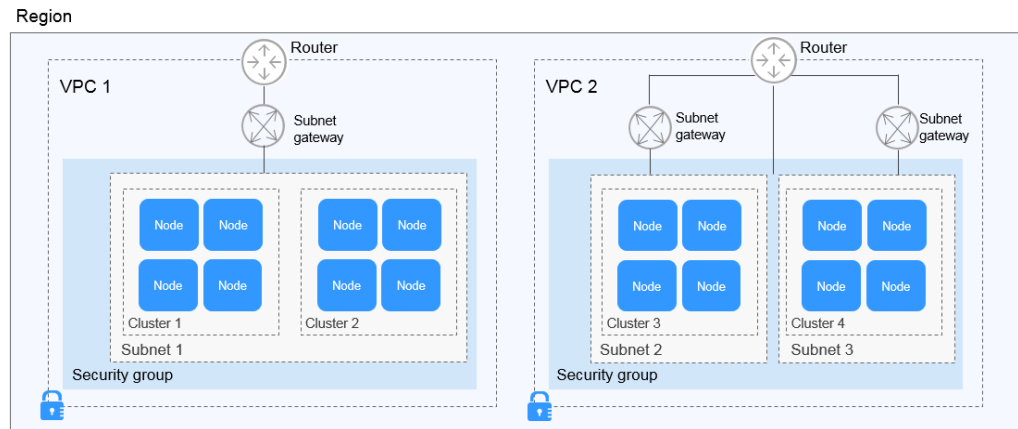
A security group is a collection of access control rules for ECSs that have the same security protection requirements and are mutually trusted in a VPC. After a security group is created, you can create different access rules for the security group to protect the ECSs that are added to this security group.

Relationship Between Clusters, VPCs, Security Groups, and Nodes

As shown in [Figure 1-11](#), a region may comprise multiple VPCs. A VPC consists of one or more subnets. The subnets communicate with each other through a subnet gateway. A cluster is created in a subnet. There are three scenarios:

- Different clusters are created in different VPCs.
- Different clusters are created in the same subnet.
- Different clusters are created in different subnets.

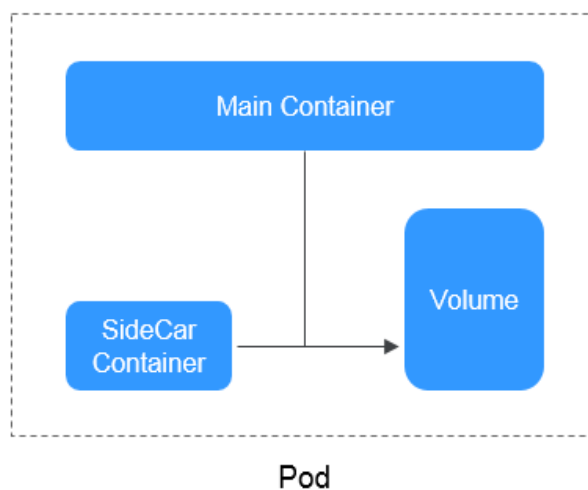
Figure 1-11 Relationship between clusters, VPCs, security groups, and nodes



Pod

A pod is the smallest and simplest unit in the Kubernetes object model that you create or deploy. A pod encapsulates an application container (or, in some cases, multiple containers), storage resources, a unique network IP address, and options that govern how the containers should run.

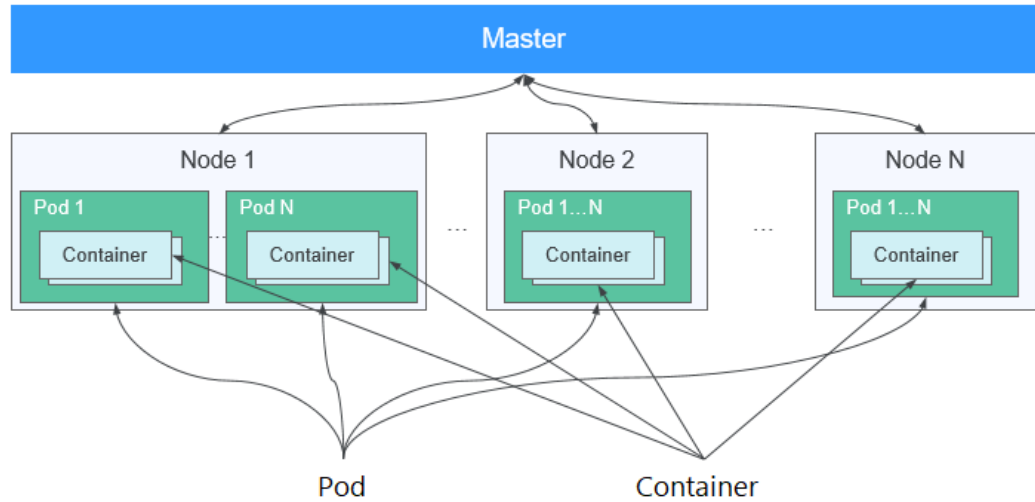
Figure 1-12 Pod



Container

A container is a running instance of a Docker image. Multiple containers can run on one node. Containers are actually software processes. Unlike traditional software processes, containers have separate namespace and do not run directly on a host.

Figure 1-13 Relationships between pods, containers, and nodes

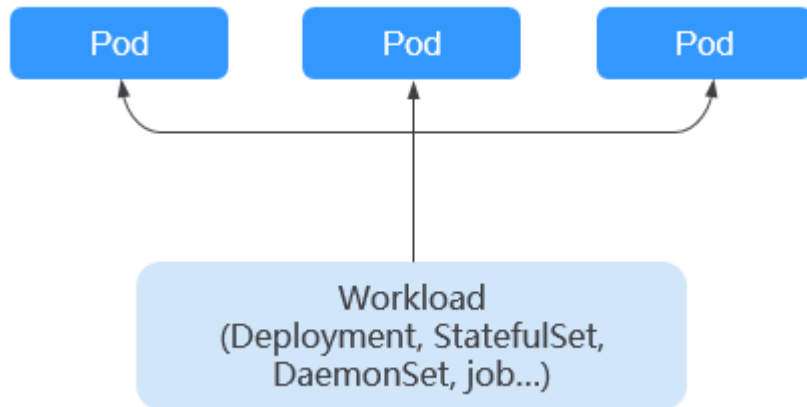


Workload

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.

- **Deployment:** Pods are completely independent of each other and functionally identical. They feature auto scaling and rolling upgrade. Typical examples include Nginx and WordPress.
- **StatefulSet:** Pods are not completely independent of each other. They have stable persistent storage, and feature orderly deployment and deletion. Typical examples include MySQL-HA and etcd.
- **DaemonSet:** A DaemonSet ensures that all or some nodes run a pod. It is applicable to pods running on every node. Typical examples include Ceph, Fluentd, and Prometheus Node Exporter.
- **Job:** It is a one-time task that runs to completion. It can be executed immediately after being created. Before creating a workload, you can execute a job to upload an image to the image repository.
- **Cron job:** It runs a job periodically on a given schedule. You can perform time synchronization for all active nodes at a fixed time point.

Figure 1-14 Relationship between workloads and pods

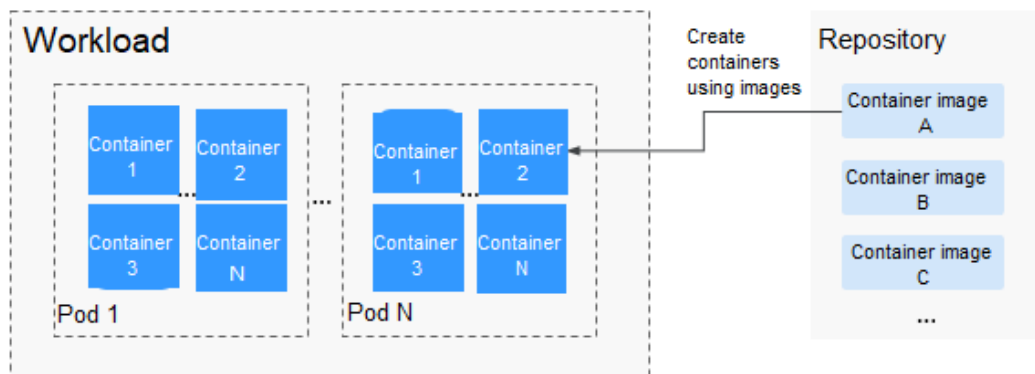


Image

Docker creates an industry standard for packaging containerized applications. Docker images are like templates that include everything needed to run containers, and are used to create Docker containers. In other words, Docker image is a special file system that includes everything needed to run containers: programs, libraries, resources, and configuration files. It also contains configuration parameters (such as anonymous volumes, environment variables, and users) required within a container runtime. An image does not contain any dynamic data. Its content remains unchanged after being built. When deploying containerized applications, you can use images from Docker Hub, SoftWare Repository for Container (SWR), and your private image registries. For example, a Docker image can contain a complete Ubuntu operating system, in which only the required programs and dependencies are installed.

Images become containers at runtime, that is, containers are created from images. Containers can be created, started, stopped, deleted, and suspended.

Figure 1-15 Relationship between images, containers, and workloads



Namespace

A namespace is an abstract collection of resources and objects. It enables resources to be organized into non-overlapping groups. 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. Examples:

- You can deploy workloads in a development environment into one namespace, and deploy workloads in a test environment into another namespace.
- Pods, Services, ReplicationControllers, and Deployments belong to a namespace (named **default**, by default), whereas nodes and PersistentVolumes do not belong to any namespace.

Service

A Service is an abstract method that exposes a group of applications running on a pod as network services.

Kubernetes provides you with a service discovery mechanism without modifying applications. In this mechanism, Kubernetes provides pods with their own IP addresses and a single DNS for a group of pods, and balances load between them.

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 Service, as the default Service type, is exposed through the internal IP address of the cluster. If this mode is selected, Services can be accessed only within the cluster.
- **NodePort:** NodePort Services are exposed through the IP address and static port of each node. A ClusterIP Service, to which a NodePort Service will route, is automatically created. By sending a request to <NodeIP>:<NodePort>, you can access a NodePort Service from outside of a cluster.
- **LoadBalancer (ELB):** LoadBalancer (ELB) Services are exposed by using load balancers of the cloud provider. External load balancers can route to NodePort and ClusterIP Services.

Layer-7 Load Balancing (Ingress)

An ingress is a set of routing rules for requests entering a cluster. It provides Services with URLs, load balancing, SSL termination, and HTTP routing for external access to the cluster.

Network Policy

Network policies provide policy-based network control to isolate applications and reduce the attack surface. A network policy uses label selectors to simulate traditional segmented networks and controls traffic between them and traffic from outside.

ConfigMap

A ConfigMap is used to store configuration data or configuration files as key-value pairs. ConfigMaps are similar to secrets, but provide a means of working with strings that do not contain sensitive information.

Secret

Secrets resolve the configuration problem of sensitive data such as passwords, tokens, and keys, and will not expose the sensitive data in images or pod specs. A secret can be used as a volume or an environment variable.

Label

A label is a key-value pair and is associated with an object, for example, a pod. Labels are used to identify special features of objects and are meaningful to users. However, labels have no direct meaning to the kernel system.

Label Selector

Label selector is the core grouping mechanism of Kubernetes. It identifies a group of resource objects with the same characteristics or attributes through the label selector client or user.

Annotation

Annotations are defined in key-value pairs as labels are.

Labels have strict naming rules. They define the metadata of Kubernetes objects and are used by label selectors.

Annotations are additional user-defined information for external tools to search for a resource object.

PersistentVolume

A PersistentVolume (PV) is a network storage in a cluster. Similar to a node, it is also a cluster resource.

PersistentVolumeClaim

A PV is a storage resource, and a PersistentVolumeClaim (PVC) is a request for a PV. PVC is similar to pod. Pods consume node resources, and PVCs consume PV resources. Pods request CPU and memory resources, and PVCs request data volumes of a specific size and access mode.

Auto Scaling - HPA

Horizontal Pod Autoscaling (HPA) is a function that implements horizontal scaling of pods in Kubernetes. The scaling mechanism of ReplicationController can be used to scale your Kubernetes clusters.

Affinity and Anti-Affinity

If an application is not containerized, multiple components of the application may run on the same virtual machine and processes communicate with each other. However, in the case of containerization, software processes are packed into different containers and each container has its own lifecycle. For example, the transaction process is packed into a container while the monitoring/logging process and local storage process are packed into other containers. If closely

related container processes run on distant nodes, routing between them will be costly and slow.

- **Affinity:** Containers are scheduled onto the nearest node. For example, if application A and application B frequently interact with each other, it is necessary to use the affinity feature to keep the two applications as close as possible or even let them run on the same node. In this way, no performance loss will occur due to slow routing.
- **Anti-affinity:** Instances of the same application spread across different nodes to achieve higher availability. Once a node is down, instances on other nodes are not affected. For example, if an application has multiple replicas, it is necessary to use the anti-affinity feature to deploy the replicas on different nodes. In this way, no single point of failure will occur.

Node Affinity

By selecting labels, you can schedule pods to specific nodes.

Node Anti-Affinity

By selecting labels, you can prevent pods from being scheduled to specific nodes.

Pod Affinity

You can deploy pods onto the same node to reduce consumption of network resources.

Pod Anti-Affinity

You can deploy pods onto different nodes to reduce the impact of system breakdowns. Anti-affinity deployment is also recommended for workloads that may interfere with each other.

Resource Quota

Resource quotas are used to limit the resource usage of users.

Resource Limit (LimitRange)

By default, all containers in Kubernetes have no CPU or memory limit. **LimitRange** (**limits** for short) is used to add a resource limit to a namespace, including the minimum, maximum, and default amounts of resources. When a pod is created, resources are allocated according to the **limits** parameters.

Environment Variable

An environment variable is a variable whose value can affect the way a running container will behave. A maximum of 30 environment variables can be defined at container creation time. 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.

Chart

For your Kubernetes clusters, you can use [Helm](#) to manage software packages, which are called charts. Helm is to Kubernetes what the apt command is to Ubuntu or what the yum command is to CentOS. Helm can quickly search for, download, and install charts.

Charts are a Helm packaging format. It describes only a group of related cluster resource definitions, not a real container image package. A Helm chart contains only a series of YAML files used to deploy Kubernetes applications. You can customize some parameter settings in a Helm chart. When installing a chart, Helm deploys resources in the cluster based on the YAML files defined in the chart. Related container images are not included in the chart but are pulled from the image repository defined in the YAML files.

Application developers need to push container image packages to the image repository, use Helm charts to package dependencies, and preset some key parameters to simplify application deployment.

Helm directly installs applications and their dependencies in the cluster based on the YAML files in a chart. Application users can search for, install, upgrade, roll back, and uninstall applications without defining complex deployment files.

1.7.2 Mappings Between CCE and Kubernetes Terms

Kubernetes (K8s) is an open-source system for automating deployment, scaling, and management of container clusters. It is a container orchestration tool and a leading solution based on the distributed architecture of the container technology. Kubernetes is built on the open-source Docker technology that automates deployment, resource scheduling, service discovery, and dynamic scaling of containerized applications.

This topic describes the mappings between CCE and Kubernetes terms.

Table 1-5 Mappings between CCE and Kubernetes terms

CCE	Kubernetes
Cluster	Cluster
Node	Node
Node pool	NodePool
Container	Container
Image	Image
Namespace	Namespace
Deployment	Deployment
StatefulSet	StatefulSet
DaemonSet	DaemonSet
Job	Job

CCE	Kubernetes
Cron job	CronJob
Pod	Pod
Service	Service
ClusterIP	Cluster IP
NodePort	NodePort
LoadBalancer	LoadBalancer
Layer-7 load balancing (ingress)	Ingress
Network policy	NetworkPolicy
Chart	Template
ConfigMap	ConfigMap
Secret	Secret
Label	Label
Label selector	LabelSelector
Annotation	Annotation
Volume	PersistentVolume
PersistentVolumeClaim	PersistentVolumeClaim
Auto scaling	HPA
Node affinity	NodeAffinity
Node anti-affinity	NodeAntiAffinity
Pod affinity	PodAffinity
Pod anti-affinity	PodAntiAffinity
Webhook	Webhook
Endpoint	Endpoint
Quota	Resource Quota
Resource limit	Limit Range

1.7.3 CCE Turbo Cluster

CCE launches its next-gen container cluster, CCE Turbo, for container deployments at scale. Performance, scaling, and scheduling all upgrade to the next level.

CCE Turbo clusters feature accelerated computing, networking, and scheduling. These features further free you from managing service running, and enable easier innovations.

Cluster Advantages

- Accelerated computing**
 On a software-hardware collaborative infrastructure, computing consumes fewer but performs better.
- Accelerated networking**
 The Cloud Native Network 2.0 model flattens two network layers into one. Such lossless networking allows applications to better communicate.
- Accelerated scheduling**
 Application and resource scheduling becomes more intelligent and faster.
- Security isolation**
 The Kata container engine provides VM-level security isolation for applications.

Comparison Between CCE Turbo Clusters and CCE Clusters

The following table lists the differences between CCE Turbo clusters and CCE clusters:

Table 1-6 Cluster types

Dimension	Sub-dimension	CCE Turbo Cluster	CCE Cluster
Cluster	Positioning	Next-gen container cluster, with accelerated computing, networking, and scheduling. Designed for Cloud Native 2.0	Standard cluster for common commercial use
	Node type	Deployment of VMs	Hybrid deployment of VMs and BMSs
Networking	Model	Cloud Native Network 2.0: applies to large-scale and high-performance scenarios. Max networking scale: 2,000 nodes	Cloud-native network 1.0: applies to common, smaller-scale scenarios. <ul style="list-style-type: none"> Tunnel network model VPC network model
	Performance	Flattens the VPC network and container network into one. No performance loss.	Overlays the VPC network with the container network, causing certain performance loss.

Dimension	Sub-dimension	CCE Turbo Cluster	CCE Cluster
	Container network isolation	Associates pods with security groups. Unifies security isolation in and out the cluster via security groups' network policies.	<ul style="list-style-type: none"> Tunnel network model: supports network policies for intra-cluster communications. VPC network model: supports no isolation.
Security	Isolation	<ul style="list-style-type: none"> VM: runs common containers, isolated by cgroups. 	Runs common containers, isolated by cgroups.

1.7.4 Regions and AZs

Definition

A region and availability zone (AZ) identify the location of a data center. You can create resources in a specific region and AZ.

- Regions are divided based on geographical location and network latency. Public services, such as Elastic Cloud Server (ECS), Elastic Volume Service (EVS), Object Storage Service (OBS), Virtual Private Cloud (VPC), Elastic IP (EIP), and Image Management Service (IMS), are shared within the same region. Regions are classified as universal regions and dedicated regions. A universal region provides universal cloud services for common domains. A dedicated region provides services of the same type only or for specific domains.
- An AZ contains one or more physical data centers. Each AZ has independent cooling, fire extinguishing, moisture-proof, and electricity facilities. Within an AZ, computing, network, storage, and other resources are logically divided into multiple clusters. AZs in a region are interconnected through high-speed optic fibers. This is helpful if you will deploy systems across AZs to achieve higher availability.

Cloud services are available in many regions around the world. You can select a region and AZ as needed.

How to Select a Region?

When selecting a region, consider the following factors:

- Location
Select a region close to you or your target users to reduce network latency and improve access rate.

Selecting an AZ

When deploying resources, consider your applications' requirements on disaster recovery (DR) and network latency.

- For high DR capability, deploy resources in different AZs within the same region.
- For lower network latency, deploy resources in the same AZ.

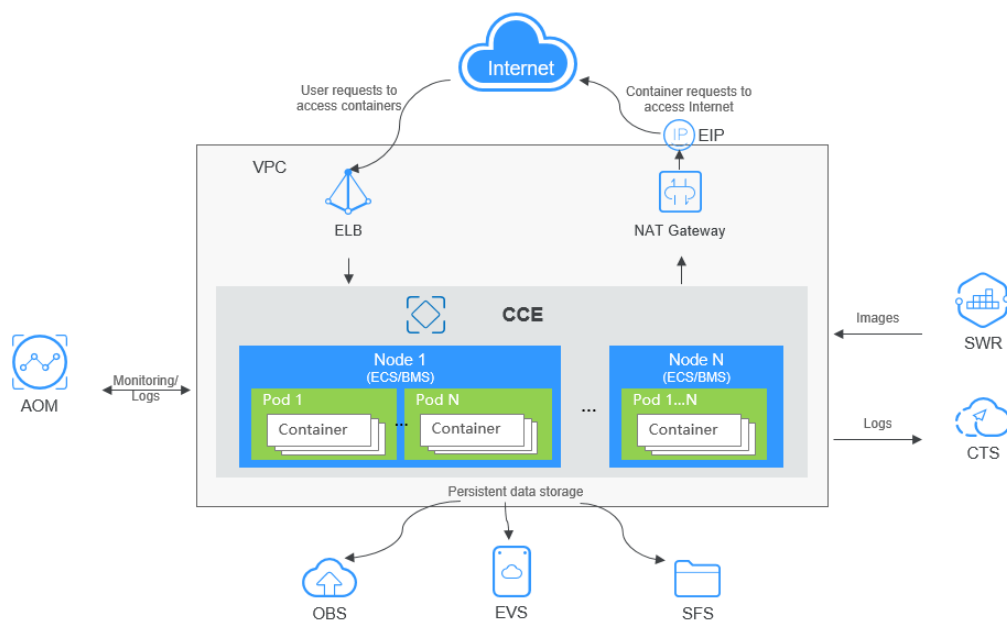
Regions and Endpoints

When using an API to access resources, you must specify a region and endpoint. For more information about regions and endpoints, see [Regions and Endpoints](#).

1.8 Related Services

CCE works with the following cloud services and requires permissions to access them.

Figure 1-16 Relationships between CCE and other services



Relationships Between CCE and Other Services

Table 1-7 Relationships between CCE and other services

Service	Relationship
Elastic Cloud Server (ECS)	An ECS with multiple EVS disks is a node in CCE. You can choose ECS specifications during node creation.
Virtual Private Cloud (VPC)	For security reasons, all clusters created by CCE must run in VPCs. When creating a namespace, create a VPC or bind an existing VPC to the namespace so all containers in the namespace will run in this VPC.
Elastic Load Balance (ELB)	CCE works with ELB to load balance a workload's access requests across multiple pods.

Service	Relationship
NAT Gateway	The NAT Gateway service provides source network address translation (SNAT) for container instances in a VPC. The SNAT feature translates private IP addresses of these container instances to the same EIP, which is a public IP address reachable on Internet.
Software Repository for Container (SWR)	An image repository is used to store and manage Docker images.
Elastic Volume Service (EVS)	EVS disks can be attached to cloud servers and scaled to a higher capacity whenever needed. An ECS with multiple EVS disks is a node in CCE. You can choose ECS specifications during node creation.
Object Storage Service (OBS)	OBS provides stable, secure, cost-efficient, and object-based cloud storage for data of any size. With OBS, you can create, modify, and delete buckets, as well as uploading, downloading, and deleting objects. CCE allows you to create an OBS volume and attach it to a path inside a container.
Scalable File Service (SFS)	SFS is a shared, fully managed file storage service. Compatible with the Network File System protocol, SFS file systems can elastically scale up to petabytes, thereby ensuring top performance of data-intensive and bandwidth-intensive applications. You can use SFS file systems as persistent storage for containers and attach the file systems to containers when creating a workload.
Application Operations Management (AOM)	AOM collects container log files in formats like .log from CCE and dumps them to AOM. On the AOM console, you can easily query and view log files. In addition, AOM monitors CCE resource usage. You can define metric thresholds for CCE resource usage to trigger auto scaling.
Cloud Trace Service (CTS)	CTS records operations on your cloud resources, allowing you to query, audit, and backtrack resource operation requests initiated from the management console or open APIs as well as responses to these requests.

2 Product Bulletin

2.1 Risky Operations on Cluster Nodes

Precautions for Using a Cluster

- When performing operations such as creating, deleting, and scaling clusters, do not change user permissions on the Identity and Access Management (IAM) console. Otherwise, these operations may fail.
- Canal, the CNI plug-ins used by CCE nodes, uses a CIDR block as the CIDR block of the container network. This CIDR block can be configured during cluster creation and defaults to 172.16.0.0/16. The Docker service creates a docker0 bridge by default. The default docker0 address is 172.17.0.1. When creating a cluster, ensure that the CIDR block of the VPC in the cluster is different from those of the container network and the docker0 bridge. If VPC peering connections are used, ensure that the CIDR block of the peer VPC is different from those of the container network and the docker0 bridge.
- For clusters of Kubernetes v1.15, the DNS server of nodes in the cluster uses the DNS server in the VPC subnet. The CoreDNS address of Kubernetes is not added. Ensure that the DNS address in the subnet exists and is configurable.
- For clusters of Kubernetes v1.17, the network of a node is a single network plane. In the multi-network plane scenario, if you bind a new NIC to the ECS, you need to configure the NIC information on the node and restart the NIC after the binding.
- Do not modify the security groups, Elastic Volume Service (EVS) disks, and other resources created by CCE. Otherwise, clusters may not function properly. The resources created by CCE are labeled "cce", for example, "cce-evs-jwh9pcl7-****".
- When a node is added, the DNS server in the subnet must be able to resolve the domain name of the corresponding service. Otherwise, the node cannot be installed properly.

Precautions for Using a Node

Some of the node resources will be used to run necessary Kubernetes system components and resources to make the node as part of your cluster. Therefore, the

amount of your node resources differs from that of node allocatable resources in Kubernetes. 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 amount of CCE node resources will be reserved for Kubernetes components (such as kubelet, kube-proxy, and docker) based on the node specifications.

 **NOTE**

You are advised not to install private software or modify the operating system (OS) configuration on a cluster node. This may cause exceptions on Kubernetes components installed on the node, and make the node unavailable.

Risky Operations on Nodes

After logging in to a node created by CCE, do not perform the following operations. Otherwise, the node will become unready.

Table 2-1 Operations that will cause nodes to become unready

No.	Operation	Impact	Solution
1	Reinstalling the operating system using the original image or another image	The node will become unavailable.	Delete the node and create a new one.
2	Modifying OS configuration	The node will become unavailable.	Restore the original configuration or create a new node.
3	Deleting the opt directory, /var/paas directory, or a data disk	The node will become unavailable.	Delete the node and create a new one.
4	Formatting and partitioning a node disk	The node will become unavailable.	Delete the node and create a new one.
5	Modifying a security group	The node will become unready or the cluster will exhibit unexpected behavior.	Correct the security group settings based on security group settings of normal clusters.

2.2 CCE Security Guide

Node Security Hardening

Cloud Container Engine (CCE) cluster nodes are created from the public images published by Image Management Service (IMS) in the public cloud. CCE performs

security hardening on the CCE software installed on nodes. The OS configurations are the same as those of public images by default. You need to perform security hardening based on your own security requirements.

Container Security Suggestions

CCE provides you with an exclusive cluster. You are not advised to share a cluster with multiple tenants. If a cluster must be shared by multiple tenants, perform security hardening by following the suggestions in the following table.

Table 2-2 Container security configuration suggestions

Category	Suggestion
Least privilege	<p>1. Use the sudo whitelist to restrict the Docker commands that can be run in containers by common users.</p> <p>2. It is recommended that privileged containers be disabled for Kubernetes pods. To be specific, do not use the following settings:</p> <pre data-bbox="671 864 1428 1149">apiVersion: v1 kind: Pod metadata: name: hello-world spec: containers: - name: hello-world-container # The container definition # ... securityContext: privileged: true</pre> <p>By default, the token of ServiceAccount default is not used.</p> <ul style="list-style-type: none"> The automatic mounting function is disabled for ServiceAccount default by default. If automountServiceAccountToken is set to false, the created workload will not be mounted to a token by default. <p>CAUTION Set this field for each namespace as required. After the setting is complete, perform a rolling restart for the pods in the corresponding namespace.</p> <pre data-bbox="671 1514 1428 1641">apiVersion: v1 kind: ServiceAccount metadata: name: default automountServiceAccountToken: false</pre> <ul style="list-style-type: none"> Mount the token of ServiceAccount default to specific workloads. <pre data-bbox="671 1720 1428 1899">... spec: template: spec: serviceAccountName: default automountServiceAccountToken: true ...</pre>

Category	Suggestion
Network isolation	<p>Network isolation between containers in a cluster Use network policies to implement access control.</p> <p>Network isolation between containers and hosts outside the cluster Use security groups in a VPC to implement access control.</p> <p>Network isolation between containers and the cluster control plane CCE clusters are private and workloads would access the cluster API server. Therefore, CCE does not restrict the network communication between containers and the Kubernetes control plane. You can harden the network security by limiting the pod QoS.</p>
Mounting of sensitive host directories	<ul style="list-style-type: none"> • If not necessary, do not mount the <code>/var/run/docker.sock</code> file on the host to a container when starting the container. • Do not mount the hostPath directory, such as <code>/</code>, <code>/boot</code>, <code>/dev</code>, <code>/etc</code>, <code>/lib</code>, <code>/proc</code>, <code>/sys</code>, and <code>/usr</code>, on the host to the container. If necessary, mount the hostPath directory in read-only mode.

2.3 Cluster Node OS Patch Notes

CCE Cluster

In a CCE cluster, the OSs and the corresponding kernel versions supported by nodes are as follows:

Table 2-3 Node OS kernel information

OS	Cluster Version	Kernel
EulerOS release 2.9	v1.27	4.18.0-147.5.1.6.h541.eulerosv2r9.x86_64
	v1.25	4.18.0-147.5.1.6.h541.eulerosv2r9.x86_64
	v1.23	4.18.0-147.5.1.6.h541.eulerosv2r9.x86_64
	v1.21	4.18.0-147.5.1.6.h541.eulerosv2r9.x86_64
	v1.19.10	4.18.0-147.5.1.6.h541.eulerosv2r9.x86_64
EulerOS release 2.5	v1.25	3.10.0-862.14.1.0.h197.eulerosv2r7.x86_64

OS	Cluster Version	Kernel
	v1.23	3.10.0-862.14.1.0.h197.eulerosv 2r7.x86_64
	v1.21	3.10.0-862.14.1.0.h197.eulerosv 2r7.x86_64
	v1.19.10	3.10.0-862.14.1.0.h197.eulerosv 2r7.x86_64
	v1.17.9	3.10.0-862.14.1.5.h470.eulerosv 2r7.x86_64
	v1.15.11	3.10.0-862.14.1.5.h470.eulerosv 2r7.x86_64
CentOS Linux release 7.7	v1.23	3.10.0-1062.12.1.el7.x86_64
	v1.21	3.10.0-1062.12.1.el7.x86_64
	v1.19.10	3.10.0-1062.12.1.el7.x86_64
	v1.17.9	3.10.0-1062.12.1.el7.x86_64
Ubuntu 22.04 (only the VPC network model is supported)	v1.25	5.15.0-46-generic
	v1.23	5.15.0-46-generic

The OS patch release and verification results will be updated periodically. You can update the operating system if needed.

2.4 Vulnerability Notice

2.4.1 Notice on the Kubernetes Security Vulnerability (CVE-2022-3172)

Recently, the Kubernetes community detected a security issue in kube-apiserver. This issue allows the aggregated API server to redirect client traffic to any URL, which may cause the client to perform unexpected operations and forward the client's API server credentials to a third party.

Vulnerability Details

Table 2-4 Vulnerability information

Vulnerability Type	CVE-ID	Discovered
SSRF	CVE-2022-3172	2022-09-09

Threat Severity

Medium

Impact Scope

Affected versions:

- kube-apiserver <= v1.23.10

CCE clusters of the preceding versions configured with the aggregated API server will be affected, especially for CCE clusters with logical multi-tenancy.

Identification Method

For CCE clusters and CCE Turbo clusters of version 1.23 or earlier, use `kubectl` to connect to the cluster. Run the following command to check whether the aggregated API server is running:

```
kubectl get apiservices.apiregistration.k8s.io -o=jsonpath='{range .items[?(@.spec.service)]}{.metadata.name}{\n}{end}'
```

If the returned value is not empty, the aggregated API server exists.

Preventive Measures

Upgrades are the currently available solution. The cluster administrator must control permissions to prevent untrusted personnel from deploying and controlling the aggregated API server through the API service interface. A fixed version will be provided soon. Please follow the official announcement.

Helpful Links

<https://github.com/kubernetes/kubernetes/issues/112513>

2.4.2 Privilege Escalation Vulnerability in Linux openvswitch Kernel Module (CVE-2022-2639)

A privilege escalation vulnerability (CVE-2022-2639) was found in the Linux openvswitch kernel module. The `reserve_sfa_size()` function in this module has a defect. As a result, a local user can exploit this vulnerability to escalate their privileges on the system. Currently, the POC of this vulnerability have been disclosed and the risk is high.

Vulnerability Details

Table 2-5 Vulnerability information

Vulnerability Type	CVE-ID	Discovered
Privilege escalation	CVE-2022-2639	2022-09-01

Threat Severity

Critical

Affected Products

1. CCE clusters that use the container tunnel network model; node OS images that use EulerOS 2.9;

2. Node OS images that use Ubuntu

Cluster nodes running EulerOS 2.5 and CentOS 7.6 **are not affected by this vulnerability.**

Workarounds and Mitigation Measures

1. If a process in a container is started by a non-root user, you can configure seccomp, the security computing mode, for the workload. You are advised to use the RuntimeDefault mode or disable system calls such as unshare. For details about the configuration, see the community documentation [Restrict a Container's Syscalls with seccomp](#).

2. Ubuntu images are embedded with the openvswitch kernel module. You can disable the loading of this module to avoid this problem. The procedure is as follows:

```
echo "blacklist openvswitch" >>/etc/modprobe.d/blacklist.conf
```

Then, restart the node for the settings to take effect.

Helpful Links

<https://github.com/torvalds/linux/commit/cefa91b2332d7009bc0be5d951d6cbbf349f90f8>

2.4.3 Notice on CRI-O Container Runtime Engine Arbitrary Code Execution Vulnerability (CVE-2022-0811)

The CrowdStrike security team disclosed a security vulnerability (tracked as CVE-2022-0811) in CRI-O v1.19. An attacker can exploit this vulnerability to escape to the host.

Vulnerability Details

Table 2-6 Information

Vulnerability Type	CVE-ID	Discovered
Container escape	CVE-2022-0811	2021-03-16

Threat Severity

Critical

Root Cause

A vulnerability is introduced in CRI-O v1.19. An attacker can exploit this vulnerability to bypass protection measures and set arbitrary kernel parameters on the host. As a result, any user with permissions to deploy a pod on a Kubernetes cluster that uses CRI-O runtime can abuse the **kernel.core_pattern** kernel parameter to achieve container escape and arbitrary code execution as root on any node in the cluster.

Impact Scope

1. Kubernetes clusters that use CRI-O v1.19 or later, including patch versions 1.19.6, 1.20.7, 1.21.6, 1.22.3, 1.23.2, and 1.24.0.

CCE clusters are not affected by this vulnerability because they do not use CRI-O.

Workarounds and Mitigation Measures

1. For CRI-O v1.19 and v1.20, set **manage_ns_lifecycle** to **false**, and use Open Container Initiative (OCI) runtimes to configure sysctls.
2. Create a PodSecurityPolicy and set all sysctls to **false**.
3. Upgrade the CRI-O runtime to the latest version.

References

1. Red Hat community vulnerability notice: <https://access.redhat.com/security/cve/cve-2022-0811>
2. *cr8escape: New Vulnerability in CRI-O Container Engine Discovered by CrowdStrike*: <https://www.crowdstrike.com/blog/cr8escape-new-vulnerability-discovered-in-cri-o-container-engine-cve-2022-0811/>

2.4.4 Notice on the Container Escape Vulnerability Caused by the Linux Kernel (CVE-2022-0492)

Recently, the security team detected that in some scenarios, the **release_agent** feature of the Linux kernel cgroup v1 can be exploited to escalate privileges and bypass the namespace isolation unexpectedly.

Vulnerability Details

Table 2-7 Vulnerability information

Vulnerability Type	CVE-ID	Discovered
Container escape	CVE-2022-0492	2021-02-07

Threat Severity

Critical

Trigger Conditions

On an affected node, workloads use the **root** user to run a process (or has the **CAP_SYS_ADMIN** permission), and **seccomp** is not configured.

Root Cause

The Linux kernel does not check whether the process that sets the **release_agent** file has the proper permission.

Impact Scope

1. For x86 nodes, EulerOS 2.5 and CentOS images are not affected by this vulnerability.
2. EulerOS (Arm) whose kernel version is earlier than 4.19.36-vhulk1907.1.0.h962.eulerosv2r8.aarch64 is affected by this vulnerability.
3. EulerOS (x86) whose kernel version is earlier than 4.18.0-147.5.1.6.h541.eulerosv2r9.x86_64 is affected by this vulnerability.
4. Ubuntu nodes whose kernel version is 4.15.0-136-generic or earlier is affected by this vulnerability.

Workarounds and Mitigation Measures

1. A fix version has been provided for EulerOS 2.9 images. Migrate to the 4.18.0-147.5.1.6.h541.eulerosv2r9.x86_64 nodes as soon as possible.
2. Configure **seccomp** for workloads to restrict unshare system calls. For details, see the [Kubernetes documentation](#).
3. Restrict the process permissions in a container and minimize the process permissions in the container. For example, use a non-root user to start processes and use the **capability** mechanism to refine the process permissions.

References

1. Kernel repair commit: <https://git.kernel.org/pub/scm/linux/kernel/git/torvalds/linux.git/commit/?id=24f6008564183aa120d07c03d9289519c2fe02af>
2. Red Hat community vulnerability notice: <https://access.redhat.com/security/cve/cve-2022-0492>

2.4.5 Linux Kernel Integer Overflow Vulnerability (CVE-2022-0185)

Recently, the security team detected that security researchers William Liu and Jamie Hill-Daniel discovered an integer underflow vulnerability in the Linux kernel, which may lead to out-of-bounds writes. A local attacker can use this vulnerability to cause a denial of service (system crash) or execute arbitrary code. In a container scenario, a user with the **CAP_SYS_ADMIN** permission can escape from the container to the host machine. Currently, the vulnerability POC already exists, but no disclosed exploit code is found.

Vulnerability Details

Table 2-8 Vulnerability information

Vulnerability Type	CVE-ID	Discovered
Resource management flaw	CVE-2022-0185	2022-01-27

Threat Severity

Critical

Impact and Risks

The system that uses Linux kernel 5.1 or later will be affected by this vulnerability, but **CCE is not affected**.

Trigger Conditions

In a container scenario, users have the CAP_SYS_ADMIN permission, and the kernel version is 5.1 or later. In a standard Docker environment, the Docker seccomp filter is used. Therefore, the system is not affected by this vulnerability by default. In the Kubernetes scenario, the seccomp filter is disabled by default. The system is affected by this vulnerability if the kernel and permission conditions are met.

Identification Method

Run the **uname -a** command to view the kernel version.

Workarounds and Mitigation Measures

CCE clusters are not affected by this vulnerability. For a Kubernetes cluster, you are advised to:

1. Run containers with the least privilege.
2. Configure [seccomp](#) based on the configuration method provided by Kubernetes.

References

<https://blog.aquasec.com/cve-2022-0185-linux-kernel-container-escape-in-kubernetes>

<https://ubuntu.com/security/CVE-2022-0185>

<https://access.redhat.com/security/cve/CVE-2022-0185>

<https://www.openwall.com/lists/oss-security/2022/01/18/7>

3 Kubernetes Basics

3.1 Overview

Kubernetes is an open source container orchestration system for automating containerized application deployment, scaling, and management across cloud servers.

One of the core features of Kubernetes is that it can independently manage containers to ensure that the containers run as expected. For example, if you want your Apache application to run all the time, Kubernetes will automatically monitor the running status, restart containers, and create new containers to keep Apache running. Administrator can load a microservice for the service planner. In addition, Kubernetes systematically improves tools and makes itself more user-friendly to deploy applications.

You can access CCE, a hosted Kubernetes service, using the CCE console, kubectl, or Kubernetes APIs. Before using CCE, you can learn the following Kubernetes concepts.

Containers and Kubernetes

- [Containers](#)
- [Kubernetes](#)

Pods, Labels, and Namespaces

- [Pod: the Smallest Scheduling Unit in Kubernetes](#)
- [Liveness Probe](#)
- [Label for Managing Pods](#)
- [Namespaces: Grouping Resources](#)

Pod Orchestration and Scheduling

- [Deployment](#)
- [StatefulSet](#)
- [Job and Cron Job](#)

- [DaemonSet](#)
- [Affinity and Anti-Affinity Scheduling](#)

Configuration Management

- [ConfigMap](#)
- [Secret](#)

Kubernetes Networking

- [Container Networking](#)
- [Services](#)
- [Ingresses](#)
- [Readiness Probe](#)
- [NetworkPolicy](#)

Persistent Storage

- [Volume](#)
- [PersistentVolumes, PersistentVolumeClaims, and StorageClasses](#)

Authentication and Authorization

- [ServiceAccounts](#)
- [RBAC](#)

Auto Scaling

- [Auto Scaling](#)

3.2 Container and Kubernetes

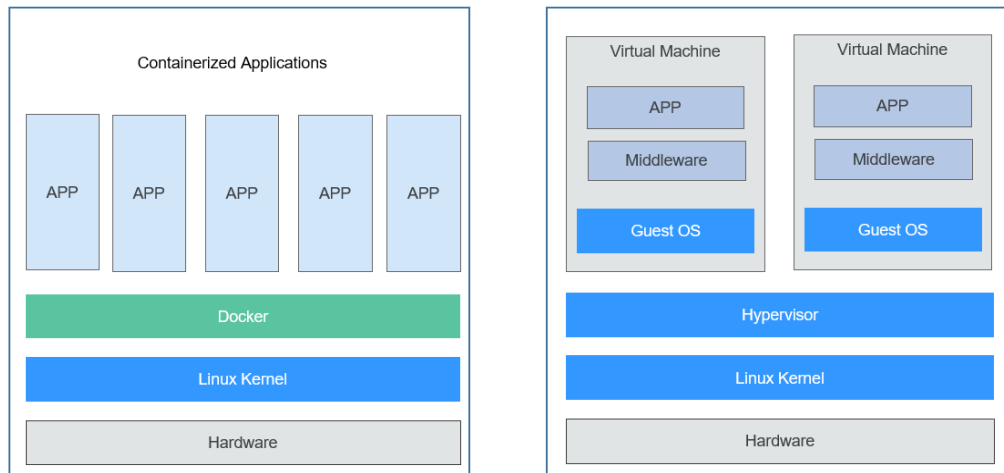
3.2.1 Containers

Container and Docker

Container technologies originate from Linux. Containers provide lightweight virtualization and allow process and resource isolation. Containers become popular since the emergence of Docker. Docker is the first system that allows containers to be portable in different machines. It simplifies both the application packaging and the application library and dependency packaging. Even the OS file system can be packaged into a simple portable package, which can be used on any other machine that runs Docker.

Except for similar resource isolation and allocation modes as VMs, containers virtualize OSs, making them more portable and efficient.

Figure 3-1 Containers vs VMs



Containers have the following advantages over VMs:

- **Higher system resource utilization**
With no overhead for virtualizing hardware and running a complete OS, containers outperform VMs no matter in application execution speed, memory loss, and file storage speed. Therefore, with same configurations, containers can run more applications than VMs.
- **Faster startup**
Traditional VMs usually take several minutes to start an application. However, Docker containerized applications run directly on the host kernel with no need to start the entire OS, so they can start within seconds or even milliseconds, greatly saving your time in development, testing, and deployment.
- **Consistent running environments**
One of the biggest problems in development is the inconsistency of application running environment. Due to inconsistent development, testing, and production environments, some bugs cannot be discovered prior to rollout. A Docker container image provides a complete runtime to ensure consistency in application running environments.
- **Easier migration**
Docker ensures the consistency in execution environment, so migrating applications becomes much easier. Docker can run on many platforms, and no matter on physical machines or virtual ones, its running results remains the same. Therefore, you can easily migrate an application from one platform to another without worrying that the environment change will cause the applications fail to function.
- **Easier maintenance and extension**
Tiered storage and image technology applied by Docker facilitate the reuse of applications and simplify application maintenance and update as well as further image extension based on base images. In addition, Docker collaborates with open-source project teams to maintain a large number of high-quality official images. You can directly use them in the production environment or form new images based on them, greatly reducing the image production cost of applications.

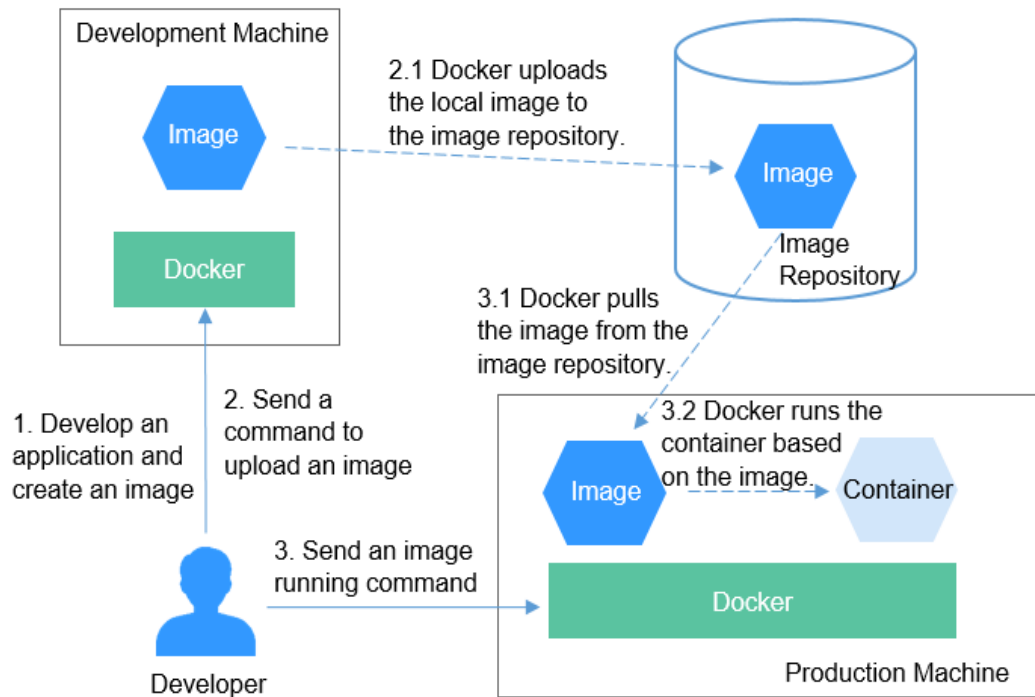
Typical Process of Using Docker Containers

Before using a Docker container, you should know the core components in Docker.

- **Image:** A Docker image is a software package that contains everything needed to run an application, such as the code and the runtime it requires, file systems, and executable file path of the runtime and other metadata.
- **Image repository:** A Docker image repository is used to store Docker images, which can be shared between different users and computers. You can run the image you compiled on the computer where it is compiled, or upload it to an image repository and then download it to another computer and run it. Some repositories are public, allowing everyone to pull images from them. Others are private, which are accessible only to some users and machines.
- **Container:** A Docker container is usually a Linux container created from a Docker image. A running container is a process running on the Docker host. However, it is isolated from the host and all other processes running on the host. The process is also resource-limited, meaning that it can access and use only resources (such as CPU and memory) allocated to it.

Figure 3-2 shows the typical process of using containers.

Figure 3-2 Typical process of using Docker containers



1. A developer develops an application and creates an image in the development machine.
Docker runs the commands to create an image and store it on the machine.
2. The developer sends a command to upload the image.
After receiving the command, Docker uploads the local image to the image repository.

3. The developer sends an image running command to the machine.
After the command is received, Docker pulls the image from the image repository to the machine, and then runs a container based on the image.

Example

In the following example, Docker packages a container image based on the Nginx image, runs an application based on the container image, and pushes the image to the image repository.

Installing Docker

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

In Linux, you can run the following command to install Docker:

```
curl -fsSL get.docker.com -o get-docker.sh
sh get-docker.sh
systemctl restart docker
```

Packaging a Docker Image

Docker provides a convenient way to package your application, which is called Dockerfile.

```
# Use the official Nginx image as the base image.
FROM nginx:alpine

# Run a command to modify the content of the nginx image index.html.
RUN echo "hello world" > /usr/share/nginx/html/index.html

# Permit external access to port 80 of the container.
EXPOSE 80
```

Run the **docker build** command to package the image.

```
docker build -t hello .
```

In the preceding command, **-t** indicates that a tag is added to the image, that is, the image is named. In this example, the image name is **hello**. **.** indicates that the packaging command is executed in the current directory.

Run the **docker images** command to view the image. You can see the hello image has been created successfully. You can also see an Nginx image, which is downloaded from the image repository and used as the base image of the hello image.

```
# docker images
REPOSITORY    TAG       IMAGE ID       CREATED        SIZE
hello         latest   d120ec16dcea  17 minutes ago 158MB
nginx         alpine   eeb27ee6b893  2 months ago  148MB
```

Running the Container Image Locally

After obtaining the image, you can run the **docker run** command on the local host to run the container image.

```
#docker run -p 8080:80 hello
```

The **docker run** command will start a container. In the preceding command, **-p** indicates that port 8080 of the local host is mapped to port 80 of the container. That is, the traffic of port 8080 of the local host will be forwarded to port 80 of

the container. When you access `http://127.0.0.1:8080` on the local host, you can access the container. In this case, the content returned by the browser is **hello world**.

Pushing the Image to the Image Repository

You can push images to SoftWare Repository for Container (SWR). The following describes how to push images to SWR.

Log in to the SWR console. In the navigation pane, choose **My Images**. On the page that is displayed, click **Upload Through Client**. In the dialog box that is displayed, click **Generate a temporary login command**. Then, copy the command and run it on the local host to log in to the SWR image repository.

Before uploading an image, you need to specify a complete name for the image.

```
# docker tag hello registry.eu-west-0.prod-cloud-ocb.orange-business.com/container/hello:v1
```

In the preceding command, **registry.eu-west-0.prod-cloud-ocb.orange-business.com** indicates the repository address. The address varies depending on the region. **v1** indicates the tag allocated to the hello image.

- **registry.eu-west-0.prod-cloud-ocb.orange-business.com** is the repository address. Addresses of each region are different.
- **container** is the organization name. Generally, an organization is created in SWR. If no organization is created, an organization is automatically created when the image is uploaded for the first time. The organization name is globally unique in a single region. You need to select a proper organization name.
- **v1** is the version number allocated to the hello image.

Run the **docker push** command to upload the image to SWR.

```
# docker push registry.eu-west-0.prod-cloud-ocb.orange-business.com/container/hello:v1
```

If you need to use the image, run the **docker pull** command to pull (download) the image.

```
# docker pull registry.eu-west-0.prod-cloud-ocb.orange-business.com/container/hello:v1
```

3.2.2 Kubernetes

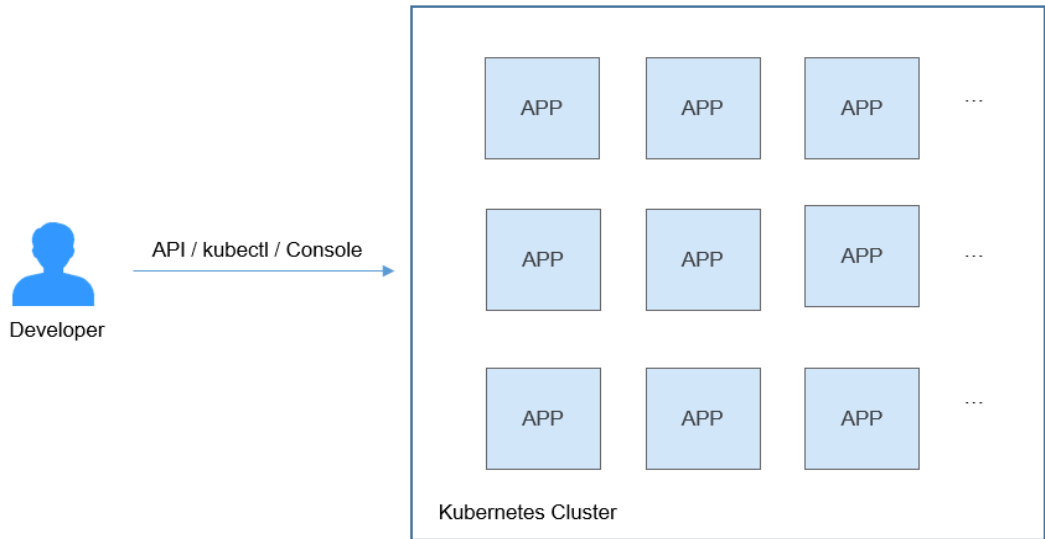
What Is Kubernetes?

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's 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.

Figure 3-3 Running applications in a Kubernetes cluster

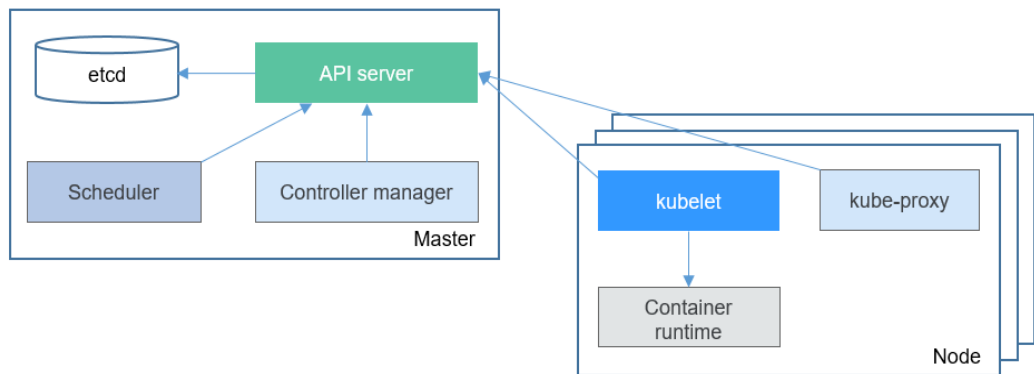


Kubernetes Cluster Architecture

A Kubernetes cluster consists of master nodes (masters) and worker nodes (nodes). Applications are deployed on nodes, and you can deploy applications on specific nodes as configured.

The following figure shows the architecture of a Kubernetes cluster.

Figure 3-4 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: functions as a transit station for components to communicate with each other, receives external requests, and writes information to etcd.
- Controller manager: performs 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: serves as a distributed data storage component that stores cluster configuration information.

In the production environment, multiple master nodes are deployed to ensure cluster high availability. For example, three master nodes will be deployed to create a CCE cluster in HA mode.

Worker node

A worker node is a compute node in a cluster, that is, a node running containerized applications. A worker node has the following components:

- kubelet: communicates with the container runtime, interacts with the API server, and manages containers on the node.
- kube-proxy: serves as an access proxy between application components.
- Container runtime: functions as the software for running containers. You can download images to build your container runtime, such as Docker.

Kubernetes Scalability

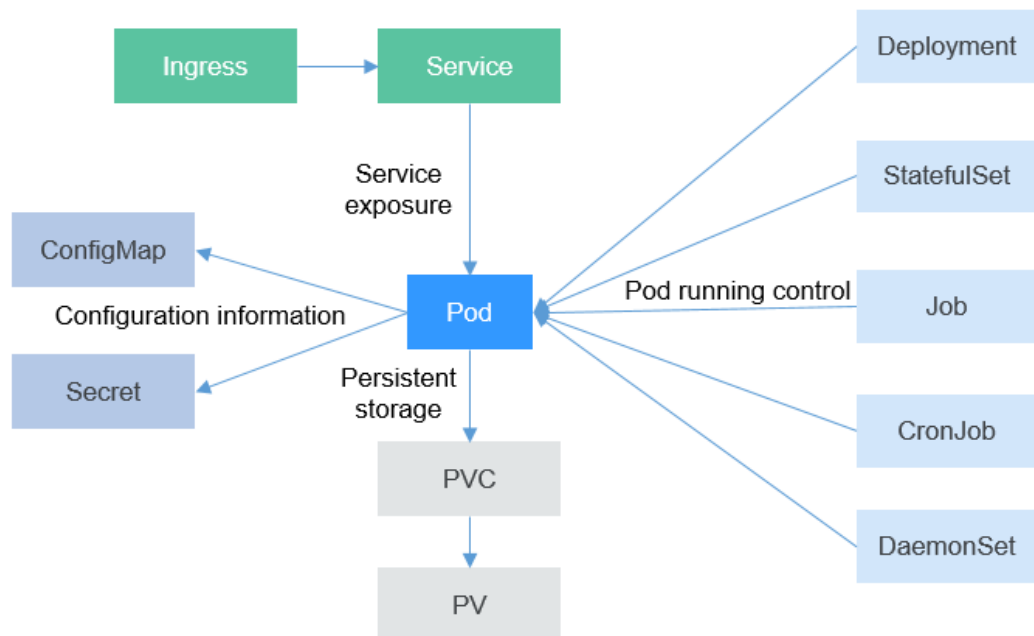
Kubernetes opens the Container Runtime Interface (CRI), Container Network Interface (CNI), and Container Storage Interface (CSI). These interfaces maximize Kubernetes scalability and allow Kubernetes to focus on container scheduling.

- Container Runtime Interface (CRI): provides computing resources when a container is running. It shields differences between container engines and interacts with each container engine through a unified interface.
- Container Network Interface (CNI): enables Kubernetes to support different networking implementations. For example, CCE has developed customized CNI plug-ins that allow your Kubernetes clusters to run in VPCs.
- Container Storage Interface (CSI): enables Kubernetes to support various classes of storage. For example, CCE is interconnected with block storage (EVS), file storage (SFS), and object storage (OBS) services.

Basic Objects in Kubernetes

The following figure describes the basic objects in Kubernetes and the relationships between them.

Figure 3-5 Basic Kubernetes objects



- Pod

A pod is the smallest and simplest unit that you create or deploy in Kubernetes. A pod encapsulates one or more containers, storage resources, a unique network IP address, and options that govern how the containers should run.
- Deployment

A Deployment can be viewed as an application encapsulating pods. It can contain one or more pods. Each pod has the same role, and the system automatically distributes requests to the pods of a Deployment.
- StatefulSet

A StatefulSet is used to manage stateful applications. Like Deployments, StatefulSets manage a group of pods based on an identical container spec. Where they differ is that StatefulSets maintain a fixed ID for each of their pods. These pods are created based on the same declaration but cannot replace each other. Each pod has a permanent ID regardless of how it is scheduled.
- Job

A job is used to control batch tasks. Jobs are different from long-term servo tasks (such as Deployments). The former can be started and terminated at specific time, while the latter runs unceasingly unless it is terminated. Pods managed by a job will be automatically removed after successfully completing tasks based on user configurations.
- Cron job

A cron job is a time-based job. Similar to the crontab of the Linux system, it runs a specified job in a specified time range.
- 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.

- **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.
- **Ingress**
Services forward requests based on Layer 4 TCP and UDP protocols. Ingresses can forward requests based on Layer 7 HTTPS and HTTP protocols and make forwarding more targeted by domain names and paths.
- **ConfigMap**
A ConfigMap stores configuration information in key-value pairs required by applications. With a ConfigMap, you can easily decouple configurations and use different configurations in different environments.
- **Secret**
A secret lets you store and manage sensitive information, such as password, authentication information, certificates, and private keys. Storing confidential information in a secret is safer and more flexible than putting it verbatim in a pod definition or in a container image.
- **PersistentVolume (PV)**
A PV describes a persistent data storage volume. It defines a directory for persistent storage on a host machine, for example, a mount directory of a network file system (NFS).
- **PersistentVolumeClaim (PVC)**
Kubernetes provides PVCs to apply for persistent storage. With PVCs, you only need to specify the type and capacity of storage without concerning about how to create and release underlying storage resources.

Setting Up a Kubernetes Cluster

Kubernetes introduces multiple methods for setting up a Kubernetes cluster, such as minikube and kubeadm.

If you do not want to create a Kubernetes cluster by coding, you can create one on the CCE console. The following sections use clusters created on the CCE console as examples.

kubectl

kubectl is a command line tool for Kubernetes clusters. You can install kubectl on any machine and run kubectl commands to operate your Kubernetes cluster.

For details about how to install kubectl, see [Connecting to a Cluster Through kubectl](#). After connection, you can run the **kubectl cluster-info** command to view the cluster information, as shown below.

```
# kubectl cluster-info
Kubernetes master is running at https://*.*.*:5443
CoreDNS is running at https://*.*.*:5443/api/v1/namespaces/kube-system/services/coredns/dns/proxy
```

To further debug and diagnose cluster problems, use 'kubectl cluster-info dump'.

Run the **kubectl get nodes** command to view information about nodes in the cluster.

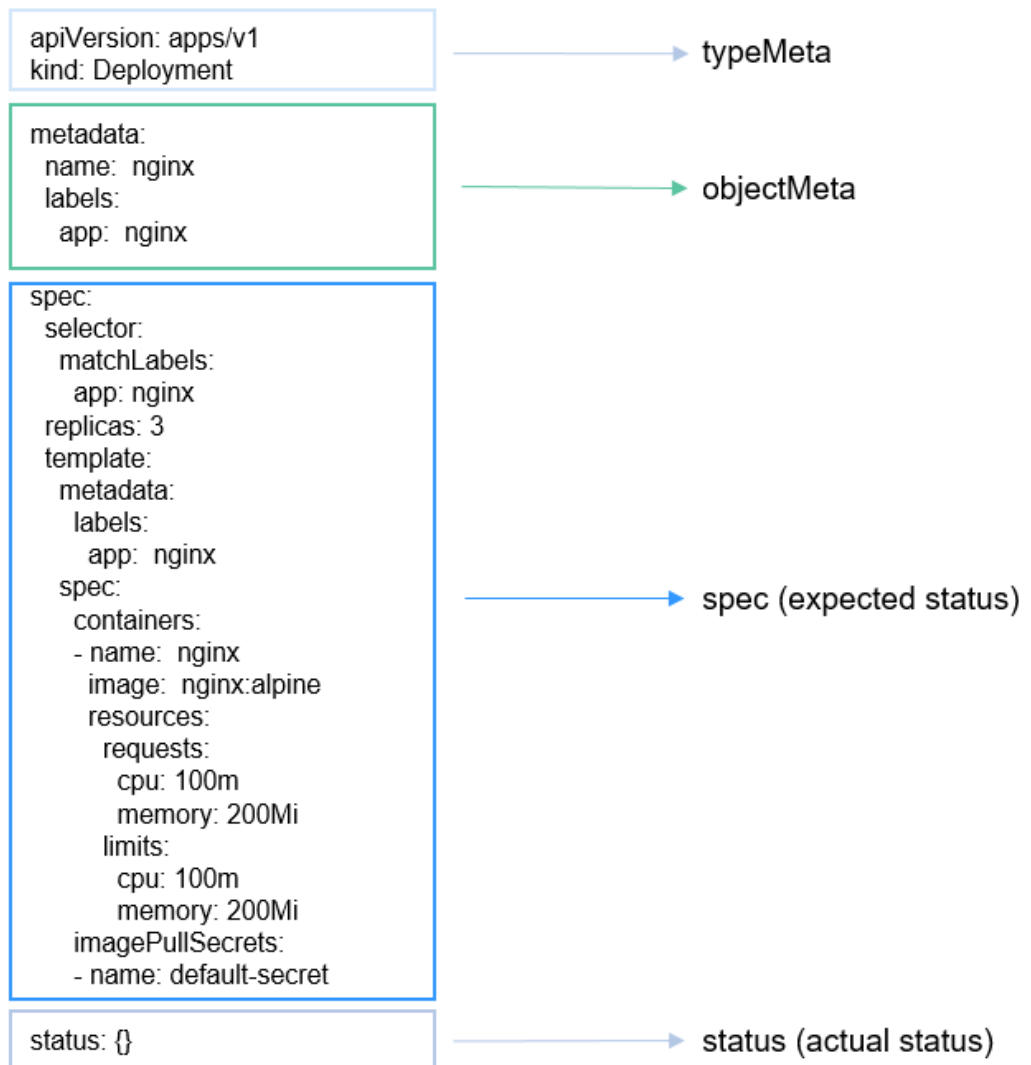
NAME	STATUS	ROLES	AGE	VERSION
192.168.0.153	Ready	<none>	7m	v1.15.6-r1-20.3.0.2.B001-15.30.2
192.168.0.207	Ready	<none>	7m	v1.15.6-r1-20.3.0.2.B001-15.30.2
192.168.0.221	Ready	<none>	7m	v1.15.6-r1-20.3.0.2.B001-15.30.2

Description of Kubernetes Objects

Resources in Kubernetes can be described using a YAML or JSON file. An object description can be divided into the following four parts:

- typeMeta: metadata of the object type, specifying the API version and type of the object.
- objectMeta: metadata about the object, including the object name and used labels.
- spec: expected status of the object, for example, which image the object uses and how many replicas the object has.
- status: actual status of the object, which can be viewed only after the object is created. You do not need to specify the status when creating an object.

Figure 3-6 YAML description file



Running Applications on Kubernetes

Delete **status** from the content in [Figure 3-6](#) and save it as the **nginx-deployment.yaml** file, as shown below:

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx
  labels:
    app: nginx
spec:
  selector:
    matchLabels:
      app: nginx
  replicas: 3
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx:alpine
        resources:
          requests:
            cpu: 100m
            memory: 200Mi
          limits:
            cpu: 100m
            memory: 200Mi
      imagePullSecrets:
      - name: default-secret
```

Use `kubectl` to connect to the cluster and run the following command:

```
# kubectl create -f nginx-deployment.yaml
deployment.apps/nginx created
```

After the command is executed, three pods are created in the Kubernetes cluster. You can run the following command to query the Deployment and pods:

```
# kubectl get deploy
NAME    READY  UP-TO-DATE  AVAILABLE  AGE
nginx   3/3    3           3          9s

# kubectl get pods
NAME                                READY  STATUS   RESTARTS  AGE
nginx-685898579b-qrt4d             1/1    Running  0         15s
nginx-685898579b-t9zd2             1/1    Running  0         15s
nginx-685898579b-w59jn             1/1    Running  0         15s
```

By now, we have walked you through the Kubernetes basics of containers and clusters, and provided you an example of how to use `kubectl`. The following sections will go deeper into Kubernetes objects, such as how they are used and related.

3.3 Pod, Label, and Namespace

3.3.1 Pod: the Smallest Scheduling Unit in Kubernetes

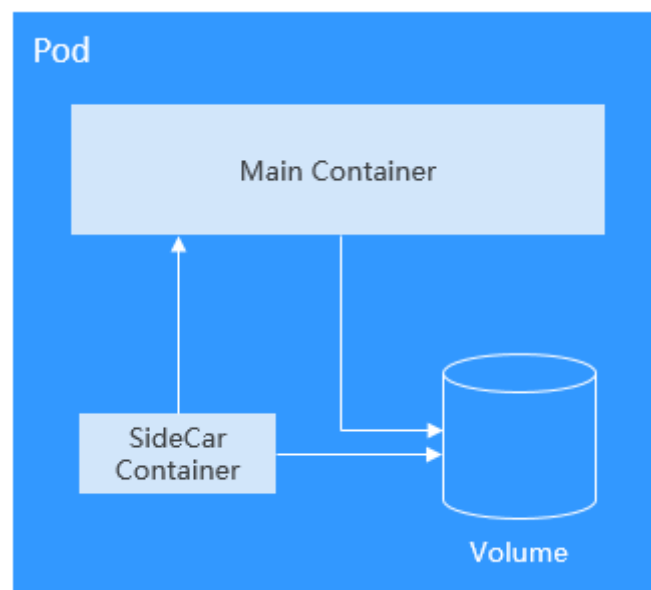
Pod

A pod is the smallest and simplest unit in the Kubernetes object model that you create or deploy. A pod encapsulates one or more containers, storage volumes, a unique network IP address, and options that govern how the containers should run.

Pods can be used in either of the following ways:

- A container is running in a pod. This is the most common usage of pods in Kubernetes. You can view the pod as a single encapsulated container, but Kubernetes directly manages pods instead of containers.
- Multiple containers that need to be coupled and share resources run in a pod. In this scenario, an application contains a main container and several sidecar containers, as shown in [Figure 3-7](#). For example, the main container is a web server that provides file services from a fixed directory, and a sidecar container periodically downloads files to the directory.

Figure 3-7 Pod



In Kubernetes, pods are rarely created directly. Instead, controllers such as Deployments and jobs, are used to manage pods. Controllers can create and manage multiple pods, and provide replica management, rolling upgrade, and self-healing capabilities. A controller generally uses a pod template to create corresponding pods.

Creating a Pod

Kubernetes resources can be described using YAML or JSON files. The following example describes a pod named **nginx**. This pod contains a container named **container-0** and uses the **nginx:alpine** image, 100m CPU, and 200 MiB memory.

```
apiVersion: v1          # Kubernetes API version
kind: Pod               # Kubernetes resource type
```

```
metadata:
  name: nginx          # Pod name
spec:                 # Pod specifications
  containers:
  - image: nginx:alpine # The image used is nginx:alpine.
    name: container-0  # Container name
    resources:         # Resources required for a container
      limits:
        cpu: 100m
        memory: 200Mi
      requests:
        cpu: 100m
        memory: 200Mi
    imagePullSecrets:  # Secret used to pull the image, which must be default-secret on CCE
  - name: default-secret
```

As shown in the annotation of YAML, the YAML description file includes:

- **metadata**: information such as name, label, and namespace
- **spec**: pod specification such as image and volume used

If you query a Kubernetes resource, you can see the **status** field. This field indicates the status of the Kubernetes resource, and does not need to be set when the resource is created. This example is a minimum set. Other parameter definition will be described later.

After the pod is defined, you can create it using `kubectl`. Assume that the preceding YAML file is named **nginx.yaml**, run the following command to create the file. **-f** indicates that it is created in the form of a file.

```
$ kubectl create -f nginx.yaml
pod/nginx created
```

After the pod is created, you can run the **kubectl get pods** command to query the pod information, as shown below.

```
$ kubectl get pods
NAME      READY STATUS  RESTARTS  AGE
nginx    1/1   Running  0         40s
```

The preceding information indicates that the **nginx** pod is in the **Running** state, indicating that the pod is running. **READY** is **1/1**, indicating that there is one container in the pod, and the container is in the **Ready** state.

You can run the **kubectl get** command to query the configuration information about a pod. In the following command, **-o yaml** indicates that the pod is returned in YAML format. **-o json** indicates that the pod is returned in JSON format.

```
$ kubectl get pod nginx -o yaml
```

You can also run the **kubectl describe** command to view the pod details.

```
$ kubectl describe pod nginx
```

When a pod is deleted, Kubernetes stops all containers in the pod. Kubernetes sends the SIGTERM signal to the process and waits for a period (30 seconds by default) to stop the container. If it is not stopped within the period, Kubernetes sends a SIGKILL signal to kill the process.

You can stop and delete a pod in multiple methods. For example, you can delete a pod by name, as shown below.

```
$ kubectl delete po nginx
pod "nginx" deleted
```

Delete multiple pods at one time.

```
$ kubectl delete po pod1 pod2
```

Delete all pods.

```
$ kubectl delete po --all
pod "nginx" deleted
```

Delete pods by labels. For details about labels, see [Label for Managing Pods](#).

```
$ kubectl delete po -l app=nginx
pod "nginx" deleted
```

Environment Variables

Environment variables are set in the container running environment.

Environment variables add flexibility to workload configuration. The environment variables for which you have assigned values during container creation will take effect when the container is running. This saves you the trouble of rebuilding the container image.

The following shows how to use an environment variable. You only need to configure the `spec.containers.env` field.

```
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
    env:
      - name: env_key          # Environment variable
        value: env_value
  imagePullSecrets:
  - name: default-secret
```

Run the following command to check the environment variables in the container. The value of the `env_key` environment variable is `env_value`.

```
$ kubectl exec -it nginx -- env
PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
HOSTNAME=nginx
TERM=xterm
env_key=env_value
```

Environment variables can also reference **ConfigMap** and **secret**. For details, see [Referencing a ConfigMap as an Environment Variable](#) and [Referencing a Secret as an Environment Variable](#).

Setting Container Startup Commands

Starting a container is to start the main process. Some preparations must be made before the main process is started. For example, you may configure or initialize

MySQL databases before running MySQL servers. You can set **ENTRYPOINT** or **CMD** in the Dockerfile when creating an image. As shown in the following example, the **ENTRYPOINT ["top", "-b"]** command is set in the Dockerfile. This command will be executed during container startup.

```
FROM ubuntu
ENTRYPOINT ["top", "-b"]
```

When calling an API, you only need to configure the **containers.command** field of the pod. This field is of the list type. The first parameter in the field is the command to be executed, and the subsequent parameters are the command arguments.

```
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
    command:
      - top
      - "-b"
  imagePullSecrets:
  - name: default-secret
```

Container Lifecycle

Kubernetes provides **container lifecycle hooks**. The hooks enable containers to run code triggered by events during their management lifecycle. For example, if you want a container to perform a certain operation before it is stopped, you can register a hook. The following lifecycle hooks are provided:

- **postStart**: triggered immediately after the workload is started
- **preStop**: triggered immediately before the workload is stopped

You only need to set the **lifecycle.postStart** or **lifecycle.preStop** parameter of the pod, as shown in the following:

```
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
    lifecycle:
      postStart:
        exec:
          command:
            - top
            - "-b"
```

```

command:
- "/postStart.sh"
preStop:      # Pre-stop processing
exec:
command:
- "/preStop.sh"
imagePullSecrets:
- name: default-secret

```

3.3.2 Liveness Probe

Overview

Kubernetes applications have the self-healing capability, that is, when an application container crashes, the container can be detected and restarted automatically. However, this mechanism does not work for deadlocks. Assume that a Java program is having a memory leak. The program is unable to make any progress, while the JVM process is running. To address this issue, Kubernetes introduces liveness probes to check whether containers response normally and determine whether to restart containers. This is a good health check mechanism.

It is advised to define the liveness probe for every pod to gain a better understanding of pods' running statuses.

Supported detection mechanisms are as follows:

- **HTTP GET:** The kubelet sends an HTTP GET request to the container. Any 2XX or 3XX code indicates success. Any other code returned indicates failure.
- **TCP Socket:** The kubelet attempts to open a socket to your container on the specified port. If it can establish a connection, the container is considered healthy. If it fails to establish a connection, the container is considered a failure.
- **Exec:** kubelet executes a command in the target container. If the command succeeds, it returns **0**, and kubelet considers the container to be alive and healthy. If the command returns a non-zero value, kubelet kills the container and restarts it.

In addition to liveness probes, readiness probes are also available for you to detect pod status. For details, see [Readiness Probe](#).

HTTP GET

HTTP GET is the most common detection method. An HTTP GET request is sent to a container. Any 2xx or 3xx code returned indicates that the container is healthy. The following example shows how to define such a request:

```

apiVersion: v1
kind: Pod
metadata:
  name: liveness-http
spec:
  containers:
  - name: liveness
    image: nginx:alpine
    livenessProbe:      # liveness probe
      httpGet:          #HTTP GET definition
        path: /
        port: 80
    imagePullSecrets:
    - name: default-secret

```

Create pod **liveness-http**.

```
$ kubectl create -f liveness-http.yaml
pod/liveness-http created
```

The probe sends an HTTP Get request to port 80 of the container. If the request fails, Kubernetes restarts the container.

View details of pod **liveness-http**.

```
$ kubectl describe po liveness-http
Name:          liveness-http
.....
Containers:
  liveness:
    .....
    State:      Running
      Started:   Mon, 03 Aug 2020 03:08:55 +0000
    Ready:      True
    Restart Count: 0
    Liveness:    http-get http://:80/ delay=0s timeout=1s period=10s #success=1 #failure=3
    Environment: <none>
    Mounts:
      /var/run/secrets/kubernetes.io/serviceaccount from default-token-vssmw (ro)
    .....
.....
```

The preceding output reports that the pod is **Running** with **Restart Count** being **0**, which indicates that the container is normal and no restarts have been triggered. If the value of **Restart Count** is not **0**, the container has been restarted.

TCP Socket

TCP Socket: The kubelet attempts to open a socket to your container on the specified port. If it can establish a connection, the container is considered healthy. If it fails to establish a connection, the container is considered a failure. For detailed defining method, see the following example.

```
apiVersion: v1
kind: Pod
metadata:
  labels:
    test: liveness
  name: liveness-tcp
spec:
  containers:
  - name: liveness
    image: nginx:alpine
    livenessProbe: # liveness probe
      tcpSocket:
        port: 80
    imagePullSecrets:
    - name: default-secret
```

Exec

kubelet executes a command in the target container. If the command succeeds, it returns **0**, and kubelet considers the container to be alive and healthy. The following example shows how to define the command.

```
apiVersion: v1
kind: Pod
metadata:
  labels:
    test: liveness
  name: liveness-exec
```

```
spec:
  containers:
  - name: liveness
    image: nginx:alpine
    args:
    - /bin/sh
    - -c
    - touch /tmp/healthy; sleep 30; rm -rf /tmp/healthy; sleep 600
    livenessProbe:
      # liveness probe
      exec:
        # Exec definition
        command:
        - cat
        - /tmp/healthy
    imagePullSecrets:
    - name: default-secret
```

In the preceding configuration file, kubelet executes the command **cat /tmp/healthy** in the container. If the command succeeds and returns **0**, the container is considered healthy. For the first 30 seconds, there is a **/tmp/healthy** file. So during the first 30 seconds, the command **cat /tmp/healthy** returns a success code. After 30 seconds, the **/tmp/healthy** file is deleted. The probe will then consider the pod to be unhealthy and restart it.

Advanced Settings of a Liveness Probe

The **describe** command of **liveness-http** returns the following information:

```
Liveness: http-get http://:80/ delay=0s timeout=1s period=10s #success=1 #failure=3
```

This is the detailed configuration of the liveness probe.

- **delay=0s** indicates that the probe starts immediately after the container is started.
- **timeout=1** indicates that the container must respond within one second. Otherwise, the health check is recorded as failed.
- **period=10s** indicates that the probe checks containers every 10 seconds.
- **#success=1** indicates that the operation is recorded as successful if it is successful for once.
- **#failure=3** indicates that a container will be restarted after three consecutive failures.

The preceding liveness probe indicates that the probe checks containers immediately after they are started. If a container does not respond within one second, the check is recorded as failed. The health check is performed every 10 seconds. If the check fails for three consecutive times, the container is restarted.

These are the default configurations when the probe is created. You can customize them as follows:

```
apiVersion: v1
kind: Pod
metadata:
  name: liveness-http
spec:
  containers:
  - name: liveness
    image: nginx:alpine
    livenessProbe:
      httpGet:
        path: /
        port: 80
      initialDelaySeconds: 10 # Liveness probes are initiated after the container has started for 10s.
```



```
timeoutSeconds: 2    # The container must respond within 2s. Otherwise, it is considered as a
failure.
periodSeconds: 30   # The probe is performed every 30s.
successThreshold: 1 # The container is considered healthy as long as the probe succeeds once.
failureThreshold: 3 # The container is considered unhealthy after three consecutive failures.
```

Normally, the value of **initialDelaySeconds** must be greater than **0**, because it takes a while for the application to be ready. The probe often fails if the probe is initiated before the application is ready.

In addition, you can set the value of **failureThreshold** to be greater than **1**. In this way, the kubelet checks the container for multiple times in one probe rather than performing the probe for multiple times.

Configuring a Liveness Probe

- **What to check**

An effective liveness probe should check all the key parts of an application and use a dedicated URL, such as **/health**. When the URL is accessed, the probe is triggered and a result is returned. Note that no authentication should be involved. Otherwise, the probe keeps failing and restarting the container.

In addition, a probe must not check parts that have external dependencies. For example, if a frontend web server cannot connect to a database, the web server should not be considered unhealthy for the connection failure.

- **To be lightweight**

A liveness probe must not occupy too many resources or certain resources for too long. Otherwise, resource shortage may affect service running. For example, the HTTP GET method is recommended for a Java application. If the Exec method is used, the JVM startup process occupies too many resources.

3.3.3 Label for Managing Pods

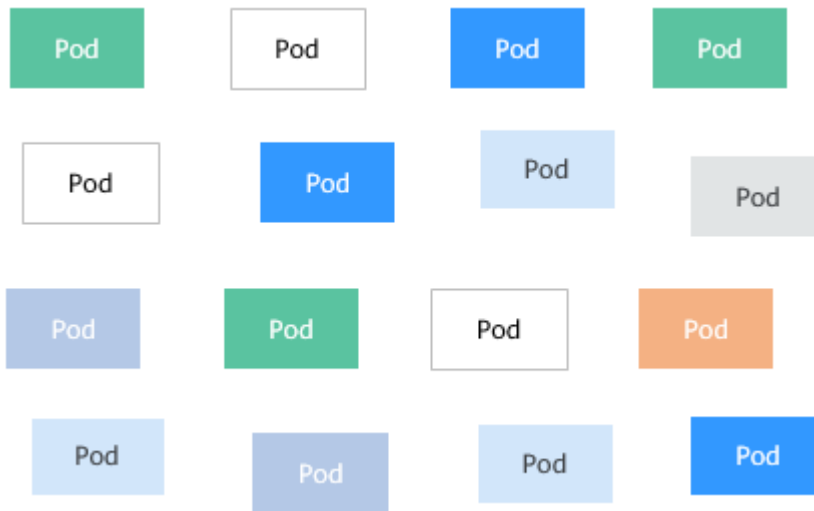
Why We Need Labels

As resources increase, how to classify and manage resources becomes important. Kubernetes provides a mechanism to classify resources, that is, using labels. Labels are simple but powerful. Almost all resources in the Kubernetes can be organized by labels.

A label is a key-value pair. It can be set either during or after resource creation. You can easily modify it when needed at any time.

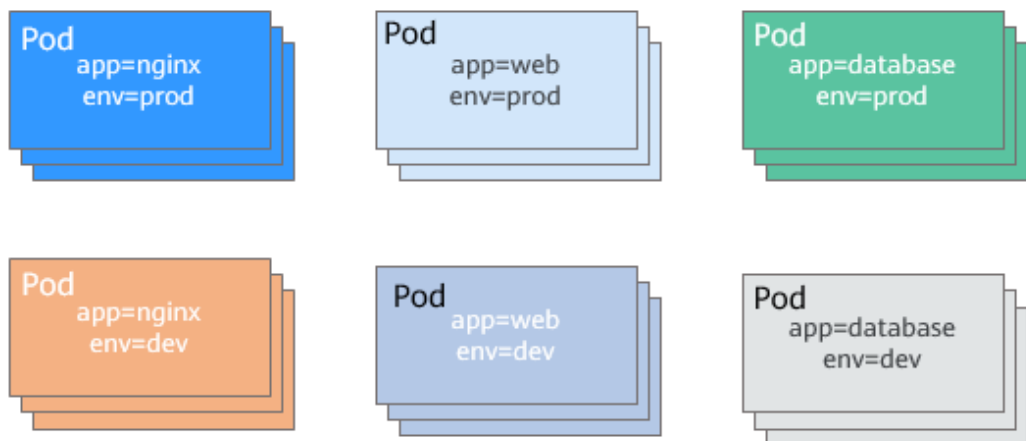
The following figures show how labels work. Assume that you have multiple pods of various kinds. It could be challenging when you manage them.

Figure 3-8 Pods without classification



After we add labels to them. It is much clearer.

Figure 3-9 Pods classified using labels



Adding a Label

The following example shows how to add labels when you are creating a pod.

```
apiVersion: v1
kind: Pod
metadata:
  name: nginx
  labels:           # Add labels app=nginx and env=prod to the pod.
    app: nginx
    env: prod
spec:
  containers:
  - image: nginx:alpine
    name: container-0
  resources:
    limits:
      cpu: 100m
      memory: 200Mi
```

```
requests:
  cpu: 100m
  memory: 200Mi
imagePullSecrets:
- name: default-secret
```

After you add labels to a pod, you can view the labels by adding **--show-labels** when querying the pod.

```
$ kubectl get pod --show-labels
NAME          READY STATUS  RESTARTS  AGE  LABELS
nginx         1/1   Running  0         50s  app=nginx,env=prod
```

You can also use **-L** to query only certain labels.

```
$ kubectl get pod -L app,env
NAME          READY STATUS  RESTARTS  AGE  APP  ENV
nginx         1/1   Running  0         1m  nginx  prod
```

For an existing pod, you can run the **kubectl label** command to add labels.

```
$ kubectl label pod nginx creation_method=manual
pod/nginx labeled

$ kubectl get pod --show-labels
NAME          READY STATUS  RESTARTS  AGE  LABELS
nginx         1/1   Running  0         50s  app=nginx, creation_method=manual,env=prod
```

Modifying a Label

Add **--overwrite** to the command to modify a label.

```
$ kubectl label pod nginx env=debug --overwrite
pod/nginx labeled

$ kubectl get pod --show-labels
NAME          READY STATUS  RESTARTS  AGE  LABELS
nginx         1/1   Running  0         50s  app=nginx,creation_method=manual,env=debug
```

3.3.4 Namespaces: Grouping Resources

Why We Need Namespaces

Although labels are simple and efficient, too many labels can cause chaos and make querying inconvenient. Labels can overlap with each other, which is not suitable for certain scenarios. This is where namespace comes in. Namespaces allow you to isolate and manage resources in a more systematic way. Multiple namespaces can divide systems that contain multiple components into different non-overlapped groups. Namespaces also enable you to divide cluster resources between users. In this way, multiple teams can share one cluster.

Resources of the same type can share the same name as long as they are in different namespaces. Unlike most resources in Kubernetes can be managed by namespace, global resources do not belong to a specific namespace. Later sections will discuss this topic in detail.

Run the following command to query namespaces in the current cluster:

```
$ kubectl get ns
NAME          STATUS AGE
default       Active 36m
kube-node-release Active 36m
kube-public   Active 36m
kube-system   Active 36m
```

By now, we are performing operations in the default namespace. When **kubectl get** is used but no namespace is specified, the default namespace is used by default.

You can run the following command to view resources in namespace **kube-system**.

```
$ 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
everest-csi-controller-6d796fb9c5-v22df 2/2   Running 0      9m11s
everest-csi-driver-snzrr             1/1   Running 0      12m
everest-csi-driver-ttj28             1/1   Running 0      12m
everest-csi-driver-wtrk6            1/1   Running 0      12m
icagent-2kz8g                       1/1   Running 0      12m
icagent-hjz4h                       1/1   Running 0      12m
icagent-m4bbl                       1/1   Running 0      12m
```

You can see that there are many pods in **kube-system**. **coredns** is used for service discovery, **everest-csi** for connecting to storage services, and **icagent** for connecting to the monitoring system.

These general, must-have applications are put in the **kube-system** namespace to isolate them from other pods. They are invisible to and free from being affected by resources in other namespaces.

Creating 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
```

Create resources in the namespace.

```
$ kubectl create -f nginx.yaml -n custom-namespace
pod/nginx created
```

By now, namespaces **default** and **custom-namespace** both have a pod named **nginx**.

The Isolation function of Namespaces

Namespaces are used to group resources only for organization purposes. Running objects in different namespaces are not essentially isolated. For example, if pods in two namespaces know the IP address of each other and the underlying network on which Kubernetes depends does not provide network isolation between namespaces, the two pods can access each other.

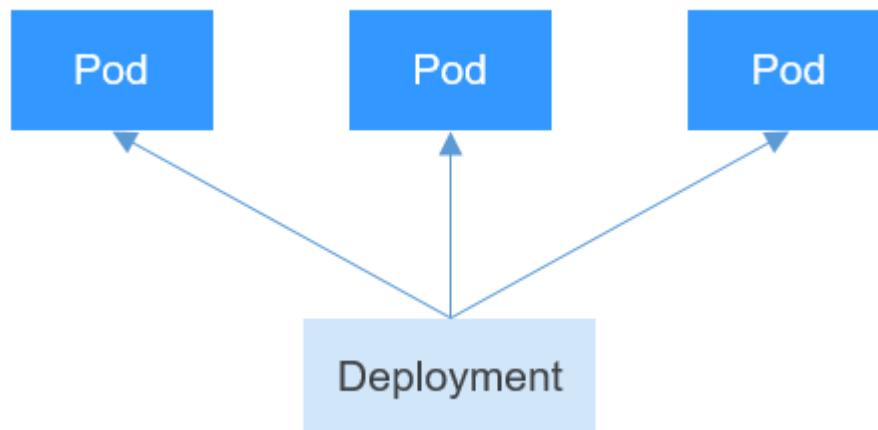
3.4 Pod Orchestration and Scheduling

3.4.1 Deployment

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 3-10 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.

Creating a Deployment

In the following example, a Deployment named **nginx** is created, and two pods are created from the **nginx:latest** image. Each pod occupies 100m CPU and 200 MiB memory.

```
apiVersion: apps/v1 # Note the difference with a pod. It is apps/v1 instead of v1 for a Deployment.
kind: Deployment # The resource type is Deployment.
metadata:
  name: nginx # Name of the Deployment
spec:
  replicas: 2 # Number of pods. The Deployment ensures that two pods are running.
  selector: # Label Selector
    matchLabels:
```

```

app: nginx
template: # Definition of a pod, which is used to create pods. It is also known as pod 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

```

In this definition, the name of the Deployment is **nginx**, and **spec.replicas** defines the number of pods. That is, the Deployment controls two pods. **spec.selector** is a label selector, indicating that the Deployment selects the pod whose label is **app=nginx**. **spec.template** is the definition of the pod and is the same as that defined in [Pods](#).

Save the definition of the Deployment to **deployment.yaml** and use `kubectl` to create the Deployment.

Run `kubectl get` to view the Deployment and pods. In the following example, the value of **READY** is **2/2**. The first **2** indicates that two pods are running, and the second **2** indicates that two pods are expected in this Deployment. The value **2** of **AVAILABLE** indicates that two pods are available.

```

$ kubectl create -f deployment.yaml
deployment.apps/nginx created

$ kubectl get deploy
NAME      READY   UP-TO-DATE   AVAILABLE   AGE
nginx     2/2     2            2           4m5s

```

How Does the Deployment Control Pods?

Continue to query pods, as shown below.

```

$ kubectl get pods
NAME                                READY   STATUS    RESTARTS   AGE
nginx-7f98958cdf-tdmqk             1/1     Running   0          13s
nginx-7f98958cdf-txckx             1/1     Running   0          13s

```

If you delete a pod, a new pod is immediately created, as shown below. As mentioned above, the Deployment ensures that there are two pods running. If a pod is deleted, the Deployment creates a new pod. If a pod becomes faulty, the Deployment automatically restarts the pod.

```

$ kubectl get pods
NAME                                READY   STATUS    RESTARTS   AGE
nginx-7f98958cdf-tdmqk             1/1     Running   0          21s
nginx-7f98958cdf-tesqr             1/1     Running   0          21s

```

You see two pods, **nginx-7f98958cdf-tdmqk** and **nginx-7f98958cdf-tesqr**. **nginx** is the name of the Deployment. **-7f98958cdf-tdmqk** and **-7f98958cdf-tesqr** are the suffixes randomly generated by Kubernetes.

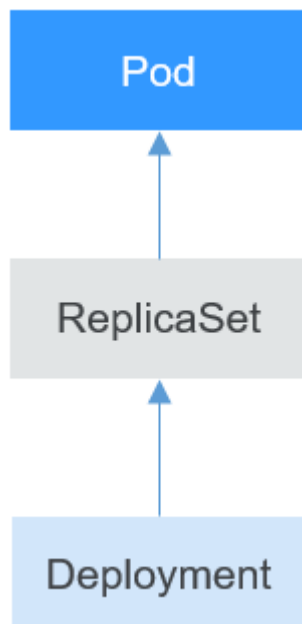
You may notice that the two suffixes share the same content **7f98958cdf** in the first part. This is because the Deployment does not control the pods directly, but through a controller named ReplicaSet. You can run the following command to query the ReplicaSet. In the command, **rs** is the abbreviation of ReplicaSet.

```
$ kubectl get rs
NAME          DESIRED  CURRENT  READY  AGE
nginx-7f98958cdf  2        2        2      1m
```

The ReplicaSet is named **nginx-7f98958cdf**, in which the suffix **-7f98958cdf** is generated randomly.

As shown in **Figure 3-11**, the Deployment controls the ReplicaSet, which then controls pods.

Figure 3-11 How does the Deployment control the pod



If you run the **kubectl describe** command to view the details of the Deployment, you can see the ReplicaSet (**NewReplicaSet: nginx-7f98958cdf (2/2 replicas created)**). In **Events**, the number of pods of the ReplicaSet is scaled out to 2. In practice, you may not operate ReplicaSet directly, but understanding that a Deployment controls a pod by controlling a ReplicaSet helps you locate problems.

```
$ kubectl describe deploy nginx
Name:          nginx
Namespace:    default
CreationTimestamp:  Sun, 16 Dec 2018 19:21:58 +0800
Labels:       app=nginx
...

NewReplicaSet: nginx-7f98958cdf (2/2 replicas created)
Events:
  Type      Reason          Age    From          Message
  ----      -
  Normal    ScalingReplicaSet  5m    deployment-controller  Scaled up replica set nginx-7f98958cdf to 2
```

Upgrade

In actual applications, upgrade is a common operation. A Deployment can easily support application upgrade.

You can set different upgrade policies for a Deployment:

- **RollingUpdate:** New pods are created gradually and then old pods are deleted. This is the default policy.
- **Recreate:** The current pods are deleted and then new pods are created.

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         21s
nginx-6f9f58dff-tesqr  1/1    Running  0         21s
```

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
```

- **maxSurge** specifies the maximum number of pods that can exist over **spec.replicas** in the Deployment. The default value is 25%. For example, if **spec.replicas** is set to **4**, no more than 5 pods can exist during the upgrade process, that is, the upgrade step is 1. The absolute number is calculated from the percentage by rounding up. The value can also be set to an absolute number.
- **maxUnavailable:** specifies the maximum number of pods that can be unavailable during the update process. The default value is 25%. For example, if **spec.replicas** is set to **4**, at least 3 pods exist during the upgrade process, that is, the deletion step is 1. The value can also be set to an absolute number.

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 the earlier version when a fault occurs during the 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.

3.4.2 StatefulSet

StatefulSet

All pods under a Deployment have the same characteristics except for the name and IP address. If required, a Deployment can use the pod template to create a new pod. If not required, the Deployment can delete any one of the pods.

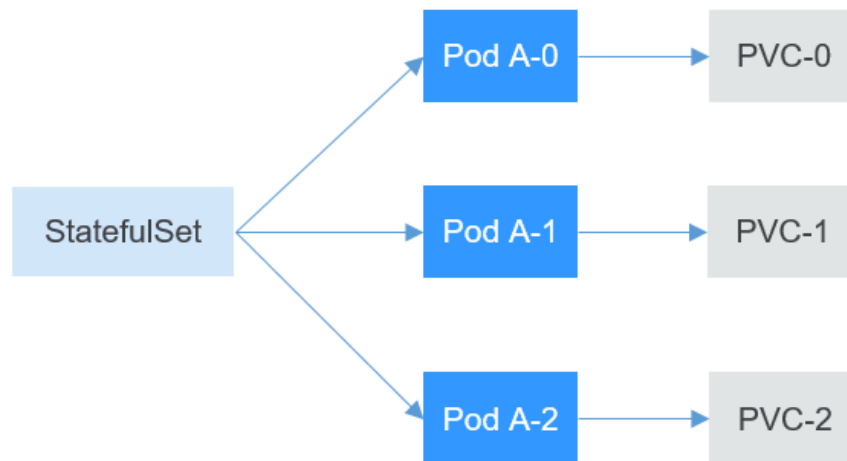
However, Deployments cannot meet the requirements in some distributed scenarios when each pod requires its own status or in a distributed database where each pod requires independent storage.

With detailed analysis, it is found that each part of distributed stateful applications plays a different role. For example, the database nodes are deployed in active/standby mode, and pods are dependent on each other. In this case, you need to meet the following requirements for the pods:

- A pod can be recognized by other pods. Therefore, a pod must have a fixed identifier.
- Each pod has an independent storage device. After a pod is deleted and then restored, the data read from the pod must be the same as the previous one. Otherwise, the pod status is inconsistent.

To address the preceding requirements, Kubernetes provides StatefulSets.

1. A StatefulSet provides 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 host name remain unchanged.
2. A StatefulSet provides a fixed access domain name for each pod through the headless Service (described in following sections).
3. The StatefulSet creates PersistentVolumeClaims (PVCs) with fixed identifiers to ensure that pods can access the same persistent data after being rescheduled.



The following describes how to create a StatefulSet and experience its features.

Creating a Headless Service

As described above, a headless Service is required for pod access when a StatefulSet is created. For details about the Service, see [Services](#). The following describes how to create a headless Service.

Use the following file to describe the headless Service:

- **spec.clusterIP:** Set it to **None**, which indicates a headless Service is to be created.
- **spec.ports.port:** indicates the number of the port used for communication between pods.
- **spec.ports.name:** indicates the name of the port used for communication between pods.

```

apiVersion: v1
kind: Service      # Object type (Service)
metadata:
  name: nginx
  labels:
    app: nginx
spec:
  ports:
    - name: nginx  # Name of the port for communication between pods
      port: 80    # Port number for communication between pods
  selector:
    app: nginx    # Select the pod whose label is app:nginx.
  clusterIP: None # Set this parameter to None, indicating the headless Service.
  
```

Run the following command to create a headless Service:

```

# kubectl create -f headless.yaml
service/nginx created
  
```

After the Service is created, you can query the Service information.

```

# kubectl get svc
NAME      TYPE      CLUSTER-IP  EXTERNAL-IP  PORT(S)  AGE
nginx     ClusterIP None         <none>       80/TCP   5s
  
```

Creating a StatefulSet

The YAML definition of StatefulSets is basically the same as that of other objects. The differences are as follows:

- **serviceName** specifies the headless Service used by the StatefulSet. You need to specify the name of the headless Service.
- **volumeClaimTemplates** is used to apply for a **PVC**. A template named **data** is defined, which will create a PVC for each pod. **storageClassName** specifies the persistent storage class. For details, see [PersistentVolumes, PersistentVolumeClaims, and StorageClasses](#). **volumeMounts** is used to mount storage to pods. If no storage is required, you can delete the **volumeClaimTemplates** and **volumeMounts** fields.

```
apiVersion: apps/v1
kind: StatefulSet
metadata:
  name: nginx
spec:
  serviceName: nginx           # Name of the headless Service
  replicas: 3
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
        - name: container-0
          image: nginx:alpine
          resources:
            limits:
              cpu: 100m
              memory: 200Mi
            requests:
              cpu: 100m
              memory: 200Mi
          volumeMounts:         # Storage mounted to the pod
            - name: data
              mountPath: /usr/share/nginx/html # Mount the storage to /usr/share/nginx/html.
      imagePullSecrets:
        - name: default-secret
      volumeClaimTemplates:
        - metadata:
            name: data
          spec:
            accessModes:
              - ReadWriteMany
            resources:
              requests:
                storage: 1Gi
            storageClassName: csi-nas          # Persistent storage class
```

Run the following command to create a StatefulSet:

```
# kubectl create -f statefulset.yaml
statefulset.apps/nginx created
```

After the command is executed, query the StatefulSet and pods. The suffix of the pod names starts from 0 and increases to 2.

```
# kubectl get statefulset
NAME READY AGE
nginx 3/3 107s
```

```
# kubectl get pods
NAME     READY   STATUS    RESTARTS   AGE
nginx-0  1/1     Running   0           112s
nginx-1  1/1     Running   0           69s
nginx-2  1/1     Running   0           39s
```

In this case, if you manually delete the **nginx-1** pod and query the pods again, you can see that a pod with the same name is created. According to **5s** under **AGE**, it is found that the **nginx-1** pod is newly created.

```
# kubectl delete pod nginx-1
pod "nginx-1" deleted

# kubectl get pods
NAME     READY   STATUS    RESTARTS   AGE
nginx-0  1/1     Running   0           3m4s
nginx-1  1/1     Running   0           5s
nginx-2  1/1     Running   0           1m10s
```

Access the container and check its host names. The host names are **nginx-0**, **nginx-1**, and **nginx-2**.

```
# kubectl exec nginx-0 -- sh -c 'hostname'
nginx-0
# kubectl exec nginx-1 -- sh -c 'hostname'
nginx-1
# kubectl exec nginx-2 -- sh -c 'hostname'
nginx-2
```

In addition, you can view the PVCs created by the StatefulSet. These PVCs are named in the format of **PVC name-StatefulSet name-No.** and are in the **Bound** state.

```
# kubectl get pvc
NAME          STATUS   VOLUME                                     CAPACITY   ACCESS MODES   STORAGECLASS   AGE
data-nginx-0  Bound   pvc-f58bc1a9-6a52-4664-a587-a9a1c904ba29  1Gi        RWX             csi-nas        2m24s
data-nginx-1  Bound   pvc-066e3a3a-fd65-4e65-87cd-6c3fd0ae6485  1Gi        RWX             csi-nas        101s
data-nginx-2  Bound   pvc-a18cf1ce-708b-4e94-af83-766007250b0c  1Gi        RWX             csi-nas        71s
```

Network Identifier of a StatefulSet

After a StatefulSet is created, you can see that each pod has a fixed name. The headless Service provides a fixed domain name for pods by using DNS. In this way, pods can be accessed using the domain name. Even if the IP address of the pod changes when the pod is re-created, the domain name remains unchanged.

After a headless Service is created, the IP address of each pod corresponds to a domain name in the following format:

<pod-name>.<svc-name>.<namespace>.svc.cluster.local

For example, the domain names of the three pods are as follows:

- nginx-0.nginx.default.svc.cluster.local
- nginx-1.nginx.default.svc.cluster.local
- nginx-2.nginx.default.svc.cluster.local

In actual access, **.<namespace>.svc.cluster.local** can be omitted.

Create a pod from the **tutum/dnsutils** image. Then, access the container of the pod and run the **nslookup** command to view the domain name of the pod. The IP

address of the pod can be parsed. The IP address of the DNS server is **10.247.3.10**. When a CCE cluster is created, the `coredns` add-on is installed by default to provide the DNS service. The functions of `coredns` will be described in [Kubernetes Networking](#).

```
$ kubectl run -i --tty --image tutum/dnsutils dnsutils --restart=Never --rm /bin/sh
If you don't 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

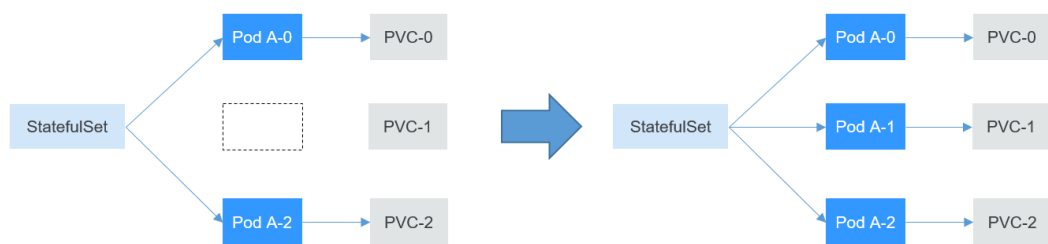
/# nslookup nginx-2.nginx
Server:      10.247.3.10
Address:    10.247.3.10#53
Name:      nginx-2.nginx.default.svc.cluster.local
Address: 172.16.0.19
```

In this case, if you manually delete the two pods, query the IP addresses of the pods re-created by the StatefulSet, and run the `nslookup` command to resolve the domain names of the pods, you can still get `nginx-0.nginx` and `nginx-1.nginx`. This ensures that the network identifier of the StatefulSet remains unchanged.

StatefulSet Storage Status

As mentioned above, StatefulSets can use PVCs for persistent storage to ensure that the same persistent data can be accessed after pods are rescheduled. When pods are deleted, PVCs are not deleted.

Figure 3-12 Process for a StatefulSet to re-create a pod



After the Pod A-1 is deleted and recreated, the PVC-1 is rebound to the Pod A-1.

Run the following command to write some data into the `/usr/share/nginx/html` directory of `nginx-1`. For example, change the content of `index.html` to **hello world**.

```
# kubectl exec nginx-1 -- sh -c 'echo hello world > /usr/share/nginx/html/index.html'
```

After the modification, if you access `https://localhost`, **hello world** is returned.

```
# kubectl exec -it nginx-1 -- curl localhost
hello world
```

In this case, if you manually delete the **nginx-1** pod and query the pods again, you can see that a pod with the same name is created. According to **4s** under **AGE**, it is found that the **nginx-1** pod is newly created.

```
# kubectl delete pod nginx-1
pod "nginx-1" deleted

# kubectl get pods
NAME      READY   STATUS    RESTARTS   AGE
nginx-0   1/1     Running   0           14m
nginx-1   1/1     Running   0           4s
nginx-2   1/1     Running   0           13m
```

Access the **index.html** page of the pod again. **hello world** is still returned, which indicates that the same storage medium is accessed.

```
# kubectl exec -it nginx-1 -- curl localhost
hello world
```

3.4.3 Job and Cron Job

Job and Cron Job

Jobs and cron jobs 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 cron job runs a job periodically on a specified schedule. A cron job 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).

Creating a Job

The following is an example job, which calculates π till the 2000th digit and prints the output. 50 pods need to be run before the job is ended. In this example, print π calculation results for 50 times, and run five pods concurrently. If a pod fails to be run, a maximum of five retries are supported.

```
apiVersion: batch/v1
kind: Job
metadata:
  name: pi-with-timeout
spec:
  completions: 50      # Number of pods that need to run successfully to end the job
  parallelism: 5       # Number of pods that run concurrently. The default value is 1.
  backoffLimit: 5      # Maximum number of retries performed if a pod fails. When the limit is reached,
it will not try again.
  activeDeadlineSeconds: 10 # Timeout interval of pods. Once the time is reached, all pods of the job are
terminated.
  template:            # Pod definition
    spec:
      containers:
        - name: pi
          image: perl
          command:
```

```
- perl
- "-Mbignum=bpi"
- "-wle"
- print bpi(2000)
restartPolicy: Never
```

Based on the **completions** and **Parallelism** settings, jobs can be classified as follows:

Table 3-1 Job types

Job Type	Description	Example
One-off job	One pod runs until it is successfully ends.	Database migration
Jobs with a fixed completion count	One pod runs until the specified completion count is reached.	Pod for processing work queues
Parallel jobs with a fixed completion count	Multiple pods run until the specified completion count is reached.	Multiple pods for processing work queues concurrently
Parallel jobs	One or more pods run until one pod is successfully ended.	Multiple pods for processing work queues concurrently

Creating a Cron Job

Compared with a job, a cron job is a scheduled job. A cron job runs a job periodically on a specified schedule, and the job creates pods.

```
apiVersion: batch/v1beta1
kind: CronJob
metadata:
  name: cronjob-example
spec:
  schedule: "0,15,30,45 * * * *" # Scheduling configuration
  jobTemplate: # Job definition
    spec:
      template:
        spec:
          restartPolicy: OnFailure
          containers:
            - name: main
              image: pi
```

The format of the cron job is as follows:

- Minute
- Hour
- Day of month
- Month
- Day of week

For example, in `0,15,30,45 * * * *`, commas separate minutes, the first asterisk (*) indicates the hour, the second asterisk indicates the day of the month, the third asterisk indicates the month, and the fourth asterisk indicates the day of the week.

If you want to run the job every half an hour on the first day of each month, set this parameter to `0,30 * 1 * *`. If you want to run the job on 3:00 a.m. every Sunday, set this parameter to `0 3 * * 0`.

For details about the cron job format, visit <https://en.wikipedia.org/wiki/Cron>.

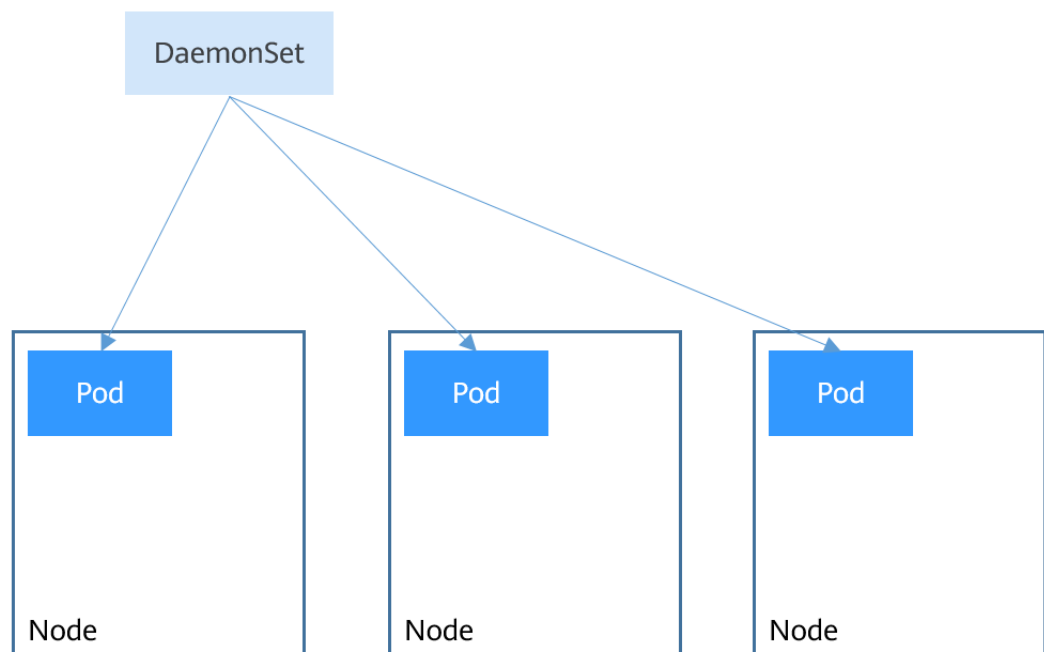
3.4.4 DaemonSet

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 3-13 DaemonSet



Creating a DaemonSet

The following is an example of a DaemonSet:

```
apiVersion: apps/v1
kind: DaemonSet
metadata:
  name: nginx-daemonset
  labels:
    app: nginx-daemonset
spec:
```



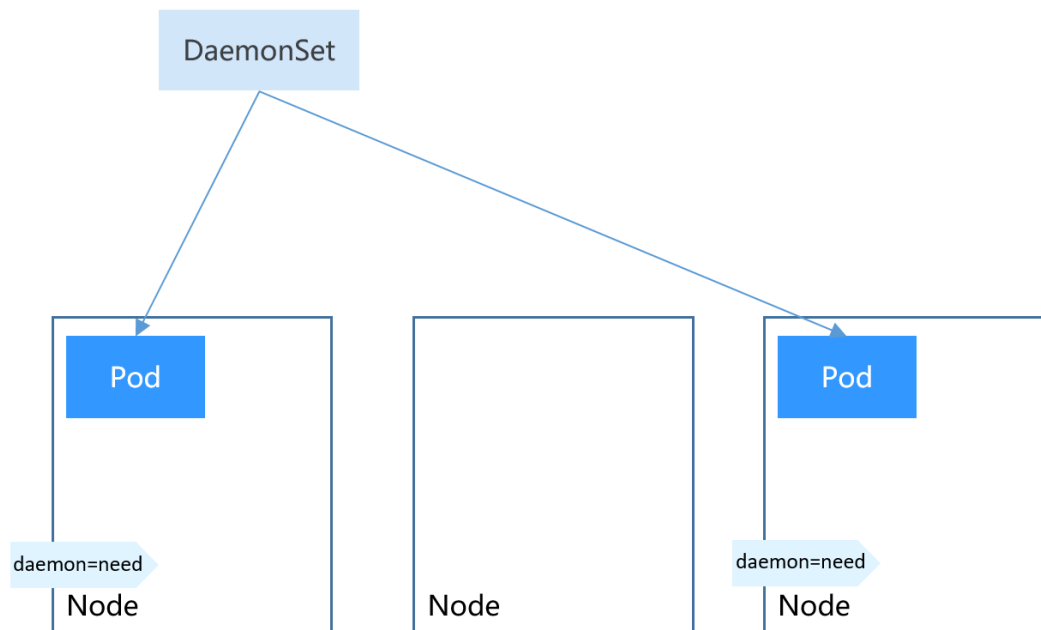
```

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**, as shown in the following figure. If you want to create a pod on each node, delete the label.

Figure 3-14 DaemonSet creating a pod on nodes with a specified label



Create a DaemonSet.

```

$ kubectl create -f daemonset.yaml
daemonset.apps/nginx-daemonset created
  
```

Run the following command. The output shows that **nginx-daemonset** creates no pods on nodes.

```
$ kubectl get ds
NAME           DESIRED  CURRENT  READY  UP-TO-DATE  AVAILABLE  NODE SELECTOR  AGE
nginx-daemonset  0        0        0      0           0          daemon=need    16s
```

```
$ kubectl get pods
No resources found in default namespace.
```

This is because no nodes have the **daemon=need** label. Run the following command to query the labels of nodes:

```
$ kubectl get node --show-labels
NAME           STATUS  ROLES  AGE  VERSION  LABELS
192.168.0.212  Ready  <none>  83m  v1.15.6-r1-20.3.0.2.B001-15.30.2  beta.kubernetes.io/arch=amd64 ...
192.168.0.94   Ready  <none>  83m  v1.15.6-r1-20.3.0.2.B001-15.30.2  beta.kubernetes.io/arch=amd64 ...
192.168.0.97   Ready  <none>  83m  v1.15.6-r1-20.3.0.2.B001-15.30.2  beta.kubernetes.io/arch=amd64 ...
```

Add the **daemon=need** label to node **192.168.0.212**, and then query the pods of **nginx-daemonset** again. It is found that a pod has been created on node **192.168.0.212**.

```
$ kubectl label node 192.168.0.212 daemon=need
node/192.168.0.212 labeled
```

```
$ kubectl get ds
NAME           DESIRED  CURRENT  READY  UP-TO-DATE  AVAILABLE  NODE SELECTOR  AGE
nginx-daemonset  1        1        0      1           0          daemon=need    116s
```

```
$ kubectl get pod -o wide
NAME           READY  STATUS  RESTARTS  AGE  IP           NODE
nginx-daemonset-g9b7j  1/1    Running  0          18s  172.16.3.0  192.168.0.212
```

Add the **daemon=need** label to node **192.168.0.94**. You can find that a pod is created on this node as well.

```
$ kubectl label node 192.168.0.94 daemon=need
node/192.168.0.94 labeled
```

```
$ kubectl get ds
NAME           DESIRED  CURRENT  READY  UP-TO-DATE  AVAILABLE  NODE SELECTOR  AGE
nginx-daemonset  2        2        1      2           1          daemon=need    2m29s
```

```
$ kubectl get pod -o wide
NAME           READY  STATUS             RESTARTS  AGE  IP           NODE
nginx-daemonset-6jjxz  0/1    ContainerCreating  0      8s    <none>      192.168.0.94
nginx-daemonset-g9b7j  1/1    Running            0      42s   172.16.3.0  192.168.0.212
```

Modify the **daemon=need** label of node **192.168.0.94**. You can find the DaemonSet deletes its pod from the node.

```
$ kubectl label node 192.168.0.94 daemon=no --overwrite
node/192.168.0.94 labeled
```

```
$ kubectl get ds
NAME           DESIRED  CURRENT  READY  UP-TO-DATE  AVAILABLE  NODE SELECTOR  AGE
nginx-daemonset  1        1        1      1           1          daemon=need    4m5s
```

```
$ kubectl get pod -o wide
NAME           READY  STATUS  RESTARTS  AGE  IP           NODE
nginx-daemonset-g9b7j  1/1    Running  0          2m23s  172.16.3.0  192.168.0.212
```

3.4.5 Affinity and Anti-Affinity Scheduling

A nodeSelector provides a very simple way to constrain pods to nodes with particular labels, as mentioned in [DaemonSet](#). The affinity and anti-affinity feature greatly expands the types of constraints you can express.

You can define affinity and anti-affinity in node and pod levels. You can configure custom rules to achieve affinity and anti-affinity scheduling. For example, you can

deploy frontend pods and backend pods together, deploy the same type of applications on a specific node, or deploy different applications on different nodes.

Node Affinity

Node affinity is conceptually similar to a nodeSelector as it allows you to constrain which nodes your pod is eligible to be scheduled on, based on labels on the node. The following output lists the labels of node **192.168.0.212**.

```
$ kubectl describe node 192.168.0.212
Name:          192.168.0.212
Roles:         <none>
Labels:        beta.kubernetes.io/arch=amd64
               beta.kubernetes.io/os=linux
               failure-domain.beta.kubernetes.io/is-baremetal=false
               failure-domain.beta.kubernetes.io/region=eu-west-0
               failure-domain.beta.kubernetes.io/zone=eu-west-0a
               kubernetes.io/arch=amd64
               kubernetes.io/availablezone=eu-west-0a
               kubernetes.io/eniquota=12
               kubernetes.io/hostname=192.168.0.212
               kubernetes.io/os=linux
               node.kubernetes.io/subnetid=fd43acad-33e7-48b2-a85a-24833f362e0e
               os.architecture=amd64
               os.name=EulerOS_2.0_SP5
               os.version=3.10.0-862.14.1.5.h328.eulerosv2r7.x86_64
```

These labels are automatically added by CCE during node creation. The following describes a few that are frequently used during scheduling.

- **failure-domain.beta.kubernetes.io/region**: region where the node is located. In the preceding output, the label value is **eu-west-0**, which indicates that the node is located in the Paris (France) region.
- **failure-domain.beta.kubernetes.io/zone**: availability zone to which the node belongs.
- **kubernetes.io/hostname**: host name of the node.

In addition to these automatically added labels, you can tailor labels to your service requirements, as introduced in [Label for Managing Pods](#). Generally, large Kubernetes clusters have various kinds of labels.

When you deploy pods, you can use a nodeSelector, as described in [DaemonSet](#), to constrain pods to nodes with specific labels. The following example shows how to use a nodeSelector to deploy pods only on the nodes with the **gpu=true** label.

```
apiVersion: v1
kind: Pod
metadata:
  name: nginx
spec:
  nodeSelector:      #Node selection. A pod is deployed on a node only when the node has the
                    #gpu=true label.
                    gpu: true
...
```

Node affinity rules can achieve the same results, as shown in the following example.

```
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:
        nodeAffinity:
          requiredDuringSchedulingIgnoredDuringExecution:
            nodeSelectorTerms:
            - matchExpressions:
              - key: gpu
                operator: In
                values:
                - "true"
```

Even though the node affinity rule requires more lines, it is more expressive, which will be further described later.

requiredDuringSchedulingIgnoredDuringExecution seems to be complex, but it can be easily understood as a combination of two parts.

- The **requiredDuringScheduling** part indicates that the rule is a hard requirement, which means that the rule must be met for a pod to be scheduled onto a node.
- The **IgnoredDuringExecution** part indicates that the pods that have been running on the node are not affected. Currently, all node affinity rules provided by Kubernetes end with **IgnoredDuringExecution** as these rules affect only the pods that are being scheduled. In the future, affinity rules end with **RequiredDuringExecution** will be supported, which means that pods will be evicted from nodes that cease to satisfy the pods' label requirements.

In addition, the value of **operator** is **In**, indicating that the label value must be in the values list. Other available operator values are as follows:

- **NotIn**: The label value is not in a list.
- **Exists**: A specific label exists.
- **DoesNotExist**: A specific label does not exist.
- **Gt**: The label value is greater than a specified value (string comparison).
- **Lt**: The label value is less than a specified value (string comparison).

Note that there is no such thing as **nodeAntiAffinity** because operators **NotIn** and **DoesNotExist** provide the same function.

Now, check whether the node affinity rule takes effect. Add the **gpu=true** tag to the **192.168.0.212** node.

```
$ kubectl label node 192.168.0.212 gpu=true
node/192.168.0.212 labeled

$ kubectl get node -L gpu
NAME          STATUS  ROLES  AGE  VERSION          GPU
192.168.0.212 Ready  <none> 13m  v1.15.6-r1-20.3.0.2.B001-15.30.2  true
192.168.0.94  Ready  <none> 13m  v1.15.6-r1-20.3.0.2.B001-15.30.2
192.168.0.97  Ready  <none> 13m  v1.15.6-r1-20.3.0.2.B001-15.30.2
```

Create the Deployment. You can find that all pods are deployed on the **192.168.0.212** node.

```
$ kubectl create -f affinity.yaml
deployment.apps/gpu created

$ kubectl get pod -o wide
NAME          READY  STATUS  RESTARTS  AGE  IP          NODE
gpu-6df65c44cf-42xw4  1/1    Running  0         15s  172.16.0.37 192.168.0.212
gpu-6df65c44cf-jzjvs  1/1    Running  0         15s  172.16.0.36 192.168.0.212
gpu-6df65c44cf-zv5cl  1/1    Running  0         15s  172.16.0.38 192.168.0.212
```

Node Preference Rule

The preceding **requiredDuringSchedulingIgnoredDuringExecution** rule is a hard selection rule. There is another type of selection rule, that is, **preferredDuringSchedulingIgnoredDuringExecution**. It is used to specify which nodes are preferred during scheduling.

To demonstrate its effect, add a node to the cluster and ensure that the node is not in the same AZ with other nodes. After the node is created, query the AZ of the node. As shown in the following output, the newly added node is in eu-west-0a.

```
$ kubectl get node -L failure-domain.beta.kubernetes.io/zone,gpu
NAME          STATUS  ROLES  AGE  VERSION          ZONE  GPU
192.168.0.100 Ready  <none> 7h23m v1.15.6-r1-20.3.0.2.B001-15.30.2  eu-west-0a
192.168.0.212 Ready  <none> 8h    v1.15.6-r1-20.3.0.2.B001-15.30.2  eu-west-0b  true
192.168.0.94  Ready  <none> 8h    v1.15.6-r1-20.3.0.2.B001-15.30.2  eu-west-0b
192.168.0.97  Ready  <none> 8h    v1.15.6-r1-20.3.0.2.B001-15.30.2  eu-west-0b
```

Define a Deployment. Use the **preferredDuringSchedulingIgnoredDuringExecution** rule to set the weight of nodes in **eu-west-0a** as **80** and nodes with the **gpu=true** label as **20**. In this way, pods are preferentially deployed on the node in eu-west-0a.

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: gpu
  labels:
    app: gpu
spec:
  selector:
    matchLabels:
      app: gpu
  replicas: 10
  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:
  nodeAffinity:
    preferredDuringSchedulingIgnoredDuringExecution:
    - weight: 80
      preference:
        matchExpressions:
        - key: failure-domain.beta.kubernetes.io/zone
          operator: In
          values:
          - eu-west-0a
    - weight: 20
      preference:
        matchExpressions:
        - key: gpu
          operator: In
          values:
          - "true"

```

After the deployment, you can find that five pods are deployed on the **192.168.0.212** node, and two pods are deployed on the **192.168.0.100** node.

```

$ kubectl create -f affinity2.yaml
deployment.apps/gpu created

$ kubectl get po -o wide
NAME          READY  STATUS   RESTARTS  AGE  IP           NODE
gpu-585455d466-5bmcz  1/1    Running  0         2m29s  172.16.0.44  192.168.0.212
gpu-585455d466-cg2l6  1/1    Running  0         2m29s  172.16.0.63  192.168.0.97
gpu-585455d466-f2bt2  1/1    Running  0         2m29s  172.16.0.79  192.168.0.100
gpu-585455d466-hdb5n  1/1    Running  0         2m29s  172.16.0.42  192.168.0.212
gpu-585455d466-hkgvz  1/1    Running  0         2m29s  172.16.0.43  192.168.0.212
gpu-585455d466-mngvn  1/1    Running  0         2m29s  172.16.0.48  192.168.0.97
gpu-585455d466-s26qs  1/1    Running  0         2m29s  172.16.0.62  192.168.0.97
gpu-585455d466-sxtzm  1/1    Running  0         2m29s  172.16.0.45  192.168.0.212
gpu-585455d466-t56cm  1/1    Running  0         2m29s  172.16.0.64  192.168.0.100
gpu-585455d466-t5w5x  1/1    Running  0         2m29s  172.16.0.41  192.168.0.212

```

In the preceding example, the node scheduling priority is as follows. Nodes with both **eu-west-0a** and **gpu=true** labels have the highest priority. Nodes with the **eu-west-0a** label but no **gpu=true** label have the second priority (weight: 80). Nodes with the **gpu=true** label but no **eu-west-0a** label have the third priority. Nodes without any of these two labels have the lowest priority.

Figure 3-15 Scheduling priority



From the preceding output, you can find that no pods of the Deployment are scheduled to node **192.168.0.94**. This is because the node already has many pods on it and its resource usage is high. This also indicates that the

preferredDuringSchedulingIgnoredDuringExecution rule defines a preference rather than a hard requirement.

Pod Affinity

Node affinity rules affect only the affinity between pods and nodes. Kubernetes also supports configuring inter-pod affinity rules. For example, the frontend and backend of an application can be deployed together on one node to reduce access latency. There are also two types of inter-pod affinity rules:

requiredDuringSchedulingIgnoredDuringExecution and **preferredDuringSchedulingIgnoredDuringExecution**.

Assume that the backend of an application has been created and has the **app=backend** label.

```
$ kubectl get po -o wide
NAME                READY STATUS RESTARTS AGE IP          NODE
backend-658f6cb858-dlrz8 1/1   Running 0      2m36s 172.16.0.67 192.168.0.100
```

You can configure the following pod affinity rule to deploy the frontend pods of the application to the same node as the backend pods.

```
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:
        podAffinity:
          requiredDuringSchedulingIgnoredDuringExecution:
            - topologyKey: kubernetes.io/hostname
              labelSelector:
                matchLabels:
                  app: backend
```

Deploy the frontend and you can find that the frontend is deployed on the same node as the backend.

```
$ kubectl create -f affinity3.yaml
deployment.apps/frontend created

$ kubectl get po -o wide
NAME                READY STATUS RESTARTS AGE IP          NODE
```

```
backend-658f6cb858-dlrz8 1/1 Running 0 5m38s 172.16.0.67 192.168.0.100
frontend-67ff9b7b97-dsqzn 1/1 Running 0 6s 172.16.0.70 192.168.0.100
frontend-67ff9b7b97-hxm5t 1/1 Running 0 6s 172.16.0.71 192.168.0.100
frontend-67ff9b7b97-z8pdb 1/1 Running 0 6s 172.16.0.72 192.168.0.100
```

The **topologyKey** field specifies the selection range. The scheduler selects nodes within the range based on the affinity rule defined. The effect of **topologyKey** is not fully demonstrated in the preceding example because all the nodes have the **kubernetes.io/hostname** label, that is, all the nodes are within the range.

To see how **topologyKey** works, assume that the backend of the application has two pods, which are running on different nodes.

```
$ kubectl get po -o wide
NAME          READY STATUS RESTARTS AGE IP      NODE
backend-658f6cb858-5bpd6 1/1 Running 0 23m 172.16.0.40 192.168.0.97
backend-658f6cb858-dlrz8 1/1 Running 0 2m36s 172.16.0.67 192.168.0.100
```

Add the **prefer=true** label to nodes **192.168.0.97** and **192.168.0.94**.

```
$ kubectl label node 192.168.0.97 prefer=true
node/192.168.0.97 labeled
$ kubectl label node 192.168.0.94 prefer=true
node/192.168.0.94 labeled

$ kubectl get node -L prefer
NAME          STATUS ROLES AGE VERSION PERFER
192.168.0.100 Ready <none> 44m v1.15.6-r1-20.3.0.2.B001-15.30.2
192.168.0.212 Ready <none> 91m v1.15.6-r1-20.3.0.2.B001-15.30.2
192.168.0.94 Ready <none> 91m v1.15.6-r1-20.3.0.2.B001-15.30.2 true
192.168.0.97 Ready <none> 91m v1.15.6-r1-20.3.0.2.B001-15.30.2 true
```

Define **topologyKey** in the **podAffinity** section as **prefer**.

```
affinity:
  podAffinity:
    requiredDuringSchedulingIgnoredDuringExecution:
    - topologyKey: prefer
      labelSelector:
        matchLabels:
          app: backend
```

The scheduler recognizes the nodes with the **prefer** label, that is, **192.168.0.97** and **192.168.0.94**, and then find the pods with the **app=backend** label. In this way, all frontend pods are deployed onto **192.168.0.97**.

```
$ kubectl create -f affinity3.yaml
deployment.apps/frontend created

$ kubectl get po -o wide
NAME          READY STATUS RESTARTS AGE IP      NODE
backend-658f6cb858-5bpd6 1/1 Running 0 26m 172.16.0.40 192.168.0.97
backend-658f6cb858-dlrz8 1/1 Running 0 5m38s 172.16.0.67 192.168.0.100
frontend-67ff9b7b97-dsqzn 1/1 Running 0 6s 172.16.0.70 192.168.0.97
frontend-67ff9b7b97-hxm5t 1/1 Running 0 6s 172.16.0.71 192.168.0.97
frontend-67ff9b7b97-z8pdb 1/1 Running 0 6s 172.16.0.72 192.168.0.97
```

Pod Anti-affinity

Unlike the scenarios in which pods are preferred to be scheduled onto the same node, sometimes, it could be the exact opposite. For example, if certain pods are deployed together, they will affect the performance.

The following example defines an inter-pod anti-affinity rule, which specifies that pods must not be scheduled to nodes that already have pods with the

app=frontend label, that is, to deploy the pods of the frontend to different nodes with each node has only one replica.

```
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:
          requiredDuringSchedulingIgnoredDuringExecution:
            - topologyKey: kubernetes.io/hostname
          labelSelector:
            matchLabels:
              app: frontend
```

Deploy the frontend and query the deployment results. You can find that each node has only one frontend pod and one pod of the Deployment is **Pending**. This is because when the scheduler is deploying the fifth pod, all nodes already have one pod with the **app=frontend** label on them. There is no available node. Therefore, the fifth pod will remain in the **Pending** status.

```
$ kubectl create -f affinity4.yaml
deployment.apps/frontend created
```

```
$ kubectl get po -o wide
NAME                                READY STATUS RESTARTS AGE IP NODE
frontend-6f686d8d87-8dlsc          1/1 Running 0 18s 172.16.0.76 192.168.0.100
frontend-6f686d8d87-d6l8p          0/1 Pending 0 18s <none> <none>
frontend-6f686d8d87-hgcq2          1/1 Running 0 18s 172.16.0.54 192.168.0.97
frontend-6f686d8d87-q7cfq          1/1 Running 0 18s 172.16.0.47 192.168.0.212
frontend-6f686d8d87-xl8hx          1/1 Running 0 18s 172.16.0.23 192.168.0.94
```

3.5 Configuration Management

3.5.1 ConfigMap

A ConfigMap is a type of resource used to store the configurations required by applications. It is used to store configuration data or configuration files in key-value pairs.

A ConfigMap allows you to decouple configurations from your environments, so that your environments can use different configurations.

Creating a ConfigMap

In the following example, a ConfigMap named **configmap-test** is created. The ConfigMap configuration data is defined in the **data** field.

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: configmap-test
data:
  # Configuration data
  property_1: Hello
  property_2: World
```

Referencing a ConfigMap as an Environment Variable

ConfigMaps are usually referenced as environment variables and in volumes.

In the following example, **property_1** of **configmap-test** is used as the value of the environment variable **EXAMPLE_PROPERTY_1**. After the container is started, it will reference the value of **property_1** as the value of **EXAMPLE_PROPERTY_1**, that is, **Hello**.

```
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
    env:
    - name: EXAMPLE_PROPERTY_1
      valueFrom:
        configMapKeyRef:
          # Reference the ConfigMap.
          name: configmap-test
          key: property_1
    imagePullSecrets:
    - name: default-secret
```

Referencing a ConfigMap in a Volume

Referencing a ConfigMap in a volume is to fill its data in configuration files in the volume. Each piece of data is saved in a file. The key is the file name, and the key value is the file content.

In the following example, create a volume named **vol-configmap**, reference the ConfigMap named **configmap-test** in the volume, and mount the volume to the **/tmp** directory of the container. After the pod is created, the two files **property_1** and **property_2** are generated in the **/tmp** directory of the container, and the values are **Hello** and **World**.

```
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
    volumeMounts:
    - name: vol-configmap          # Mount the volume named vol-configmap.
      mountPath: "/tmp"
  imagePullSecrets:
  - name: default-secret
  volumes:
  - name: vol-configmap
    configMap:                    # Reference the ConfigMap.
      name: configmap-test
```

3.5.2 Secret

A secret is a resource object that is encrypted for storing the authentication information, certificates, and private keys. The sensitive data will not be exposed in images or pod definitions, which is safer and more flexible.

Similar to a ConfigMap, a secret stores data in key-value pairs. The difference is that a secret is encrypted, and is suitable for storing sensitive information.

Base64 Encoding

A secret stores data in key-value pairs, the same form as that of a ConfigMap. The difference is that the value must be encoded using Base64 when a secret is created.

To encode a character string using Base64, run the **echo -n *to-be-encoded content* | base64** command. The following is an example:

```
root@ubuntu:~# echo -n "3306" | base64
MzMwNg==
```

Creating a Secret

The secret defined in the following example contains two key-value pairs.

```
apiVersion: v1
kind: Secret
metadata:
  name: mysecret
data:
  key1: aGVsbG8gd29ybGQ= # hello world, a value encoded using Base64
  key2: MzMwNg==        # 3306, a value encoded using Base64
```

Referencing a Secret as an Environment Variable

Secrets are usually injected into containers as environment variables, as shown in the following example.

```
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
    env:
    - name: key
      valueFrom:
        secretKeyRef:
          name: mysecret
          key: key1
    imagePullSecrets:
    - name: default-secret
```

Referencing a Secret in a Volume

Referencing a secret in a volume is to fill its data in configuration files in the volume. Each piece of data is saved in a file. The key is the file name, and the key value is the file content.

In the following example, create a volume named **vol-secret**, reference the secret named **mysecret** in the volume, and mount the volume to the **/tmp** directory of the container. After the pod is created, the two files **key1** and **key2** are generated in the **/tmp** directory of the container.

```
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
    volumeMounts:
    - name: vol-secret          # Mount the volume named vol-secret.
      mountPath: "/tmp"
    imagePullSecrets:
    - name: default-secret
  volumes:
  - name: vol-secret
    secret:                    # Reference the secret.
      secretName: mysecret
```

In the pod container, you can find the two files **key1** and **key2** in the **/tmp** directory. The values in the files are the values encoded using Base64, which are **hello world** and **3306**.

3.6 Kubernetes Networking

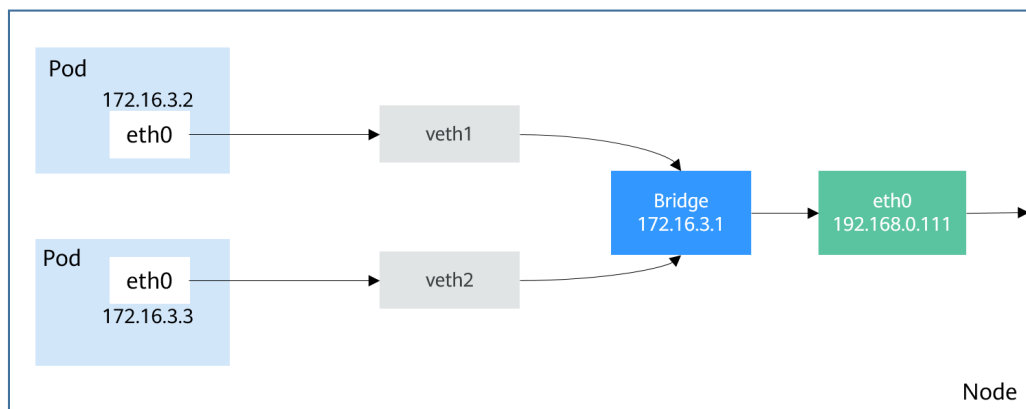
3.6.1 Container Networking

Networking among pods, clusters, and nodes is not implemented by Kubernetes itself, but by the Container Network Interface (CNI) plug-ins. There are many open-source CNI plug-ins, such as Flannel and Calico. CCE also provides customized CNI plug-ins for you to use VPCs when running Kubernetes.

Kubernetes requires that pods in a cluster can communicate with each other and the pods must be connected through a non-NAT network. That is, the source IP address of the received data packet is that of the pod that sends the data packet. Pods are also required to communicate with nodes through a non-NAT network. However, when the pod accesses an object outside the cluster, the source IP address is changed to the node IP address.

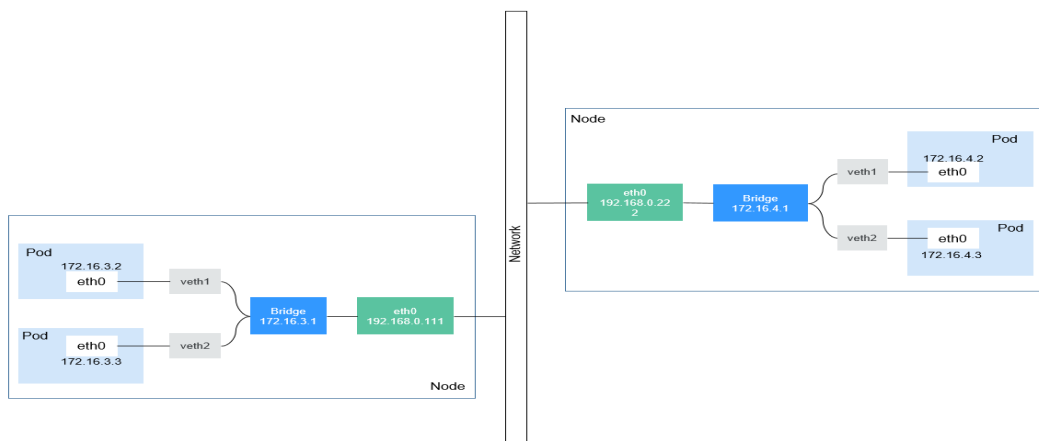
A pod is connected to external systems through a virtual Ethernet interface pair (veth pair). For pods on the same node, they communicate with each other through a Linux bridge, as shown in the following figure.

Figure 3-16 Communication for pods on the same node



Bridges between different nodes can be implemented in multiple modes. However, in a cluster, the pod IP address must be unique. Therefore, cross-node bridges will use different CIDR blocks to prevent duplicate pod IP addresses.

Figure 3-17 Communication for pods on different nodes



The following sections [Services](#) and [Ingresses](#) will describe how Kubernetes provides access solutions for users based on the container networking.

3.6.2 Services

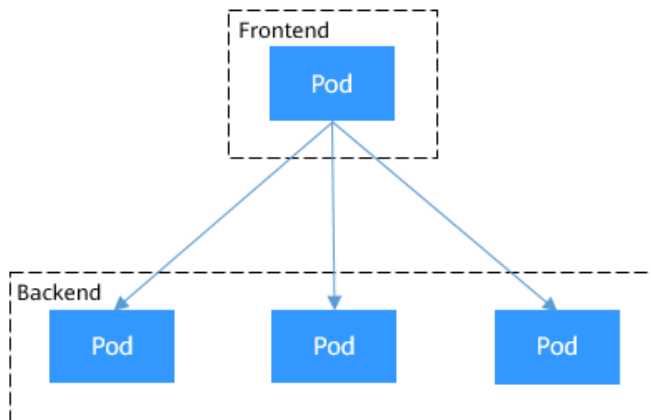
Direct Access to a Pod

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 3-18](#). 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 3-18 Inter-pod access

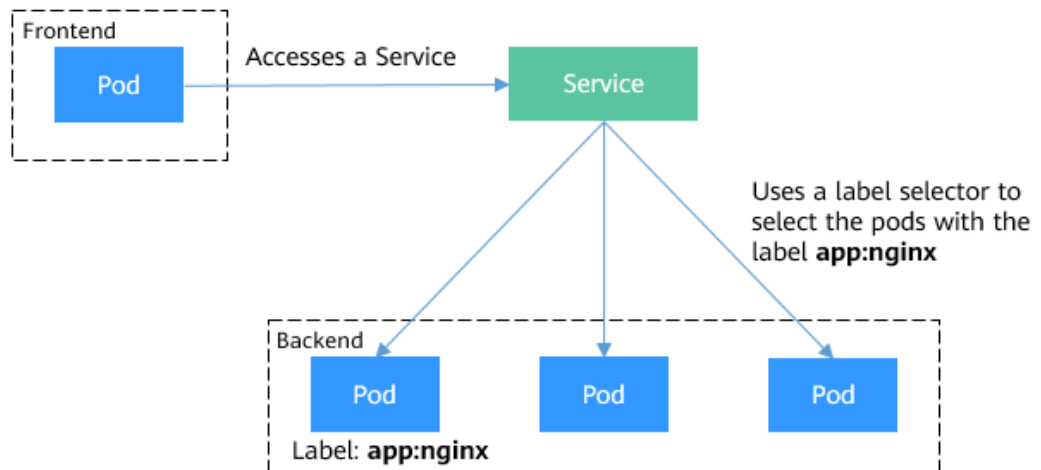


Using Services for Pod Access

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 3-19](#).

Figure 3-19 Accessing pods through a Service



Creating Backend Pods

Create a Deployment with three replicas, that is, three pods with label **app:nginx**.

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx
spec:
  replicas: 3
  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
```

Creating a Service

In the following example, we create a Service named **nginx**, and use a selector to select the pod with the label **app:nginx**. The port of the target pod is port 80 while the exposed port of the Service is port 8080.

The Service can be accessed using **Service name:Exposed port**. In the example, **nginx:8080** is used. In this case, other pods can access the pod associated with **nginx** using **nginx:8080**.

```
apiVersion: v1
kind: Service
metadata:
```

```
name: nginx #Service name
spec:
  selector: #Label selector, which selects pods with the label of app=nginx
    app: nginx
  ports:
    - name: service0
      targetPort: 80 #Pod port
      port: 8080 #Service external port
      protocol: TCP #Forwarding protocol type. The value can be TCP or UDP.
      type: ClusterIP #Service type
```

Save the Service definition to **nginx-svc.yaml** and use `kubectl` to create the Service.

```
$ kubectl create -f nginx-svc.yaml
service/nginx created

$ kubectl get svc
NAME         TYPE        CLUSTER-IP   EXTERNAL-IP  PORT(S)    AGE
kubernetes  ClusterIP   10.247.0.1    <none>       443/TCP    7h19m
nginx       ClusterIP   10.247.124.252 <none>       8080/TCP   5h48m
```

You can see that the Service has a ClusterIP, which is fixed unless the Service is deleted. You can use this ClusterIP to access the Service inside the cluster.

Create a pod and use the ClusterIP to access the pod. Information similar to the following is returned.

```
$ kubectl run -i --tty --image nginx:alpine test --rm /bin/sh
If you don't see a command prompt, try pressing enter.
/ # curl 10.247.124.252:8080
<!DOCTYPE html>
<html>
<head>
<title>Welcome to nginx!</title>
...
```

Using ServiceName to Access a Service

After the DNS resolves the domain name, you can use **ServiceName:Port** to access the Service, the most common practice in Kubernetes. When you are creating a CCE cluster, you are required to install the `coredns` add-on by default. You can view the pods of CoreDNS 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.

For example, if the Service named **nginx** is created, you can access the Service through **nginx:8080** and then access backend pods.

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.

Now, create a pod and access the pod. Query the IP address of the nginx Service domain name, which is 10.247.124.252. Access the domain name of the pod and information similar to the following is returned.

```
$ kubectl run -i --tty --image tutum/dnsutils dnsutils --restart=Never --rm /bin/sh
If you don't see a command prompt, try pressing enter.
/ # nslookup nginx
Server:      10.247.3.10
Address:    10.247.3.10#53

Name:   nginx.default.svc.cluster.local
Address: 10.247.124.252

/ # curl nginx:8080
<!DOCTYPE html>
<html>
<head>
<title>Welcome to nginx!</title>
...
```

Using Services for Service Discovery

After a Service is deployed, it can discover the pod no matter how the pod changes.

If you run the **kubectl describe** command to query the Service, information similar to the following is displayed:

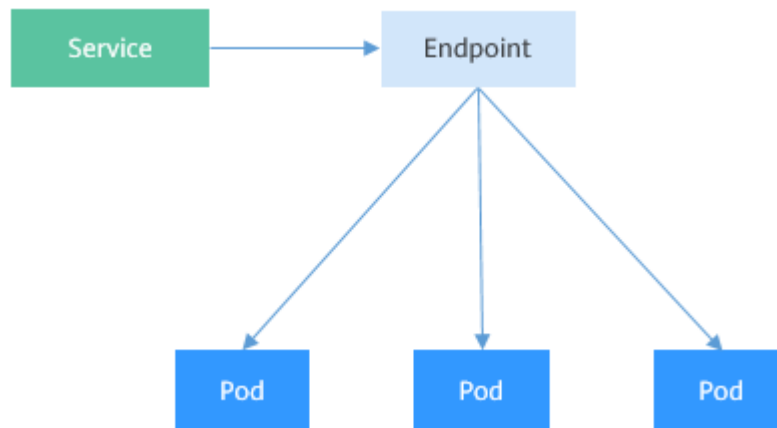
```
$ kubectl describe svc nginx
Name:          nginx
.....
Endpoints:    172.16.2.132:80,172.16.3.6:80,172.16.3.7:80
.....
```

One Endpoints record is displayed. An endpoint is also a resource object in Kubernetes. Kubernetes monitors the pod IP addresses through endpoints so that a Service can discover pods.

```
$ kubectl get endpoints
NAME           ENDPOINTS                                     AGE
kubernetes    192.168.0.127:5444                            7h19m
nginx         172.16.2.132:80,172.16.3.6:80,172.16.3.7:80 5h48m
```

In this example, **172.16.2.132:80** is the **IP:port** of the pod. You can run the following command to view the IP address of the pod, which is the same as the preceding IP address.

```
$ kubectl get po -o wide
NAME                READY   STATUS    RESTARTS   AGE   IP           NODE
nginx-869759589d-dnknn 1/1     Running  0          5h40m 172.16.3.7   192.168.0.212
nginx-869759589d-fcxhh 1/1     Running  0          5h40m 172.16.3.6   192.168.0.212
nginx-869759589d-r69kh 1/1     Running  0          5h40m 172.16.2.132 192.168.0.94
```



If a pod is deleted, the Deployment re-creates the pod and the IP address of the new pod changes.

```
$ kubectl delete po nginx-869759589d-dnknn
pod "nginx-869759589d-dnknn" deleted
```

```
$ kubectl get po -o wide
NAME                READY STATUS RESTARTS AGE IP          NODE
nginx-869759589d-fcxhh 1/1   Running 0       5h41m 172.16.3.6 192.168.0.212
nginx-869759589d-r69kh 1/1   Running 0       5h41m 172.16.2.132 192.168.0.94
nginx-869759589d-w98wg 1/1   Running 0       7s    172.16.3.10 192.168.0.212
```

Check the endpoints again. You can see that the content under **ENDPOINTS** changes with the pod.

```
$ kubectl get endpoints
NAME      ENDPOINTS
kubernetes 192.168.0.127:5444
nginx     172.16.2.132:80,172.16.3.10:80,172.16.3.6:80
```

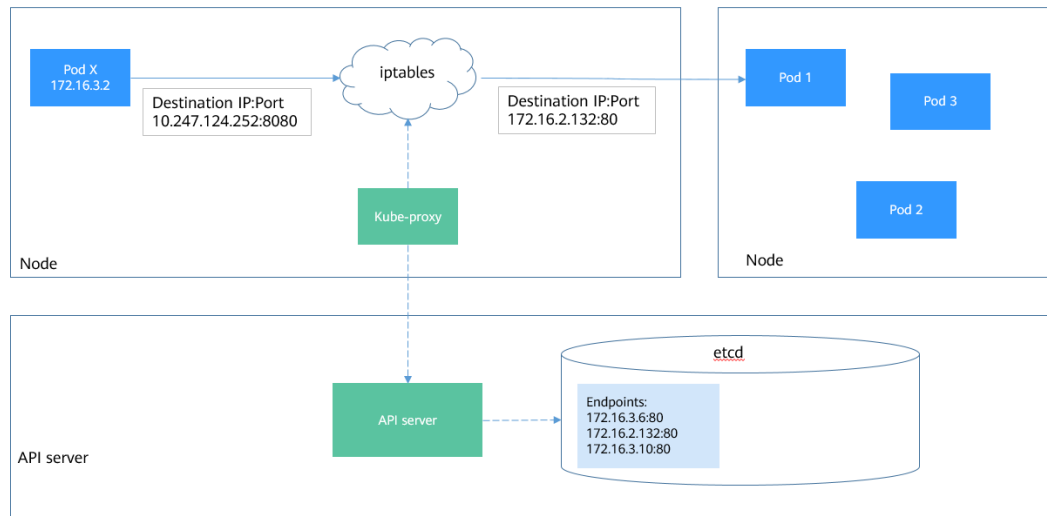
Let's take a closer look at how this happens.

We have introduced kube-proxy on worker nodes in [Figure 3-4](#). Actually, all Service-related operations are performed by kube-proxy. When a Service is created, Kubernetes allocates an IP address to the Service and notifies kube-proxy on all nodes of the Service creation through the API server. After receiving the notification, each kube-proxy records the relationship between the Service and the IP address/port pair through iptables. In this way, the Service can be queried on each node.

The following figure shows how a Service is accessed. Pod X accesses the Service (10.247.124.252:8080). When pod X sends data packets, the destination IP:Port is replaced with the IP:Port of pod 1 based on the iptables rule. In this way, the real backend pod can be accessed through the Service.

In addition to recording the relationship between Services and IP address/port pairs, kube-proxy also monitors the changes of Services and endpoints to ensure that pods can be accessed through Services after pods are rebuilt.

Figure 3-20 Service access process



Service Types and Application Scenarios

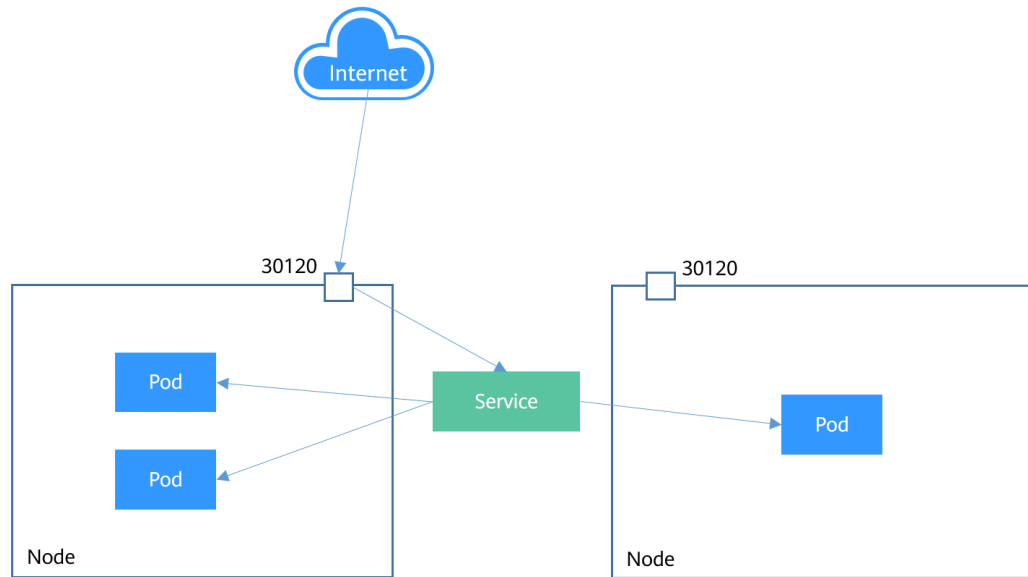
Services of the ClusterIP, NodePort, LoadBalancer, and None types have different functions.

- 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. For details, see [NodePort Services](#).
- 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, see [LoadBalancer Services](#).
- None: used for mutual discovery between pods. This type of Service is also called headless Service. For details, see [Headless Service](#).

NodePort Services

A NodePort Service enables each node in a Kubernetes cluster to reserve the same port. External systems first access the node IP:Port and then the NodePort Service forwards the requests to the pod corresponding to the Service.

Figure 3-21 NodePort Service



The following is an example of creating a NodePort Service. After the Service is created, you can access backend pods through IP:Port of the node.

```
apiVersion: v1
kind: Service
metadata:
  name: nodeport-service
spec:
  type: NodePort
  ports:
    - port: 8080
      targetPort: 80
      nodePort: 30120
  selector:
    app: nginx
```

Create and view the Service. The value of **PORT** for the NodePort Service is **8080:30120/TCP**, indicating that port 8080 of the Service is mapped to port 30120 of the node.

```
$ kubectl create -f nodeport.yaml
service/nodeport-service created

$ kubectl get svc -o wide
NAME          TYPE          CLUSTER-IP    EXTERNAL-IP  PORT(S)          AGE  SELECTOR
kubernetes    ClusterIP     10.247.0.1    <none>       443/TCP         107m <none>
nginx         ClusterIP     10.247.124.252 <none>       8080/TCP        16m  app=nginx
nodeport-service NodePort      10.247.210.174 <none>       8080:30120/TCP 17s  app=nginx
```

Access the Service by using Node IP:Port number to access the pod.

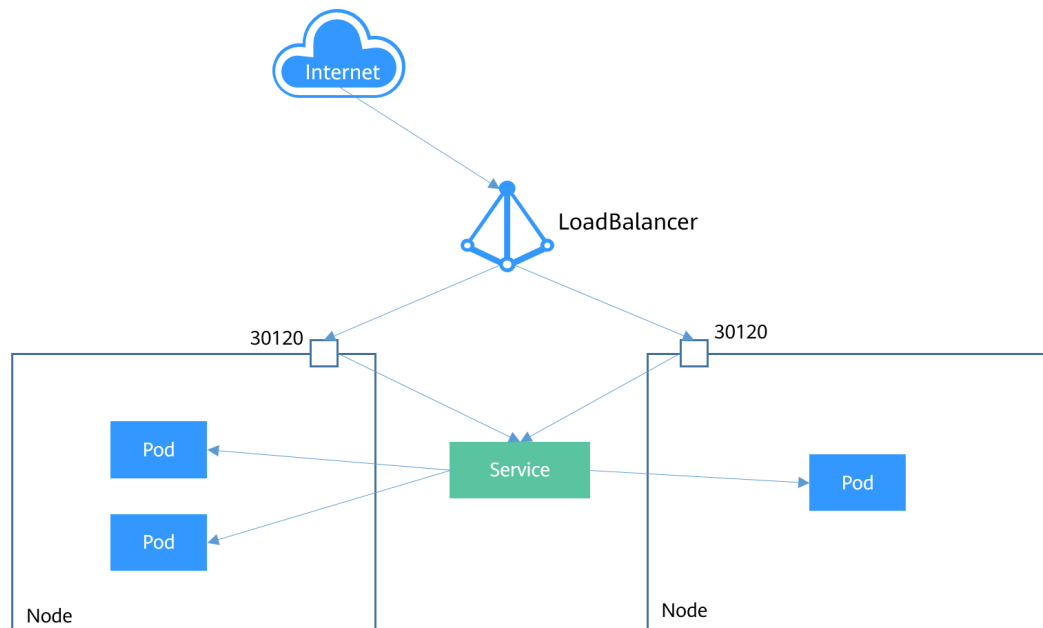
```
$ kubectl run -i --tty --image nginx:alpine test --rm /bin/sh
If you don't see a command prompt, try pressing enter.
/ # curl 192.168.0.212:30120
<!DOCTYPE html>
<html>
<head>
<title>Welcome to nginx!</title>
.....
```

LoadBalancer Services

A Service is exposed externally using a load balancer that forwards requests to the NodePort of the node.

Load balancers are not a Kubernetes component. Different cloud service providers have different load balancers. For example, CCE interconnects with Elastic Load Balance (ELB) for you to use load balancers. As a result, there are different implementation methods of creating a LoadBalancer Service.

Figure 3-22 LoadBalancer Service



The following is an example of creating a LoadBalancer Service. After the LoadBalancer Service is created, you can access backend pods through IP:Port of the load balancer.

```

apiVersion: v1
kind: Service
metadata:
  annotations:
    kubernetes.io/elb.id: 3c7caa5a-a641-4bff-801a-feace27424b6
  labels:
    app: nginx
    name: nginx
spec:
  loadBalancerIP: 10.78.42.242 # IP address of the load balancer
  ports:
  - name: service0
    port: 80
    protocol: TCP
    targetPort: 80
    nodePort: 30120
  selector:
    app: nginx
  type: LoadBalancer # Service type (LoadBalancer)
    
```

The parameters in **annotations** under **metadata** are required for CCE LoadBalancer Services. They specify the load balancer to which the Service is bound. CCE also allows you to create a load balancer when creating a LoadBalancer Service. For details, see [LoadBalancer](#).

Headless Service

The preceding types of 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 in [StatefulSet](#) 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 of the port for communication between pods
      port: 80     # Port number for communication between pods
  selector:
    app: nginx    # Select the pod whose label is app:nginx.
  clusterIP: None # Set this parameter to None, indicating the headless Service.
```

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 don't 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

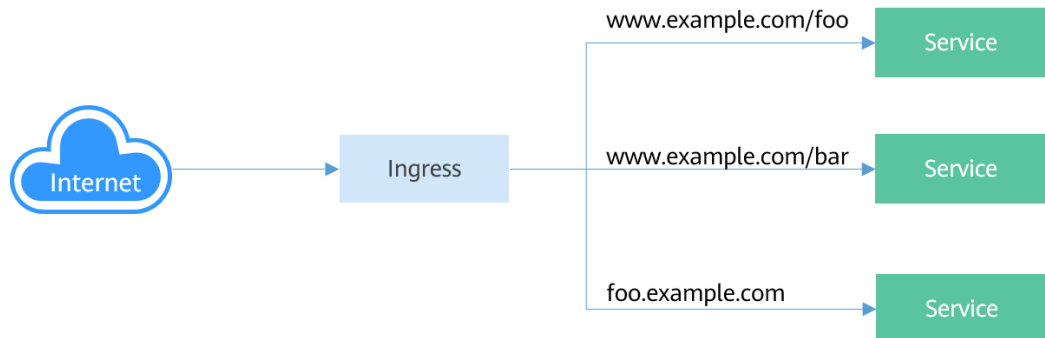
/ # nslookup nginx-2.nginx
Server:      10.247.3.10
Address:    10.247.3.10#53
Name:      nginx-2.nginx.default.svc.cluster.local
Address: 172.16.0.19
```

3.6.3 Ingresses

Why We Need Ingresses

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 3-23 Ingress and Service

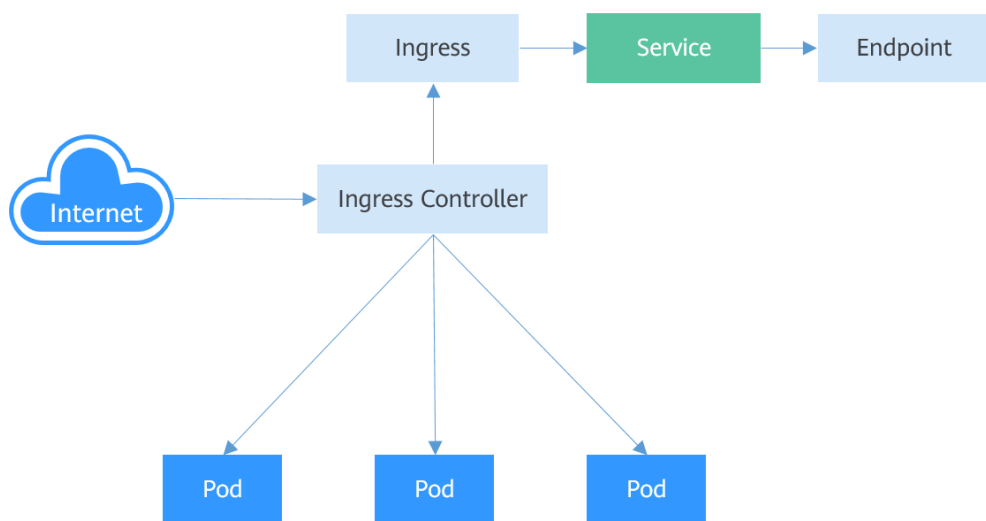


Ingress Working Mechanism

To use ingresses, you must install Ingress Controller on your Kubernetes cluster. Ingress Controller can be implemented in multiple modes. The most common one is **NGINX Ingress Controller** maintained by Kubernetes. CCE works with Elastic Load Balance (ELB) to implement layer-7 load balancing (via ingresses).

An external request is first sent to Ingress Controller. Then, Ingress Controller locates the corresponding Service based on the routing rule of an ingress, queries the IP address of the pod through the Endpoint, and forwards the request to the pod.

Figure 3-24 Ingress working mechanism



Creating an Ingress

In the following example, an ingress that uses the HTTP protocol, associates with backend Service **nginx:8080**, and uses a load balancer (specified by **metadata.annotations**) is created. After the request for accessing **http://192.168.10.155:8080/test** is initiated, the traffic is forwarded to Service **nginx:8080**, which in turn forwards the traffic to the corresponding pod.

```
apiVersion: networking.k8s.io/v1beta1
kind: Ingress
metadata:
  name: test-ingress
  annotations:
    kubernetes.io/ingress.class: cce
    kubernetes.io/elb.port: '8080'
    kubernetes.io/elb.ip: 192.168.10.155
    kubernetes.io/elb.id: aa7cf5ec-7218-4c43-98d4-c36c0744667a
spec:
  rules:
  - host: ""
    http:
      paths:
      - backend:
          serviceName: nginx
          servicePort: 8080
        path: "/test"
    property:
      ingress.beta.kubernetes.io/url-match-mode: STARTS_WITH
```

You can also set the external domain name in an ingress so that you can access the load balancer through the domain name and then access backend Services.

NOTE

Domain name-based access depends on domain name resolution. You need to point the domain name to the IP address of the load balancer. For example, you can use Domain Name Service (DNS) to resolve domain names.

```
spec:
  rules:
  - host: www.example.com # Domain name
    http:
      paths:
      - path: /
        backend:
          serviceName: nginx
          servicePort: 80
```

Accessing Multiple Services

An ingress can access multiple Services at the same time. The configuration is as follows:

- When you access **http://foo.bar.com/foo**, the backend Service **s1:80** is accessed.
- When you access **http://foo.bar.com/bar**, the backend Service **s2:80** is accessed.

```
spec:
  rules:
  - host: foo.bar.com # Host address
    http:
      paths:
      - path: "/foo"
        backend:
          serviceName: s1
```



```
servicePort: 80
- path: "/bar"
backend:
  serviceName: s2
  servicePort: 80
```

3.6.4 Readiness Probe

After a pod is created, the Service can immediately select it and forward requests to it. However, it takes time to start a pod. If the pod is not ready (it takes time to load the configuration or data, or a preheating program may need to be executed), the pod cannot process requests, and the requests will fail.

Kubernetes solves this problem by adding a readiness probe to pods. A pod with containers reporting that they are not ready does not receive traffic through Kubernetes Services.

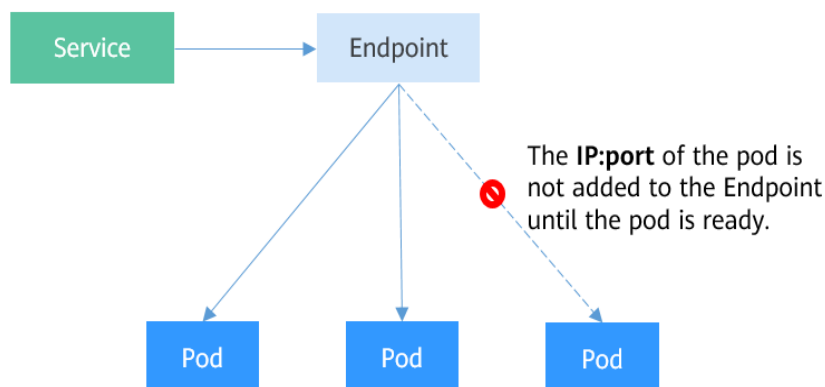
A readiness probe periodically detects a pod and determines whether the pod is ready based on its response. Similar to **Liveness Probe**, there are three types of readiness probes.

- Exec: kubelet executes a command in the target container. If the command succeeds, it returns **0**, and kubelet considers the container to be ready.
- HTTP GET: The probe sends an HTTP GET request to **IP:port** of the container. If the probe receives a 2xx or 3xx status code, the container is considered to be ready.
- TCP Socket: The kubelet attempts to establish a TCP connection with the container. If it succeeds, the container is considered ready.

How Readiness Probes Work

Endpoints can be used as a readiness probe. When a pod is not ready, the **IP:port** of the pod is deleted from the Endpoint and is added to the Endpoint after the pod is ready, as shown in the following figure.

Figure 3-25 How readiness probes work



Exec

The Exec mode is the same as the HTTP GET mode. As shown below, the probe runs the **ls /ready** command. If the file exists, **0** is returned, indicating that the pod is ready. Otherwise, a non-zero status code is returned.

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx
spec:
  replicas: 3
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - image: nginx:alpine
        name: container-0
        resources:
          limits:
            cpu: 100m
            memory: 200Mi
          requests:
            cpu: 100m
            memory: 200Mi
        readinessProbe: # Readiness Probe
        exec: # Define the ls /ready command.
          command:
            - ls
            - /ready
      imagePullSecrets:
      - name: default-secret
```

Save the definition of the Deployment to the **deploy-read.yaml** file, delete the previously created Deployment, and use the **deploy-read.yaml** file to recreate the Deployment.

```
# kubectl delete deploy nginx
deployment.apps "nginx" deleted

# kubectl create -f deploy-read.yaml
deployment.apps/nginx created
```

The **nginx** image does not contain the **/ready** file. Therefore, the container is not in the **Ready** status after the creation, as shown below. Note that the values in the **READY** column are **0/1**, indicating that the containers are not ready.

```
# kubectl get po
NAME                READY   STATUS    RESTARTS   AGE
nginx-7955fd7786-686hp 0/1     Running   0           7s
nginx-7955fd7786-9tgwq 0/1     Running   0           7s
nginx-7955fd7786-bqsbj 0/1     Running   0           7s
```

Create a Service.

```
apiVersion: v1
kind: Service
metadata:
  name: nginx
spec:
  selector:
    app: nginx
  ports:
```

```
- name: service0
  targetPort: 80
  port: 8080
  protocol: TCP
  type: ClusterIP
```

Check the Service. If there are no values in the **Endpoints** line, no Endpoints are found.

```
$ kubectl describe svc nginx
Name:          nginx
.....
Endpoints:
.....
```

If a **/ready** file is created in the container to make the readiness probe succeed, the container is in the **Ready** status. Check the pod and endpoints. It is found that the container for which the **/ready** file is created is ready and an endpoint is added.

```
# kubectl exec nginx-7955fd7786-686hp -- touch /ready

# kubectl get po -o wide
NAME                READY   STATUS    RESTARTS   AGE   IP
nginx-7955fd7786-686hp 1/1     Running   0          10m   192.168.93.169
nginx-7955fd7786-9tgwq 0/1     Running   0          10m   192.168.166.130
nginx-7955fd7786-bqsbj 0/1     Running   0          10m   192.168.252.160

# kubectl get endpoints
NAME      ENDPOINTS          AGE
nginx    192.168.93.169:80 14d
```

HTTP GET

The configuration of a readiness probe is the same as that of a **liveness probe**, which is also in the **containers** field of the pod description template. As shown below, the readiness probe sends an HTTP request to the pod. If the probe receives **2xx** or **3xx**, the pod is ready.

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx
spec:
  replicas: 3
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - image: nginx:alpine
        name: container-0
        resources:
          limits:
            cpu: 100m
            memory: 200Mi
          requests:
            cpu: 100m
            memory: 200Mi
        readinessProbe:
          httpGet:
            path: /read
            port: 80
          # readinessProbe
          # HTTP GET definition
```

```
imagePullSecrets:
- name: default-secret
```

TCP Socket

The following example shows how to define a TCP Socket-type probe.

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx
spec:
  replicas: 3
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - image: nginx:alpine
        name: container-0
        resources:
          limits:
            cpu: 100m
            memory: 200Mi
          requests:
            cpu: 100m
            memory: 200Mi
        readinessProbe:
          # readinessProbe
          tcpSocket:
            # TCP Socket definition
            port: 80
        imagePullSecrets:
          - name: default-secret
```

Advanced Settings of a Readiness Probe

Similar to a liveness probe, a readiness probe also has the same advanced configuration items. The output of the **describe** command of the **nginx** pod is as follows:

```
Readiness: exec [ls /var/ready] delay=0s timeout=1s period=10s #success=1 #failure=3
```

This is the detailed configuration information of the readiness probe.

- **delay=0s** indicates that the probe starts immediately after the container is started.
- **timeout=1s** indicates that the container must respond to the probe within 1s. Otherwise, it is considered as a failure.
- **period=10s** indicates that the probe is performed every 10s.
- **#success=1** indicates that the probe is considered successful as long as the probe succeeds once.
- **#failure=3** indicates that the probe is considered failed if it fails for three consecutive times.

These are the default configurations when the probe is created. You can customize them as follows:

```
readinessProbe: # Readiness Probe
  exec: # Define the ls /readiness/ready command
```

```

command:
- ls
- /readiness/ready
initialDelaySeconds: 10 # Readiness probes are initiated after the container has started for 10s.
timeoutSeconds: 2 # The container must respond within 2s. Otherwise, it is considered as a
failure.
periodSeconds: 30 # The probe is performed every 30s.
successThreshold: 1 # The container is considered ready as long as the probe succeeds once.
failureThreshold: 3 # The probe is considered to be failed after three consecutive failures.

```

3.6.5 NetworkPolicy

NetworkPolicy is a Kubernetes object used to restrict pod access. By setting a NetworkPolicy, you can define ingress rules specifying what addresses can be allowed to the selected pods or egress rules specifying what addresses can be allowed from the selected pods. This is equivalent to setting up a firewall at the application layer to further ensure network security.

NetworkPolicy is implemented by the network plugin of the cluster to which NetworkPolicy will apply. Therefore, the type of rules supported by NetworkPolicy depends on the network plugin. For example, CCE clusters support only ingress rules of pods.

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.

NetworkPolicy rules are classified into the following types:

- namespaceSelector: This selects particular namespaces for which all pods should be allowed as ingress sources or egress destinations.
- podSelector: This selects particular pods in the same namespace as the NetworkPolicy which should be allowed as ingress sources or egress destinations.
- ipBlock: This selects particular IP CIDR ranges to allow as ingress sources or egress destinations. Currently, CCE does not support ipBlock.

Using podSelector to Select Ingress Sources and Egress Destinations

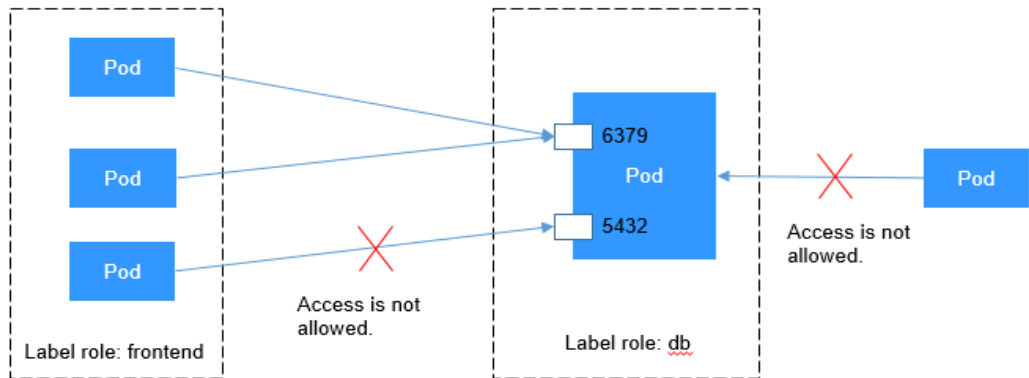
```

apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: test-network-policy
  namespace: default
spec:
  podSelector:
    matchLabels:
      role: db
  ingress:
    #This is an ingress rule.
    - from:
      - podSelector:
          #Only traffic from the pods with the "role=frontend" label is allowed.
          matchLabels:
            role: frontend
  ports:
    #Only TCP can be used to access port 6379.
    - protocol: TCP
      port: 6379

```

Figure 3-26 shows how podSelector selects ingress sources.

Figure 3-26 podSelector



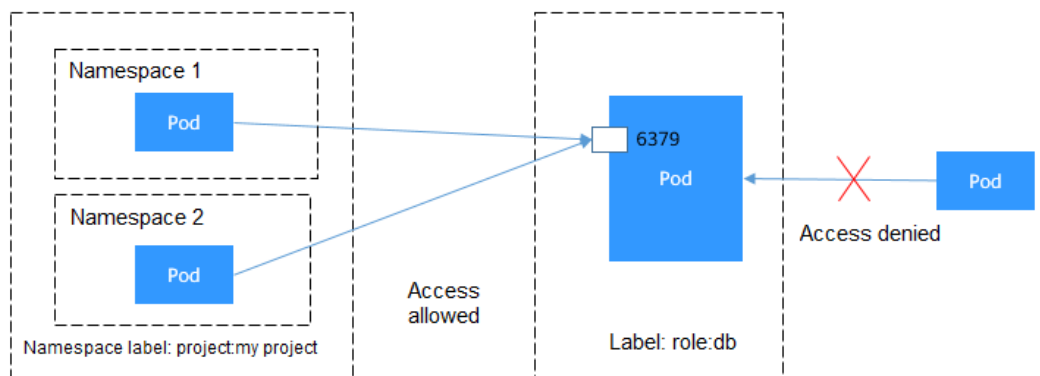
Using namespaceSelector to Select Ingress Sources

```

apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: test-network-policy
spec:
  podSelector:
    matchLabels:
      role: db
  ingress:
    #This is an ingress rule.
    - from:
      - namespaceSelector:
          #Only traffic from the pods in the namespace with the "project=myproject" label
          is allowed.
          matchLabels:
            project: myproject
  ports:
    #Only TCP can be used to access port 6379.
    - protocol: TCP
      port: 6379
    
```

Figure 3-27 shows how namespaceSelector selects ingress sources.

Figure 3-27 namespaceSelector



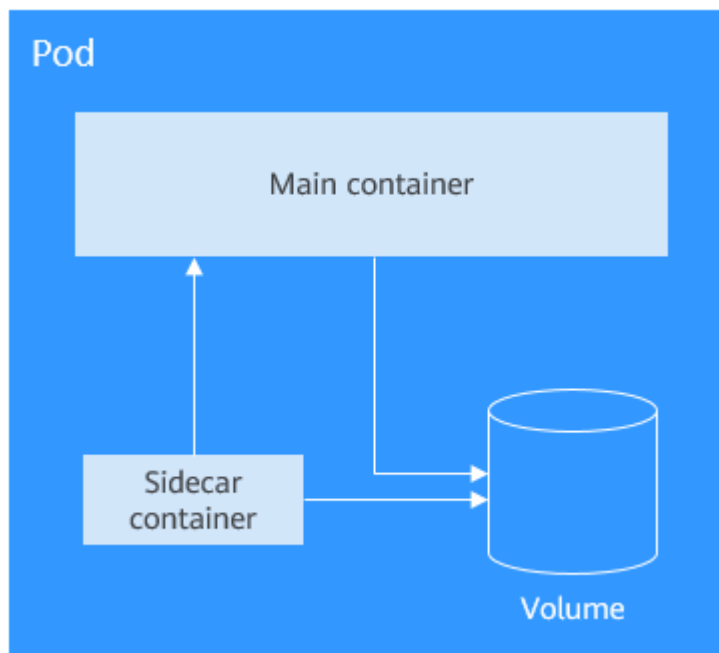
3.7 Persistent Storage

3.7.1 Volume

On-disk files in a container are ephemeral. When a container crashes and is then restarted, the files in the container will be lost. When multiple containers run in a pod, files often need to be shared between these containers. Kubernetes provides an abstraction to solve these two problems, that is, storage volumes. 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 be attached and can be attached to any directory in the 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 attached does not exist. However, files in the volume may outlive the volume, depending on the volume type.

Volume Types

Kubernetes supports multiple types of volumes. The most commonly used ones are as follows:

- `emptyDir`: an empty volume used for temporary storage
- `hostPath`: a volume that mounts a directory of the host into your pod
- `ConfigMap` and `secret`: special volumes that inject or pass information to your pod. For details about how to mount ConfigMaps and secrets, see [ConfigMap](#) and [Secret](#).
- `persistentVolumeClaim`: Kubernetes persistent storage class. For details, see [PersistentVolumes](#), [PersistentVolumeClaims](#), and [StorageClasses](#).

emptyDir

emptyDir is an empty volume in which your applications can read and write the same files. The lifetime of an emptyDir volume is the same as that of the pod it belongs to. After the pod is deleted, data in the volume is also deleted.

Some uses of an emptyDir volume are as follows:

- scratch space, such as for a disk-based merge sort
- checkpointing a long computation for recovery from crashes

Example emptyDir configuration:

```
apiVersion: v1
kind: Pod
metadata:
  name: nginx
spec:
  containers:
  - image: nginx:alpine
    name: test-container
    volumeMounts:
    - mountPath: /cache
      name: cache-volume
  volumes:
  - name: cache-volume
    emptyDir: {}
```

emptyDir volumes are stored on the disks of the node where the pod is located. You can also set the storage medium to the node memory, for example, by setting **medium** to **Memory**.

```
volumes:
- name: html
  emptyDir:
    medium: Memory
```

HostPath

hostPath is a persistent storage volume. Data in an emptyDir volume will be deleted when the pod is deleted, but not the case for a hostPath volume. Data in a hostPath volume will still be stored in the node path to which the volume was mounted. If the pod is re-created and scheduled to the same node, after a new hostPath volume is mounted, previous data written by the pod can still be read.

Data stored in hostPath volumes is related to the node. Therefore, hostPath is not suitable for applications such as databases. For example, if the pod in which a database instance runs is scheduled to another node, the read data will be totally different.

Therefore, you are not advised to use hostPath to store cross-pod data, because after the pod is rebuilt, it will be randomly scheduled to a node, which may cause inconsistency when data is written.

```
apiVersion: v1
kind: Pod
metadata:
  name: test-hostpath
spec:
  containers:
  - image: nginx:alpine
    name: hostpath-container
    volumeMounts:
```



```
- mountPath: /test-pd
  name: test-volume
volumes:
- name: test-volume
  hostPath:
    path: /data
```

3.7.2 PersistentVolumes, PersistentVolumeClaims, and StorageClasses

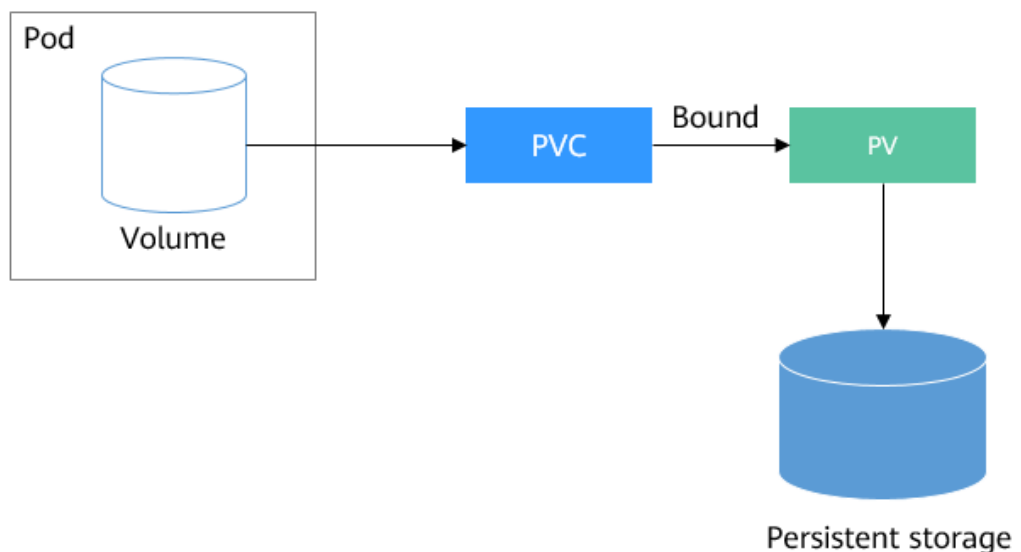
hostPath volumes are used for persistent storage. However, hostPath volumes are node-specific. Writing data into hostPath volumes after a node restart may cause data inconsistency.

If you want to read the previously written data after a pod is rebuilt and scheduled again, you can count on network storage. Typically, a cloud vendor provides at least three classes of network storage: block storage, file storage, and object storage. Kubernetes decouples how storage is provided from how it is consumed by introducing two API objects: PersistentVolume (PV) and PersistentVolumeClaim (PVC). You only need to request the storage resources you want, without being exposed to the details of how they are implemented.

- A PV describes a persistent data storage volume. It defines a directory for persistent storage on a host machine, for example, a mount directory of a network file system (NFS).
- A PVC describes the attributes of the PV that a pod wants to use, such as the volume capacity and read/write permissions.

To allow a pod to use PVs, a Kubernetes cluster administrator needs to set the network storage class and provides the corresponding PV descriptors to Kubernetes. You only need to create a PVC and bind the PVC with the volumes in the pod so that you can store data. The following figure shows the relationship between PVs and PVCs.

Figure 3-28 Binding a PVC to a PV



CSI

Kubernetes Container Storage Interface (CSI) can be used to develop plug-ins to support specific storage volumes. For example, in the namespace named `kube-system`, as shown in [Namespaces: Grouping Resources](#), `everest-csi-controller-*` and `everest-csi-driver-*` are the storage controllers and drivers developed by CCE. With these drivers, you can use cloud storage services such as EVS, SFS, and OBS.

```
$ kubectl get po --namespace=kube-system
NAME                                READY  STATUS   RESTARTS  AGE
everest-csi-controller-6d796fb9c5-v22df  2/2    Running  0          9m11s
everest-csi-driver-snzrr                1/1    Running  0          12m
everest-csi-driver-ttj28                 1/1    Running  0          12m
everest-csi-driver-wtrk6                 1/1    Running  0          12m
```

PV

Each PV contains the specification and status of the volume. For example, a file system is created in SFS. The file system ID is `68e4a4fd-d759-444b-8265-20dc66c8c502`, and the mount point is `sfs-nas01.eu-west-0a.prod-cloud-ocb.orange-business.com:/share-96314776`. If you want to use this file system in CCE, you need to create a PV to describe the volume. The following is an example PV.

```
apiVersion: v1
kind: PersistentVolume
metadata:
  name: pv-example
spec:
  accessModes:
    - ReadWriteMany          # Read/write mode
  capacity:
    storage: 10Gi           # PV capacity
  csi:
    driver: nas.csi.everest.io # Driver to be used.
    fsType: nfs              # File system type
    volumeAttributes:
      everest.io/share-export-location: sfs-nas01.eu-west-0a.prod-cloud-ocb.orange-business.com:/share-96314776 # Mount point
      volumeHandle: 68e4a4fd-d759-444b-8265-20dc66c8c502 # File system ID
```

Fields under `csi` in the example above are exclusively used in CCE.

Next, create the PV.

```
$ kubectl create -f pv.yaml
persistentvolume/pv-example created

$ kubectl get pv
NAME          CAPACITY  ACCESS MODES  RECLAIM POLICY  STATUS   CLAIM  STORAGECLASS
REASON  AGE
pv-example    10Gi     RWX           Retain          Available  4s
```

RECLAIM POLICY indicates the PV reclaim policy. The value **Retain** indicates that the PV is retained after the PVC is released. If the value of **STATUS** is **Available**, the PV is available.

PVC

A PVC can be bound to a PV. The following is an example:

```
apiVersion: v1
kind: PersistentVolumeClaim
```

```

metadata:
  name: pvc-example
spec:
  accessModes:
  - ReadWriteMany
  resources:
    requests:
      storage: 10Gi          # Storage capacity
      volumeName: pv-example # PV name
  
```

Create the PVC.

```

$ kubectl create -f pvc.yaml
persistentvolumeclaim/pvc-example created

$ kubectl get pvc
NAME          STATUS  VOLUME      CAPACITY  ACCESS MODES  STORAGECLASS  AGE
pvc-example  Bound  pv-example  10Gi      RWX            9s
  
```

The command output shows that the PVC is in **Bound** state and the value of **VOLUME** is **pv-example**, indicating that the PVC is bound to a PV.

Now check the PV status.

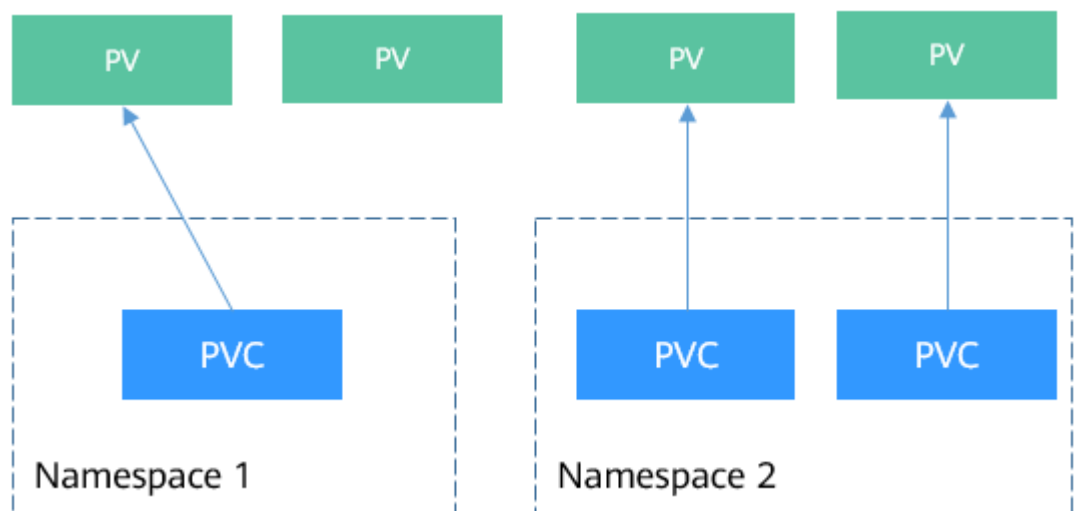
```

$ kubectl get pv
NAME          CAPACITY  ACCESS MODES  RECLAIM POLICY  STATUS  CLAIM              STORAGECLASS  REASON  AGE
pv-example    10Gi      RWX           Retain          Bound   default/pvc-example  50s
  
```

The status of the PVC is also **Bound**. The value of **CLAIM** is **default/pvc-example**, indicating that the PV is bound to the PVC named **pvc-example** in the default namespace.

Note that PVs are cluster-level resources and do not belong to any namespace, while PVCs are namespace-level resources. PVs can be bound to PVCs of any namespace. Therefore, the namespace name "default" is displayed under **CLAIM**.

Figure 3-29 Relationship between PVs and PVCs



StorageClass

Although PVs and PVCs allow you to consume abstract storage resources, you may need to configure multiple fields to create PVs and PVCs (such as the **csi** field

structure in the PV), and PVs/PVCs are generally managed by the cluster administrator, which can be inconvenient when you need PVs with varying attributes for different problems.

To solve this problem, Kubernetes supports dynamic PV provisioning to create PVs automatically. The cluster administrator can deploy a PV provisioner and define the corresponding StorageClass. In this way, developers can select the storage class to be created when creating a PVC. The PVC transfers the StorageClass to the PV provisioner, and the provisioner automatically creates a PV. In CCE, storage classes such as csi-disk, csi-nas, and csi-obs are supported. After StorageClassName is added to a PVC, PVs can be automatically provisioned and underlying storage resources can be automatically created.

 **NOTE**

The following examples are for CCE clusters of v1.15 and later, different from those for CCE clusters of v1.13 and earlier.

Run the following command to query the storage classes that CCE supports. You can use the CSI plug-ins provided by CCE to customize a storage class, which functions similarly as the default storage classes in CCE.

```
# kubectl get sc
NAME          PROVISIONER          AGE
csi-disk      everest-csi-provisioner 17d # Storage class for EVS disks
csi-disk-topology everest-csi-provisioner 17d # Storage class for EVS disks with delayed
association
csi-nas       everest-csi-provisioner 17d # Storage class for SFS file systems
csi-obs       everest-csi-provisioner 17d # Storage class for OBS buckets
csi-sfsturbo  everest-csi-provisioner 17d # Storage class for SFS Turbo file systems
```

Use **storageClassName** to create a PVC.

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: pvc-sfs-auto-example
spec:
  accessModes:
    - ReadWriteMany
  resources:
    requests:
      storage: 10Gi
  storageClassName: csi-nas # StorageClass
```

Create a PVC and view the PVC and PV details.

```
$ kubectl create -f pvc2.yaml
persistentvolumeclaim/pvc-sfs-auto-example created

$ kubectl get pvc
NAME          STATUS  VOLUME                                     CAPACITY  ACCESS MODES  STORAGECLASS  AGE
pvc-sfs-auto-example Bound   pvc-1f1c1812-f85f-41a6-a3b4-785d21063ff3 10Gi      RWX           csi-nas       29s

$ kubectl get pv
NAME          CAPACITY  ACCESS MODES  RECLAIM POLICY  STATUS
CLAIM        STORAGECLASS  REASON  AGE
pvc-1f1c1812-f85f-41a6-a3b4-785d21063ff3 10Gi      RWO          Delete          Bound   default/pvc-sfs-auto-example csi-nas 20s
```

The command output shows that after a StorageClass is used, a PVC and a PV are created and they are bound to each other.

After a StorageClass is set, PVs can be automatically created and maintained. Users only need to specify StorageClassName when creating a PVC, which greatly reduces the workload.

Note that the types of **storageClassName** vary among vendors. In this section, SFS is used as an example. For details about other storage classes, see [Storage Overview](#).

Using a PVC in a Pod

After a PVC is available, you can directly bind the PVC to a volume in the pod template and then mount the volume to the pod, as shown in the following example. You can also directly create a PVC in a StatefulSet. For details, see [StatefulSet](#).

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
spec:
  selector:
    matchLabels:
      app: nginx
  replicas: 2
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
        - image: nginx:alpine
          name: container-0
          volumeMounts:
            - mountPath: /tmp                # Mount path
              name: pvc-sfs-example
      restartPolicy: Always
      volumes:
        - name: pvc-sfs-example
          persistentVolumeClaim:
            claimName: pvc-example          # PVC name
```

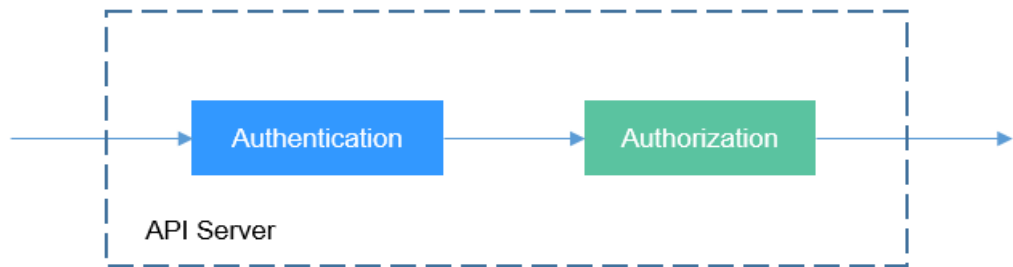
3.8 Authentication and Authorization

3.8.1 ServiceAccounts

All access requests to Kubernetes resources are processed by the API Server, regardless of whether the requests are from an external system. Therefore, the requests must be authenticated and authorized before they are sent to Kubernetes resources.

- **Authentication:** used for user identity authentication. Kubernetes uses different authentication mechanisms for external and internal service accounts. For details, see [Authentication and ServiceAccounts](#).
- **Authorization:** used for controlling users' access to resources. Currently, the role-based access control (RBAC) mechanism is used to authorize users to access resources. For details, see [RBAC](#).

Figure 3-30 Authentication and authorization of the API Server



Authentication and ServiceAccounts

Kubernetes users are classified into service accounts (ServiceAccounts) and common accounts.

- A ServiceAccount is bound to a namespace and associated with a set of credentials stored in a secret. When a pod is created, the secret is mounted to the pod so that the pod can be called by the API Server.
- Kubernetes does not have objects that represent common accounts. By default, these accounts are managed by external services. For example, CCE users are managed by Identity and Access Management (IAM).

The following only focuses on ServiceAccounts.

Similar to pods and ConfigMaps, ServiceAccounts are resources in Kubernetes and apply to independent namespaces. That is, a ServiceAccount named **default** is automatically created when a namespace is created.

Run the following command to view ServiceAccounts:

```
$ kubectl get sa
NAME      SECRETS  AGE
default  1        30d
```

In addition, Kubernetes automatically creates a secret for a ServiceAccount. Run the following command to view the secret:

```
$ 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>
```

In the pod definition file, you can assign a ServiceAccount to a pod by specifying an account name. If no account name is specified, the default ServiceAccount is used. When receiving a request with an authentication token, the API Server uses the token to check whether the ServiceAccount associated with the client that sends the request allows the request to be executed.

Creating a ServiceAccount

Run the following command to create a ServiceAccount:

```
$ kubectl create serviceaccount sa-example
serviceaccount/sa-example created

$ kubectl get sa
NAME          SECRETS  AGE
default      1        30d
sa-example   1        2s
```

The token associated with the ServiceAccount has been created.

```
$ kubectl describe sa sa-example
Name:          sa-example
Namespace:    default
Labels:       <none>
Annotations:  <none>
Image pull secrets: <none>
Mountable secrets: sa-example-token-c7bqx
Tokens:       sa-example-token-c7bqx
Events:       <none>
```

Check the secret content. You can find the **ca.crt**, **namespace**, and **token** data.

```
$ kubectl describe secret sa-example-token-c7bqx
Name:          sa-example-token-c7bqx
...
Data
====
ca.crt:       1082 bytes
namespace:    7 bytes
token:        <token content>
```

Using a ServiceAccount in a Pod

It is convenient to use a ServiceAccount in a pod. You only need to specify the name of the ServiceAccount.

```
apiVersion: v1
kind: Pod
metadata:
  name: sa-example
spec:
  serviceAccountName: sa-example
  containers:
  - image: nginx:alpine
    name: container-0
    resources:
      limits:
        cpu: 100m
        memory: 200Mi
      requests:
        cpu: 100m
        memory: 200Mi
    imagePullSecrets:
    - name: default-secret
```

Create a pod and view its information. You can see that **sa-example-token-c7bqx** is mounted to the pod, that is, the token corresponding to the ServiceAccount **sa-example**. That is, the pod uses the token for authentication.

```
$ kubectl create -f sa-pod.yaml
pod/sa-example created

$ kubectl get pod
NAME          READY  STATUS   RESTARTS  AGE
sa-example   0/1    running  0         5s

$ kubectl describe pod sa-example
...
```

```
Containers:
  sa-example:
    Mounts:
      /var/run/secrets/kubernetes.io/serviceaccount from sa-example-token-c7bqx (ro)
```

You can also view the corresponding file in the pod.

```
$ kubectl exec -it sa-example -- /bin/sh
/ # cd /run/secrets/kubernetes.io/serviceaccount
/run/secrets/kubernetes.io/serviceaccount # ls
ca.crt  namespace  token
```

As shown above, in a containerized application, **ca.crt** and **token** can be used to access the API Server.

Then check whether the authentication takes effect. In a Kubernetes cluster, a Service named **kubernetes** is created for the API Server by default. The API Server can be accessed through this service.

```
$ kubectl get svc
NAME         TYPE        CLUSTER-IP   EXTERNAL-IP  PORT(S)    AGE
kubernetes  ClusterIP   10.247.0.1   <none>       443/TCP    34
```

Go to the pod and run the **curl** command. If the following information is displayed, you do not have the permission.

```
$ kubectl exec -it sa-example -- /bin/sh
/ # curl https://kubernetes
curl: (60) SSL certificate problem: unable to get local issuer certificate
More details here: https://curl.haxx.se/docs/sslcerts.html
```

curl failed to verify the legitimacy of the server and therefore could not establish a secure connection to it. To learn more about this situation and how to fix it, please visit the web page mentioned above.

Use **ca.crt** and **token** for authentication. The specific procedure is as follows: Place **ca.crt** in the environment variable **CURL_CA_BUNDLE**, and run the **curl** command to specify the certificate using **CURL_CA_BUNDLE**. Place the token content in **TOKEN** and use the token to access the API Server.

```
# export CURL_CA_BUNDLE=/var/run/secrets/kubernetes.io/serviceaccount/ca.crt
# TOKEN=$(cat /var/run/secrets/kubernetes.io/serviceaccount/token)
# curl -H "Authorization: Bearer $TOKEN" https://kubernetes
{
  "kind": "Status",
  "apiVersion": "v1",
  "metadata": {
  },
  "status": "Failure",
  "message": "forbidden: User \"system:serviceaccount:default:sa-example\" cannot get path \"/\",",
  "reason": "Forbidden",
  "details": {
  },
  "code": 403
}
```

As shown above, the authentication is successful, but the API Server returns **cannot get path \"/\",** indicating that the API Server can be accessed only after being authorized. For details about the authorization mechanism, see [RBAC](#).

3.8.2 RBAC

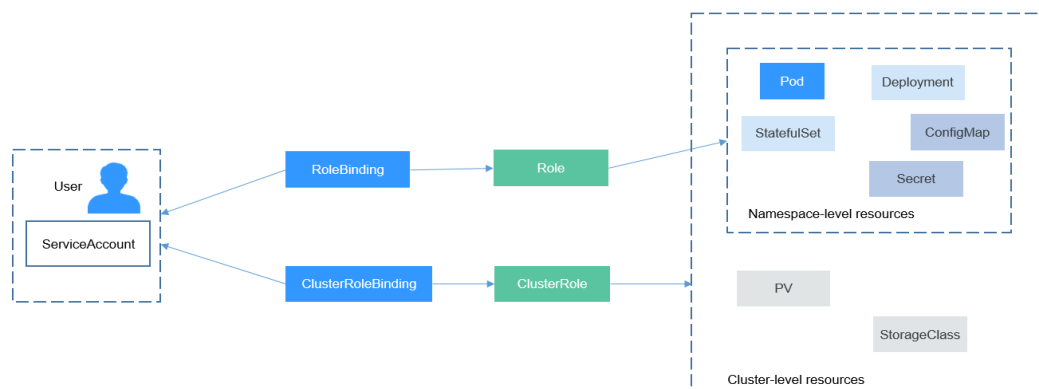
RBAC Resources

In Kubernetes, the RBAC mechanism is used for authorization. RBAC authorization uses four types of resources for configuration.

- 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](#). See the following figure.

Figure 3-31 Role binding



Creating a Role

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.
```

Creating a RoleBinding

After creating a Role, you can bind the Role to a specific user, which is called RoleBinding. The following is an example.

```
kind: RoleBinding
apiVersion: rbac.authorization.k8s.io/v1
metadata:
```

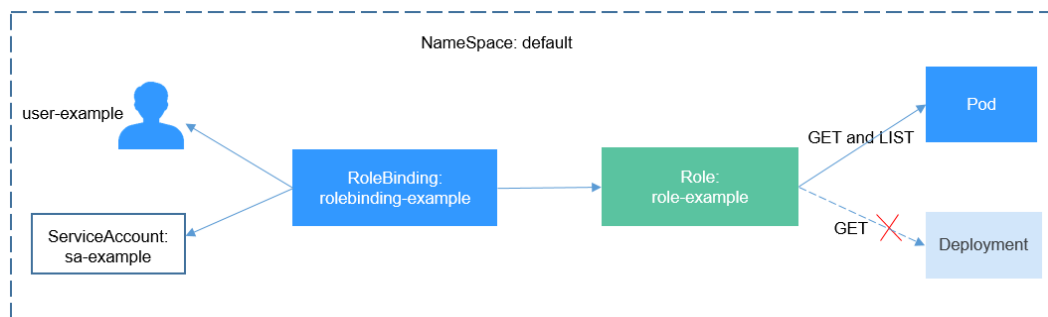
```

name: rolebinding-example
namespace: default
subjects:
- kind: User          # Specified user
  name: user-example  # Common user
  apiGroup: rbac.authorization.k8s.io
- kind: ServiceAccount # ServiceAccount
  name: sa-example
  namespace: default
roleRef:
  kind: Role
  name: role-example
  apiGroup: rbac.authorization.k8s.io

```

The **subjects** is used to bind the Role to a user. The user can be an external common user or a ServiceAccount. For details about the two user types, see [ServiceAccounts](#). The following figure shows the binding relationship.

Figure 3-32 A RoleBinding binds the Role to the user.



Then check whether the authorization takes effect.

In [Using a ServiceAccount](#), a pod is created and the ServiceAccount **sa-example** is used. The Role **role-example** is bound to **sa-example**. Access the pod and run the **curl** command to access resources through the API Server to check whether the permission takes effect.

Use **ca.crt** and **token** corresponding to **sa-example** for authentication and query all pod resources (**LIST** in [Creating a Role](#)) in the default namespace.

```

$ kubectl exec -it sa-example -- /bin/sh
# export CURL_CA_BUNDLE=/var/run/secrets/kubernetes.io/serviceaccount/ca.crt
# TOKEN=$(cat /var/run/secrets/kubernetes.io/serviceaccount/token)
# curl -H "Authorization: Bearer $TOKEN" https://kubernetes/api/v1/namespaces/default/pods
{
  "kind": "PodList",
  "apiVersion": "v1",
  "metadata": {
    "selfLink": "/api/v1/namespaces/default/pods",
    "resourceVersion": "10377013"
  },
  "items": [
    {
      "metadata": {
        "name": "sa-example",
        "namespace": "default",
        "selfLink": "/api/v1/namespaces/default/pods/sa-example",
        "uid": "c969fb72-ad72-4111-a9f1-0a8b148e4a3f",
        "resourceVersion": "10362903",
        "creationTimestamp": "2020-07-15T06:19:26Z"
      },
      "spec": {
...

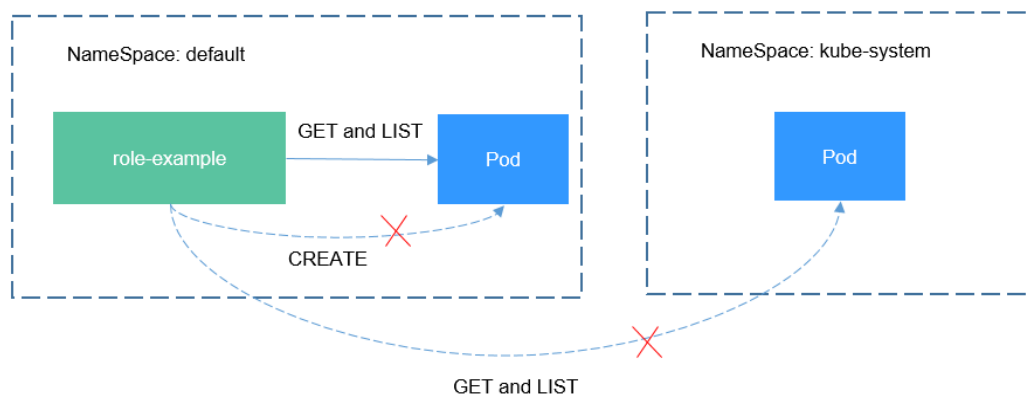
```

If the returned result is normal, **sa-example** has permission to list pods. Query the Deployment again. If the following information is displayed, you do not have the permission to access the Deployment.

```
# curl -H "Authorization: Bearer $TOKEN" https://kubernetes/api/v1/namespaces/default/deployments
...
"status": "Failure",
"message": "deployments is forbidden: User \"system:serviceaccount:default:sa-example\" cannot list resource \"deployments\" in API group \"\" in the namespace \"default\"",
...
```

Role and RoleBinding apply to namespaces and can isolate permissions to some extent. As shown in the following figure, **role-example** defined above cannot access resources in the **kube-system** namespace.

Figure 3-33 Role and RoleBinding applied to namespaces



Continue to access the pod. If the following information is displayed, you do not have the permission.

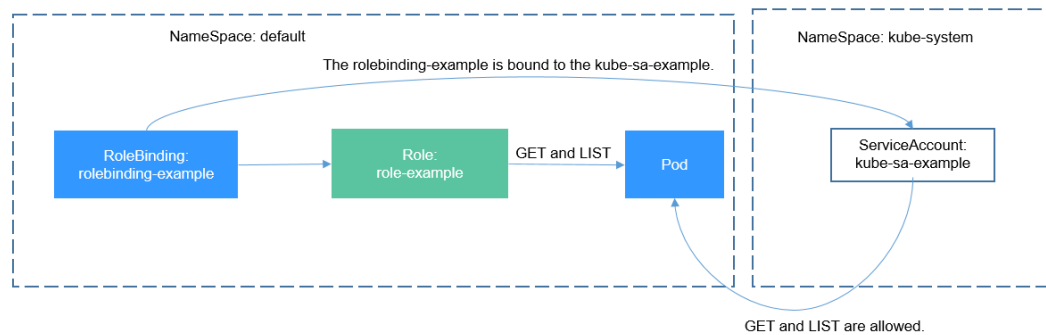
```
# curl -H "Authorization: Bearer $TOKEN" https://kubernetes/api/v1/namespaces/kube-system/pods
...
"status": "Failure",
"message": "pods is forbidden: User \"system:serviceaccount:default:sa-example\" cannot list resource \"pods\" in API group \"\" in the namespace \"kube-system\"",
"reason": "Forbidden",
...
```

In RoleBinding, you can also bind the ServiceAccounts of other namespaces by adding them under the **subjects** field.

```
subjects:
- kind: ServiceAccount          # Specified user
  name: kube-sa-example        # ServiceAccount
  namespace: kube-system
```

Then the ServiceAccount **kube-sa-example** in **kube-system** can perform GET and LIST operations on pods in the default namespace, as shown in the following figure.

Figure 3-34 Cross-namespace access



ClusterRole and ClusterRoleBinding

Compared with Role and RoleBinding, ClusterRole and ClusterRoleBinding have the following differences:

- ClusterRole and ClusterRoleBinding do not need to define the **namespace** field.
- ClusterRole can define cluster-level resources.

You can see that ClusterRole and ClusterRoleBinding control cluster-level permissions.

In Kubernetes, many ClusterRoles and ClusterRoleBindings are defined by default.

```
$ kubectl get clusterroles
NAME                                     AGE
admin                                    30d
cceaddon-prometheus-kube-state-metrics  6d3h
cluster-admin                            30d
coredns                                  30d
custom-metrics-resource-reader           6d3h
custom-metrics-server-resources         6d3h
edit                                      30d
prometheus                               6d3h
system:aggregate-customedhorizontalpodautoscalers-admin  6d2h
system:aggregate-customedhorizontalpodautoscalers-edit  6d2h
system:aggregate-customedhorizontalpodautoscalers-view  6d2h
....
view                                      30d

$ kubectl get clusterrolebindings
NAME                                     AGE
authenticated-access-network            30d
authenticated-packageversion            30d
auto-approve-csrs-for-group             30d
auto-approve-renewals-for-nodes         30d
auto-approve-renewals-for-nodes-server  30d
cceaddon-prometheus-kube-state-metrics  6d3h
cluster-admin                            30d
cluster-creator                          30d
coredns                                  30d
csrs-for-bootstrapping                  30d
system:basic-user                        30d
system:ccehpa-rolebinding                6d2h
system:cluster-autoscaler                6d1h
...
```

The most important and commonly used ClusterRoles are as follows:

- **view**: has the permission to view namespace resources.

- `edit`: has the permission to modify namespace resources.
- `admin`: has all permissions on the namespace.
- `cluster-admin`: has all permissions on the cluster.

Run the `kubectl describe clusterrole` command to view the permissions of each rule.

Generally, the four ClusterRoles are bound to users to isolate permissions. Note that Roles (rules and permissions) are separated from users. You can flexibly control permissions by combining the two through RoleBinding.

3.9 Auto Scaling

In [Pod Orchestration and Scheduling](#), we introduce controllers such as Deployment to control the number of pod replicas. You can adjust the number of replicas to manually scale your applications. However, manual scaling is sometimes complex and fails to cope with unexpected traffic spikes.

Kubernetes supports auto scaling of pods and cluster nodes. You can set rules to trigger auto scaling when certain metrics (such as CPU usage) reach the configured threshold.

Prometheus and Metrics Server

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.

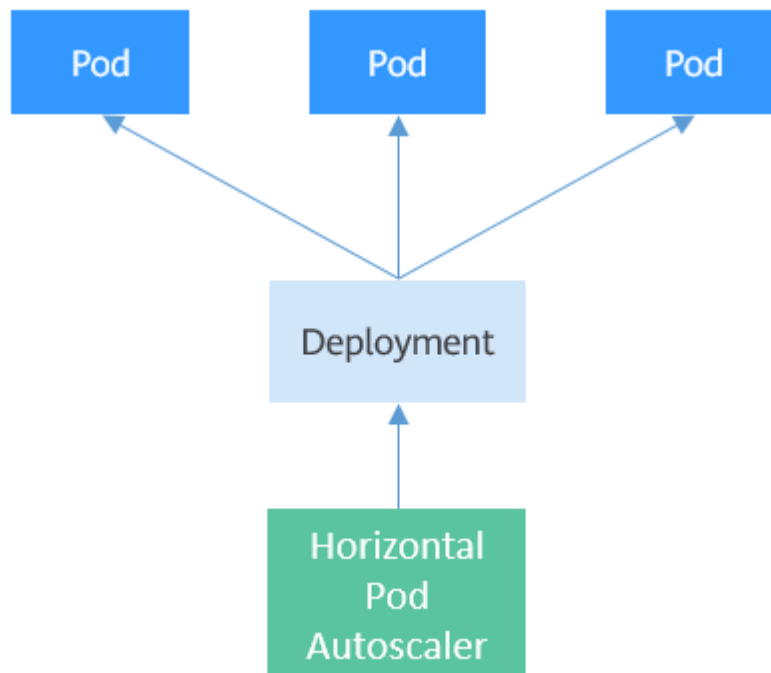
- **Prometheus** is an open-source monitoring and alarming framework that can collect multiple types of metrics. Prometheus has been a standard monitoring solution of Kubernetes.
- **Metrics Server** is a cluster-wide aggregator of resource utilization data. Metrics Server collects metrics from the Summary 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.

Horizontal Pod Autoscaler (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.

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).

Figure 3-35 HPA working mechanism



You can configure one or more metrics for the HPA. When configuring a single metric, you only need to sum up the current pod metrics, divide the sum by the expected target value, and then round up the result to obtain the expected number of replicas. For example, if a Deployment controls three pods, the CPU usage of each pod is 70%, 50%, and 90%, and the expected CPU usage configured in the HPA is 50%, the expected number of replicas is calculated as follows: $(70 + 50 + 90)/50 = 4.2$. The result is rounded up to 5. That is, the expected number of replicas is 5.

If multiple metrics are configured, the expected number of replicas of each metric is calculated and the maximum value will be used.

Using the HPA

The following example demonstrates how to use the HPA. First, use the Nginx image to create a Deployment with four replicas.

```

$ kubectl get deploy
NAME          READY   UP-TO-DATE   AVAILABLE   AGE
nginx-deployment 4/4     4            4           77s

$ kubectl get pods
NAME                                READY   STATUS    RESTARTS   AGE
nginx-deployment-7cc6fd654c-5xztl  1/1     Running   0          82s
nginx-deployment-7cc6fd654c-cwjzg  1/1     Running   0          82s
nginx-deployment-7cc6fd654c-dffkp  1/1     Running   0          82s
nginx-deployment-7cc6fd654c-j7mp8  1/1     Running   0          82s
  
```

Create an HPA. The expected CPU usage is 70% and the number of replicas ranges from 1 to 10.

```

apiVersion: autoscaling/v2beta1
kind: HorizontalPodAutoscaler
metadata:
  
```

```

name: scale
namespace: default
spec:
  maxReplicas: 10          # Maximum number of replicas of the target resource
  minReplicas: 1          # Minimum number of replicas of the target resource
  metrics:                 # Metric. The expected CPU usage is 70%.
  - resource:
    name: cpu
    targetAverageUtilization: 70
    type: Resource
  scaleTargetRef:         # Target resource
  apiVersion: apps/v1
  kind: Deployment
  name: nginx-deployment

```

Query the created HPA.

```

$ kubectl create -f hpa.yaml
horizontalpodautoscaler.autoscaling/celue created

$ kubectl get hpa
NAME      REFERENCE                TARGETS  MINPODS  MAXPODS  REPLICAS  AGE
scale    Deployment/nginx-deployment  0%/70%   1        10       4         18s

```

In the command output, the expected value of **TARGETS** is **70%**, but the actual value is **0%**. This means that the HPA will perform scale-in. The expected number of replicas can be calculated as follows: $(0 + 0 + 0 + 0)/70 = 0$. However, the minimum number of replicas has been set to **1**. Therefore, the number of pods is changed to 1. After a while, the number of pods changes to 1.

```

$ kubectl get pods
NAME                                READY  STATUS   RESTARTS  AGE
nginx-deployment-7cc6fd654c-5xzlz  1/1    Running  0         7m41s

```

Query the HPA again and a record similar to the following is displayed under **Events**. In this example, the record indicates that the HPA successfully performed a scale-in 21 seconds ago and the number of pods is changed to 1, and the scale-in is triggered because the values of all metrics are lower than the target values.

```

$ kubectl describe hpa scale
...
Events:
  Type     Reason             Age   From              Message
  ----     -
  Normal   SuccessfulRescale  21s   horizontal-pod-autoscaler  New size: 1; reason: All metrics below target

```

If you want to query the Deployment details, you can check the records similar to the following under **Events**. In this example, the second record indicates that the number of replicas of the Deployment is set to **1**, which is the same as that in the HPA.

```

$ kubectl describe deploy nginx-deployment
...
Events:
  Type     Reason             Age   From              Message
  ----     -
  Normal   ScalingReplicaSet  7m    deployment-controller  Scaled up replica set nginx-deployment-7cc6fd654c to 4
  Normal   ScalingReplicaSet  1m    deployment-controller  Scaled down replica set nginx-deployment-7cc6fd654c to 1

```

Cluster AutoScaler

The 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.

Cluster Autoscaler is a component provided by Kubernetes for auto scaling of cluster nodes based on the pod scheduling status and resource usage. You can refer to the API documentation of your cloud service provider to implement auto scaling.

For details about the implementation on CCE, see [Creating a Node Scaling Policy](#).

4 Getting Started

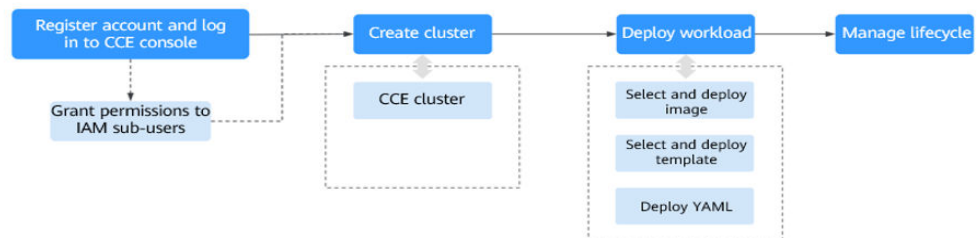
4.1 Introduction

This section describes how to use Cloud Container Engine (CCE) and provides frequently asked questions (FAQs) to help you quickly get started with CCE.

Procedure

Complete the following tasks to get started with CCE.

Figure 4-1 Procedure for getting started with CCE



Step 1 Register an account and grant permissions to IAM users.

An account has the permissions to use CCE. However, IAM users created by an account do not have the permission. You need to manually grant the permission to IAM users. For details, see .

Step 2 Create a cluster.

For details on how to create a regular Kubernetes cluster, see [Creating a Kubernetes Cluster](#).

Step 3 Create a workload from an image or chart.

- [Creating a Deployment \(Nginx\) from an Image](#)
- [Deploying WordPress and MySQL That Depend on Each Other](#)

Step 4 View workload status and logs. Upgrade, scale, and monitor the workload.

----End

FAQs

1. **Is CCE suitable for users who are not familiar with Kubernetes?**

Yes. The CCE console is easy-to-use, and the *Getting Started* guide helps you quickly understand and use CCE.

2. **Is CCE suitable for users who have little experience in building images?**

In addition to storing images created by yourself in **My Images**, CCE allows you to create containerized applications using open source images. For details, see [Creating a Deployment \(Nginx\) from an Image](#).

3. **How do I create a workload using CCE?**

Create a cluster and then create a workload in the cluster. For details, see [Creating a Deployment \(Nginx\) from an Image](#).

4. **How do I create a workload accessible to public networks?**

CCE provides different workload access types to address diverse scenarios.

5. **How can I allow multiple workloads in the same cluster to access each other?**

Set **Service Type** to **ClusterIP**, which allows workloads in the same cluster to use their cluster-internal domain names to access each other.

Cluster-internal domain names are in the format of <self-defined service name>.<workload's namespace>.svc.cluster.local:<port number>. For example, nginx.default.svc.cluster.local:80.

4.2 Preparations

Before using CCE, you need to make the following preparations:

- [Creating an IAM user](#)
- [Obtaining Resource Permissions](#)
- [\(Optional\) Creating a VPC](#)
- [\(Optional\) Creating a Key Pair](#)

Creating an IAM user

If you want to allow multiple users to manage your resources without sharing your password or keys, you can create users using IAM and grant permissions to the users. These users can use specified links and their own accounts to access the cloud and manage resources efficiently. You can also configure account security policies to ensure the security of these accounts.

Your accounts have the permissions to use CCE. However, IAM users created by your accounts do not have the permissions. You need to manually assign the permissions to IAM users.

Obtaining Resource Permissions

CCE works closely with multiple cloud services to support computing, storage, networking, and monitoring functions. When you log in to the CCE console for the

first time, CCE automatically requests permissions to access those cloud services in the region where you run your applications. Specifically:

- **Compute services**
When you create a node in a cluster, a cloud server is created accordingly. The prerequisite is that CCE has obtained the permissions to access Elastic Cloud Service (ECS) and Bare Metal Server (BMS).
- **Storage services**
CCE allows you to mount storage to nodes and containers in a cluster. The prerequisite is that CCE has obtained the permissions to access services such as Elastic Volume Service (EVS), Scalable File Service (SFS), and Object Storage Service (OBS).
- **Networking services**
CCE allows containers in a cluster to be published as services that can be accessed by external systems. The prerequisite is that CCE has obtained the permissions to access services such as Virtual Private Cloud (VPC) and Elastic Load Balance (ELB).
- **Container and monitoring services**
CCE supports functions such as container image pull, monitoring, and logging. The prerequisite is that CCE has obtained the permissions to access services such as SoftWare Repository for Container (SWR) and Application Operations Management (AOM).

After you agree to delegate the permissions, an agency named **cce_admin_trust** will be created for CCE in Identity and Access Management (IAM). The system account **op_svc_cce** will be delegated the **Tenant Administrator** role to perform operations on other cloud service resources. Tenant Administrator has the permissions on all cloud services except IAM, which calls the cloud services on which CCE depends. The delegation takes effect only in the current region.

To use CCE in multiple regions, you need to request cloud resource permissions in each region. You can go to the IAM console, choose **Agencies**, and click **cce_admin_trust** to view the delegation records of each region.

NOTE

CCE may fail to run as expected if the Tenant Administrator role is not assigned. Therefore, do not delete or modify the **cce_admin_trust** agency when using CCE.

(Optional) Creating a VPC

A VPC provides an isolated, configurable, and manageable virtual network for CCE clusters.

Before creating the first cluster, ensure that a VPC has been created.

If you already have a VPC available, skip this step.

Step 1 Log in to the management console.

Step 2 Click  in the upper left corner and select a region and a project.

Step 3 Under **Networking**, click **Virtual Private Cloud**.

Step 4 Click **Create VPC**.

Step 5 On the **Create VPC** page, configure parameters as prompted.

A default subnet will be created together with a VPC. You can click **Add Subnet** to create more subnets for the VPC.

Step 6 Click **Create Now**.

----End

(Optional) Creating a Key Pair

The cloud platform uses public key cryptography to protect the login information of your CCE nodes. Passwords or key pairs are used for identity authentication during remote login to nodes.

- You need to specify the key pair name and provide the private key when logging to CCE nodes using SSH if you choose the key pair login mode.
- If you choose the password login mode, skip this task.

NOTE

If you want to create pods in multiple regions, you need to create a key pair in each region.

Creating a Key Pair on the Management Console

If you have no key pair, create one on the management console. The procedure is as follows:

Step 1 Log in to the management console.

Step 2 Click  in the upper left corner and select a region and a project.

Step 3 Under **Computing**, click **Elastic Cloud Server**.

Step 4 In the navigation pane on the left, choose **Key Pair**.

Step 5 On the right pane, click **Create Key Pair**.

Step 6 Enter the key name and click **OK**.

Step 7 A key pair name consists of two parts: **KeyPair** and four random digits. You can enter an easy-to-remember name, for example, **KeyPair-xxxx_ecs**.

Step 8 Manually or automatically download the private key file. The file name is a specified key pair name with a suffix of .pem. Securely store the private key file. In the dialog box displayed, click **OK**.

NOTE

The private key file can be downloaded only once. Keep it secure. When creating an ECS, provide the name of your desired key pair. Each time you log in to the ECS using SSH, provide the private key.

----End

4.3 Creating a Kubernetes Cluster

Context

This section describes how to quickly create a CCE cluster. In this example, the default or simple configurations are in use.

Creating a Cluster

Step 1 Log in to the CCE console.

- If you have not created a cluster, a wizard page is displayed. Click **Create** under **CCE cluster**.
- If you have created a cluster, choose **Clusters** from the navigation pane and click **Create**.

Step 2 On the **Configure Cluster** page, configure cluster parameters.

In this example, a majority of parameters retain default values. Only mandatory parameters are described. For details, see [Table 4-1](#).

Table 4-1 Parameters for creating a cluster

Parameter	Description
Basic Settings	
*Cluster Name	Name of the cluster to be created. A cluster name contains 4 to 128 characters starting with a lowercase letter and not ending with a hyphen (-). Only lowercase letters, digits, and hyphens (-) are allowed.
* Enterprise Project	This parameter is displayed only for enterprise users who have enabled Enterprise Project Management.
* Cluster Version	Choose the latest version.
* Cluster Scale	Maximum number of worker nodes that can be managed by the cluster. If you select 50 nodes , the cluster can manage a maximum of 50 worker nodes.
*High Availability	The default value is Yes .
Network Settings	
* Network Model	Retain the default settings.
*VPC	VPC where the cluster will be located. If no VPC is available, click Create VPC to create one. After the VPC is created, click refresh.

Parameter	Description
* Master Node Subnet	Subnet where master nodes of the cluster are located.
* Container CIDR Block	Retain the default value.
* IPv4 Service CIDR Block	CIDR block for Services used by 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 creation.

Step 3 Click **Next: Add-on Configuration**. Retain the default settings.

Step 4 Click **Next: Confirm**. After confirming that the cluster configuration information is correct, select **I have read and understand the preceding instructions** and click **Submit**.

It takes about 6 to 10 minutes to create a cluster.

The created cluster will be displayed on the **Clusters** page, and the number of nodes in the cluster is 0.

----End

Creating a Node

After a cluster is created, you need to create nodes in the cluster to run workloads.

Step 1 Log in to the CCE console.

Step 2 Click the created cluster.

Step 3 In the navigation pane, choose **Nodes**. Click **Create Node** in the upper right corner and set node parameters.

The following describes only important parameters. For other parameters, retain the defaults.

Compute Settings

- **AZ:** Retain the default value.
- **Node Type:** Select **Elastic Cloud Server (VM)**.
- **Specifications:** Select node specifications that fit your business needs.
- **Container Engine:** Select a container engine as required.
- **OS:** Select the operating system (OS) of the nodes to be created.
- **Node Name:** Enter a node name.
- **Login Mode:**
 - If the login mode is **Key pair**, select a key pair for logging to the node and select the check box to acknowledge that you have obtained the key file and 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 Key Pair**.

Storage Settings

- **System Disk:** Configure disk type and capacity based on your requirements. The default disk capacity is 50 GiB.
- **Data Disk:** Configure disk type and capacity based on your requirements. The default disk capacity is 100 GiB.

Network Settings

- **VPC:** Use the default value, that is, the subnet selected during cluster creation.
- **Node Subnet:** Select a subnet in which the node runs.
- **Node IP:** IP address of the specified node.
- **EIP:** The default value is **Do not use**. You can select **Use existing** and **Auto create**.

Step 4 At the bottom of the page, select the node quantity, and click **Next: Confirm**.

Step 5 Review the node specifications, read the instructions, select **I have read and understand the preceding instructions**, and click **Submit**.

It takes about 6 to 10 minutes to add a node.

The created node will be displayed on the **Nodes** page.

----End

4.4 Creating a Deployment (Nginx) from an Image

You can use images to quickly create a single-pod workload that can be accessed from public networks. This section describes how to use CCE to quickly deploy an Nginx application and manage its life cycle.

Prerequisites

- The Nginx image has been pushed to SWR.
- You have created a CCE cluster that contains a node with 4 vCPUs and 8 GB memory. The node has an elastic IP address (EIP).
- A cluster is a logical group of cloud servers that run workloads. Each cloud server is a node in the cluster.
- For details on how to create a cluster, see [Creating a Kubernetes Cluster](#).

Nginx Overview

Nginx is a lightweight web server. On CCE, you can quickly set up a Nginx web server.

This section uses the Nginx application as an example to describe how to create a workload. The creation takes about 5 minutes.

After Nginx is created successfully, you can access the Nginx web page.

Figure 4-2 Nginx web page

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.
Commercial support is available at nginx.com.

Thank you for using nginx.

Procedure

The following is the procedure for creating a containerized workload from a container image.

- Step 1** Log in to the CCE console.
- Step 2** Choose the target cluster.
- Step 3** In the navigation pane, choose **Workloads**. Then, click **Create Workload**.
- Step 4** Configure the following parameters and retain the default value for other parameters:

Basic Info

- **Workload Type:** Select **Deployment**.
- **Workload Name:** Set it to **nginx**.
- **Namespace:** Select **default**.
- **Pods:** Set the quantity of pods to **1**.

Container Settings

In the **Basic Info** area, click **Select Image**. In the dialog box displayed, select **Open Source Images**, search for **nginx**, and select the **nginx** image.

Service Settings

Click the plus sign (+) to create a Service for accessing the workload from an external network. In this example, create a LoadBalancer Service. Set the following parameters:

- **Service Name:** name of the Service exposed to external networks. In this example, the Service name is **nginx**.
- **Service Type:** Select **LoadBalancer**.
- **Service Affinity:** Retain the default value.
- **Load Balancer:** If a load balancer is available, select an existing load balancer. If not, choose **Auto create** to create one on the ELB console.
- **Port:**
 - **Protocol:** Select **TCP**.

- **Service Port:** Set this parameter to **8080**, which is mapped to the container port.
- **Container Port:** port on which the application listens. For containers created using the Nginx image, set this parameter to **80**. For other applications, set this parameter to the port of the application.

Step 5 Click **Create Workload**.

Wait until the workload is created.

The created Deployment will be displayed on the **Deployments** page.

----End

Accessing Nginx

Step 1 Obtain the external access address of Nginx.

Click the Nginx workload to enter its details page. On the **Access Mode** tab page, you can view the IP address of Nginx. The public IP address is the external access address.

Step 2 Enter the **external access address** in the address box of a browser. The following shows the welcome page if you successfully access the workload.

Figure 4-3 Accessing Nginx



----End

4.5 Deploying WordPress and MySQL That Depend on Each Other

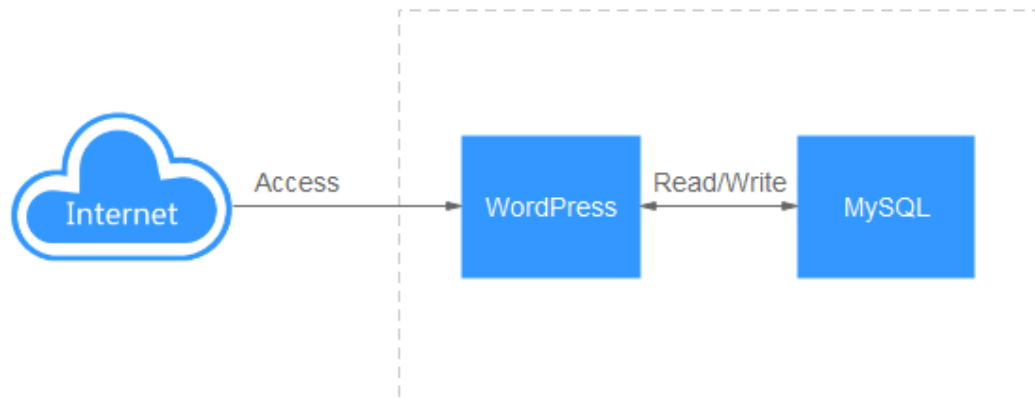
4.5.1 Overview

WordPress was originally a blog platform based on PHP and MySQL. It is gradually evolved into a content management system. You can set up your own blog website on any server that supports PHP and MySQL. Thousands of plug-ins and countless theme templates are available for WordPress and easy to install.

WordPress is a blog platform developed in hypertext preprocessor (PHP). You can set up your websites on the services that support PHP and MySQL databases, or use WordPress as a content management system. For more information about WordPress, visit <https://wordpress.org/>.

WordPress must be used together with MySQL. WordPress runs the content management program while MySQL serves as a database to store data. Generally, WordPress and MySQL run in different containers, as shown in the following figure.

Figure 4-4 WordPress



In this example, two container images are involved.

- **WordPress:** Select `wordpress:php7.3` in this example.
- **MySQL:** Select `mysql:5.7` in this example.

When WordPress accesses MySQL in a cluster, Kubernetes provides a resource object called Service for the workload access. In this example, a Service is created for MySQL and WordPress, respectively. For details about how to create and configure a Service, see the following sections.

4.5.2 Step 1: Create a MySQL Workload

WordPress must be used together with MySQL. WordPress runs the content management program while MySQL serves as a database to store data.

Prerequisites

- The WordPress and MySQL images have been pushed to SWR.
- You have created a CCE cluster that contains a node with 4 vCPUs and 8 GB memory. For details on how to create a cluster, see [Creating a Kubernetes Cluster](#).

Creating a MySQL Workload

Step 1 Log in to the CCE console.

Step 2 Choose the target cluster.

Step 3 In the navigation pane, choose **Workloads**. Then, click **Create Workload**.

Step 4 Set workload parameters.

Basic Info

- **Workload Type:** Select **Deployment**.

- **Workload Name:** Set it to **mysql**.
- **Namespace:** Select **default**.
- **Pods:** Change the value to **1** in this example.

Container Settings

In the **Basic Info** area, click **Select Image**. In the dialog box displayed, select **Open Source Images**, search for **mysql**, and select the **mysql** image.

Select **5.7** for **Image Version**.

Add the following four environment variables (details available in [MySQL](#)):

- **MYSQL_ROOT_PASSWORD:** password of the **root** user of MySQL.
- **MYSQL_DATABASE:** name of the database created during image startup.
- **MYSQL_USER:** name of the database user.
- **MYSQL_PASSWORD:** password of the database user.

Service Settings

Click the plus sign (+) to create a Service for accessing MySQL from WordPress.

Select **ClusterIP** for **Service Type**, set **Service Name** to **mysql**, set both the **Container Port** and **Service Port** to **3306**, and click **OK**.

The default access port in the MySQL image is 3306. In this example, both the container port and Service port are set to **3306** for convenience. The access port can be changed to another port.

In this way, the MySQL workload can be accessed through **Service name:Access port (mysql:3306)** in this example) from within the cluster.

Step 5 Click **Create Workload**.

Wait until the workload is created.

The created Deployment will be displayed on the **Deployments** page.

----End

4.5.3 Step 2: Create a WordPress Workload

WordPress was originally a blog platform based on PHP and MySQL. It is gradually evolved into a content management system. You can set up your own blog website on any server that supports PHP and MySQL. Thousands of plug-ins and countless theme templates are available for WordPress and easy to install.

This section describes how to create a public WordPress website from images.

Prerequisites

- The WordPress and MySQL images have been pushed to SWR.
- You have created a CCE cluster that contains a node with 4 vCPUs and 8 GB memory. For details on how to create a cluster, see [Creating a Kubernetes Cluster](#).

- The MySQL database has been created by following the instructions in [Step 1: Create a MySQL Workload](#). In this example, WordPress data is stored in the MySQL database.

Creating a WordPress Blog Website

Step 1 Log in to the CCE console.

Step 2 Choose the target cluster.

Step 3 In the navigation pane, choose **Workloads**. Then, click **Create Workload**.

Step 4 Set workload parameters.

Basic Info

- **Workload Type:** Select **Deployment**.
- **Workload Name:** Set it to **wordpress**.
- **Namespace:** Select **default**.
- **Pods:** Set this parameter to **2** in this example.

Container Settings

In the **Basic Info** area, click **Select Image**. In the dialog box displayed, select **Open Source Images**, search for **wordpress**, and select the **wordpress** image.

Select **php7.3** for **Image Version**.

Add the following environment variables:

(These variables let WordPress know the information about the MySQL database.)

- **WORDPRESS_DB_HOST:** address for accessing the database, which can be found in the Service (on the **Services** tab page) of the MySQL workload. You can use the internal domain name **mysql.default.svc.cluster.local:3306** to access the database, or use only **mysql:3306** omitting **.default.svc.cluster.local**.
- **WORDPRESS_DB_USER:** username for accessing the database. The value must be the same as that of **MYSQL_USER** in [Step 1: Create a MySQL Workload](#), which is used to connect to MySQL.
- **WORDPRESS_DB_PASSWORD:** password for accessing the database. The value must be the same as that of **MYSQL_PASSWORD** in [Step 1: Create a MySQL Workload](#).
- **WORDPRESS_DB_NAME:** name of the database to be accessed. The value must be the same as that of **MYSQL_DATABASE** in [Step 1: Create a MySQL Workload](#).

Service Settings

Click the plus sign (+) to create a Service for accessing the workload from an external network. In this example, create a LoadBalancer Service. Set the following parameters:

- **Service Name:** name of the Service exposed to external networks. In this example, the Service name is **wordpress**.
- **Service Type:** Select **LoadBalancer**.

- **Service Affinity:** Retain the default value.
- **Load Balancer:** If a load balancer is available, select an existing load balancer. If not, choose **Auto create** to create one on the ELB console.
- **Port:**
 - **Protocol:** Select **TCP**.
 - **Service Port:** Set this parameter to **80**, which is mapped to the container port.
 - **Container Port:** port on which the application listens. For containers created using the wordpress image, set this parameter to **80**. For other applications, set this parameter to the port of the application.

Step 5 Click **Create Workload**.

Wait until the workload is created.

The created Deployment will be displayed on the **Deployments** page.

----End

Accessing WordPress

Step 1 Obtain the external access address of WordPress.

Click the wordpress workload to enter its details page. On the **Access Mode** tab page, you can view the IP address of WordPress. The load balancer IP address is the external access address.

Step 2 Enter the external access address in the address box of a browser to connect to the workload.

The following figure shows the accessed WordPress page.

Figure 4-5 WordPress-1

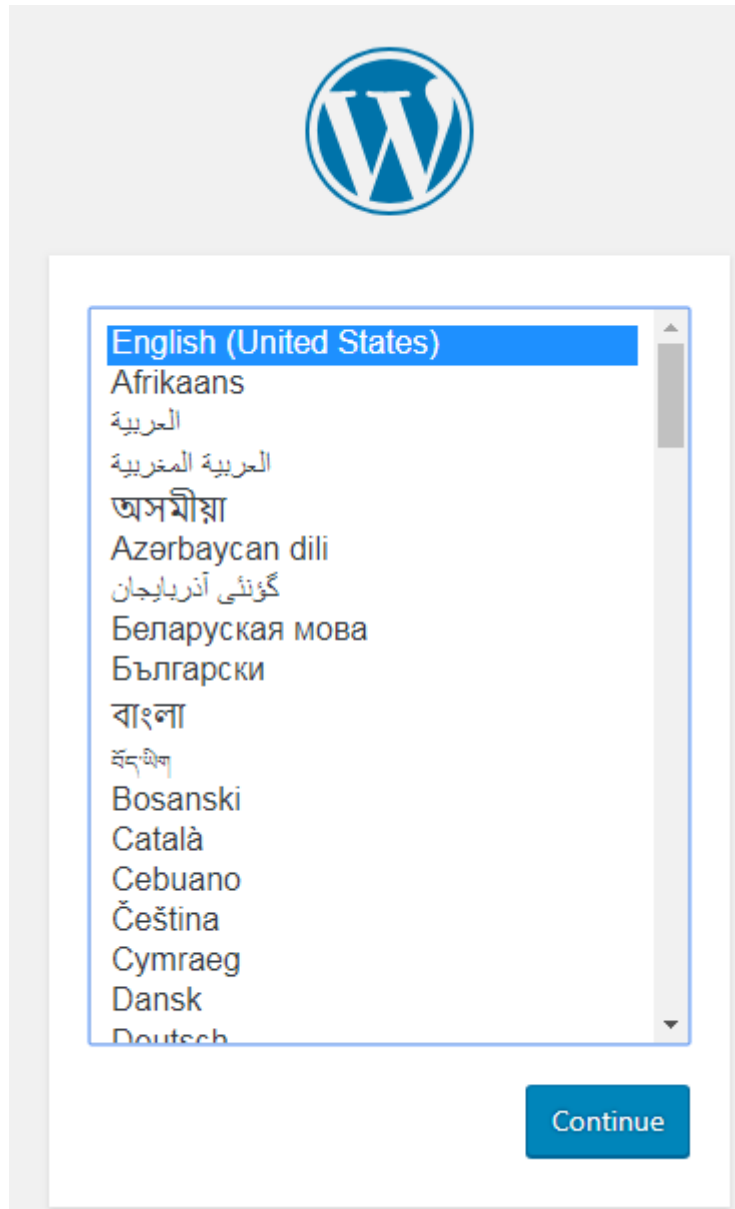
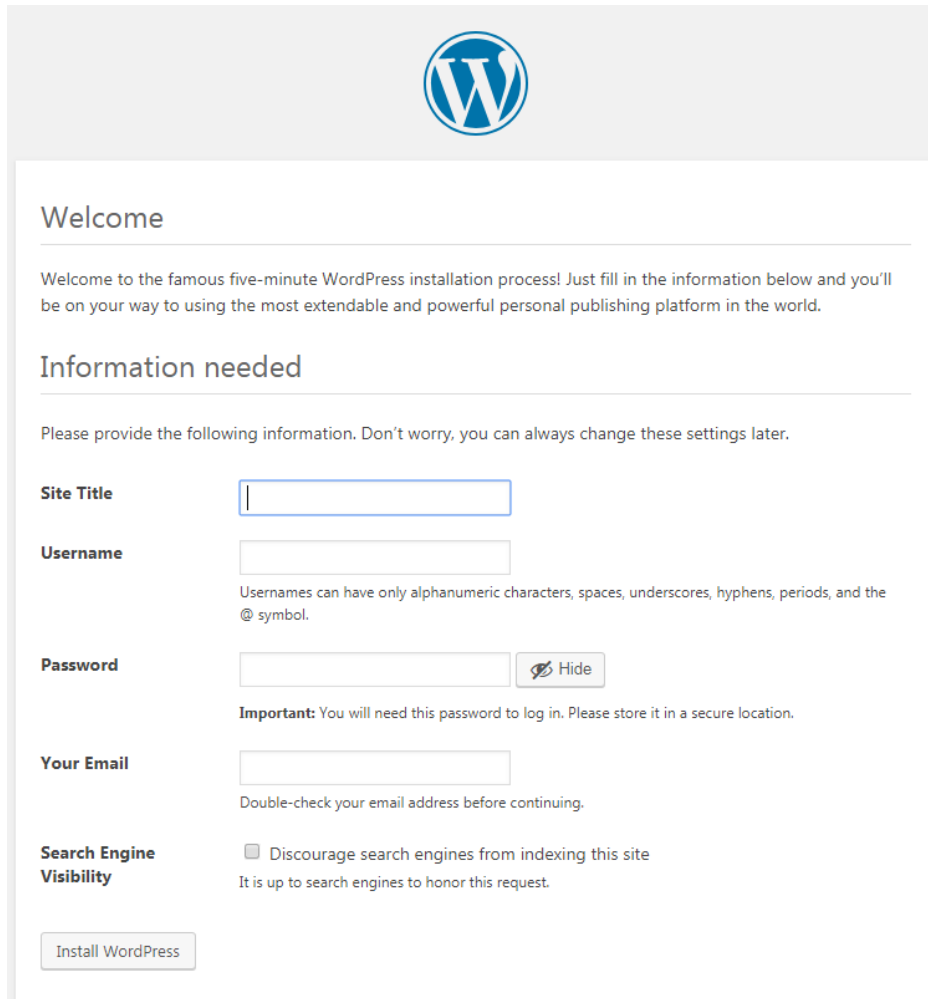


Figure 4-6 WordPress-2




The screenshot shows the WordPress installation 'Information needed' form. At the top is the WordPress logo. Below it is a 'Welcome' section with a message: 'Welcome to the famous five-minute WordPress installation process! Just fill in the information below and you'll be on your way to using the most extendable and powerful personal publishing platform in the world.' The 'Information needed' section follows, with a note: 'Please provide the following information. Don't worry, you can always change these settings later.' The form contains several fields: 'Site Title' (text input), 'Username' (text input with a note: 'Usernames can have only alphanumeric characters, spaces, underscores, hyphens, periods, and the @ symbol.'), 'Password' (password input with a 'Hide' button and an 'Important' note: 'You will need this password to log in. Please store it in a secure location.'), 'Your Email' (text input with a note: 'Double-check your email address before continuing.'), and 'Search Engine Visibility' (checkbox for 'Discourage search engines from indexing this site' with a note: 'It is up to search engines to honor this request.'). At the bottom is an 'Install WordPress' button.

----End

Deleting Resources

Until now, you have completed all the Getting Started walkthroughs and have understood how to use CCE. Fees are incurred while nodes are running. If the clusters used in the Getting Started walkthroughs are no longer in use, perform the following steps to delete them. If you will continue the CCE walkthroughs, retain the clusters.

- Step 1** Log in to the CCE console.
- Step 2** In the navigation pane on the left, choose **Clusters**.
- Step 3** Click  next to a cluster and delete the cluster as prompted.

----End

5 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 5-1 High-risk operations and solutions

Category	Operation	Impact	Solution
Master node	Modifying the security group of a node in a cluster	The master node may be unavailable. NOTE Naming rule of a master node: <i>Cluster name-cce-control-Random number</i>	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 /etc/kubernetes 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 Cluster Configuration Management .
	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	The node may be unavailable. NOTE Naming rule of a worker node: <i>Cluster name-cce-node-Random number</i>	Restore the security group 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 Resetting a Node .
	Upgrading the node kernel	The node may be unavailable or the network may be abnormal. NOTE Node running depends on the system kernel version. Do not use the yum update 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 Resetting a Node .

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 Configuring a Node Pool .
	Modifying OS configuration	The node may be unavailable.	Restore the configuration items or reset the node. For details, see Resetting a Node .
	Deleting or modifying the <code>/opt/cloud/cce</code> and <code>/var/paas</code> directories, and deleting the data disk	The node will become unready.	Reset the node. For details, see Resetting a Node .
	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 system disks, Docker disks, and kubelet disks on nodes.	The node may be unavailable.	Reset the node. For details, see Resetting a Node .
	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 Resetting a Node .
	Modifying NetworkManager configurations	The node will become unavailable.	Reset the node. For details, see Resetting a Node .

Category	Operation	Impact	Solution
	Delete system images such as cce-pause from the node.	Containers cannot be created and system images cannot be pulled.	Copy the image from another normal node for restoration.

Networking

Table 5-2 High-risk operations and solutions

Operation	Impact	Solution
Changing the value of the kernel parameter net.ipv4.ip_forward to 0	The network becomes inaccessible.	Change the value to 1 .
Changing the value of the kernel parameter net.ipv4.tcp_tw_recycle to 1	The NAT service becomes abnormal.	Change the value to 0 .
Changing the value of the kernel parameter net.ipv4.tcp_tw_reuse to 1	The network becomes abnormal.	Change the value to 0 .
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 Creating a Cluster and allow traffic from the security group to pass through.
Delete 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.

Load Balancing

Table 5-3 Service ELB

Operation	Impact	Solution
Changing the private IPv4 address of a load balancer on the ELB console	<ul style="list-style-type: none"> The network traffic forwarded using the private IPv4 addresses will be interrupted. The IP address in the status field of the Service/ingress YAML file is changed. 	You are not advised to modify the permissions. Restore the permissions if they are modified.
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 through a Service or an ingress. 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"> Service/Ingress access fails. After the master nodes are restarted, for example, due to a cluster upgrade, all your modifications will be reset by CCE. 	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 the master nodes are restarted, for example, due to a cluster upgrade, all your modifications will be reset by CCE if the listener is deleted.	You are not advised to modify the permissions. Restore the permissions if they are modified.

Operation	Impact	Solution
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"> ● Service/Ingress access fails. ● After the 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"> - The deleted backend server will be restored. - The added backend server will be removed. 	Re-create or update the Service or ingress.
Replacing the backend server group of a listener created by CCE on the ELB console	<ul style="list-style-type: none"> ● Service/Ingress access fails. ● After the master nodes are restarted, for example, due to a cluster upgrade, all servers in the backend server group will be reset by CCE. 	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 a forwarding rule	<ul style="list-style-type: none"> ● Service/Ingress access fails. ● After the master nodes are restarted, for example, due to a cluster upgrade, all your modifications will be reset by CCE if the forwarding rule is added by the ingress. 	You are not advised to modify the permissions. Restore the permissions if they are modified.
Changing the ELB certificate on the ELB console for the load balancer managed by CCE	After the 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 5-4 High-risk operations and solutions

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

EVS Disks

Table 5-5 High-risk operations and solutions

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

6 clusters

6.1 Cluster Overview

6.1.1 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 Models](#).

Master Nodes and Cluster Scale

When you create a cluster on CCE, you can have one or three master nodes. Three master nodes can create a cluster in HA mode.

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 6-1 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.
Unavailable	A cluster is unavailable.
Deleting	A cluster is being deleted.

6.1.2 Kubernetes Release Notes

6.1.2.1 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 document 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 of version 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 General Availability (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. The 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 the **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](#)

6.1.2.2 Kubernetes 1.25 Release Notes

CCE has passed the Certified Kubernetes Conformance Program and is a certified Kubernetes offering. This document 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 [Common Expression Language \(CEL\)](#). 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.

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.
The LegacyServiceAccountTokenNoAutoGeneration feature is in beta state. By default, this feature is enabled and no more secret token will be automatically generated for the service account. To use a token that never expires, create a secret and mount it. 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](#).
- The cloud service vendors' in-tree storage drivers 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 1.25 Release Notes](#)
- [Kubernetes 1.24 Release Notes](#)

6.1.2.3 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](#)

6.1.2.4 Kubernetes 1.21 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 will be 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. Update the `timeoutSeconds` field for the applications that use this feature during the upgrade. 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](#)

6.1.2.5 Kubernetes 1.19 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 **kubectrl 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 **kubectrl 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 counters** 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 AlwaysCheckAllPredictes 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](#)

6.1.2.6 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](#)

6.1.3 Release Notes for CCE Cluster Versions

Version 1.27

NOTICE

In CCE v1.27 and later versions, all nodes support only the containerd container engine.

Table 6-2 Release notes for the v1.27 patch

CCE Cluster Patch Version	Kubernetes Version	Feature Updates	Optimization	Vulnerability Fixing
v1.27.1-r0	v1.27.2	<p>CCE clusters of v1.27 are released for the first time. For more information, see Kubernetes 1.27 Release Notes.</p> <ul style="list-style-type: none"> Both soft eviction and hard eviction are supported in node pool configuration. 	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 6-3 Release notes for the v1.25 patch

CCE Cluster Patch Version	Kubernetes Version	Feature Updates	Optimization	Vulnerability Fixing
v1.25.1-r0	v1.25.5	<p>CCE clusters of v1.25 are released for the first time. For more information, see Kubernetes 1.25 Release Notes.</p>	None	None

Version 1.23

Table 6-4 Release notes for the v1.23 patch

CCE Cluster Patch Version	Kubernetes Version	Feature Updates	Optimization	Vulnerability Fixing
v1.23.8-r0	v1.23.11	None	<ul style="list-style-type: none"> Enhanced Docker reliability during upgrades. Optimized node time synchronization. 	Fixed some security issues.
v1.23.5-r0	v1.23.11	<ul style="list-style-type: none"> Fault detection and isolation are supported on GPU nodes. Security groups can be customized by cluster. containerd is supported. 	<ul style="list-style-type: none"> The ETCD version of the master node has been upgraded to the Kubernetes version 3.5.6. Scheduling is optimized so that pods are evenly distributed across AZs after pods are scaled in. Optimized the memory usage of kube-apiserver when CRDs are frequently updated. 	Fixed some security issues and the following CVE vulnerabilities: <ul style="list-style-type: none"> CVE-2022-3294 CVE-2022-3162 CVE-2022-3172 CVE-2021-25749
v1.23.1-r0	v1.23.4	CCE clusters of v1.23 are released for the first time. For more information, see Kubernetes 1.23 Release Notes .	None	None

Version 1.21

Table 6-5 Release notes for the v1.21 patch

CCE Cluster Patch Version	Kubernetes Version	Feature Updates	Optimization	Vulnerability Fixing
v1.21.10-r0	v1.21.14	None	<ul style="list-style-type: none"> Enhanced Docker reliability during upgrades. Optimized node time synchronization. Enhanced the stability of the Docker runtime for pulling images after nodes are restarted. 	Fixed some security issues.
v1.21.7-r0	v1.21.14	<ul style="list-style-type: none"> Fault detection and isolation are supported on GPU nodes. Security groups can be customized by cluster. 	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"> CVE-2022-3294 CVE-2022-3162 CVE-2022-3172
v1.21.1-r0	v1.21.7	CCE clusters of v1.21 are released for the first time. For more information, see Kubernetes 1.21 Release Notes .	None	None

Version 1.19

Table 6-6 Release notes of the v1.19 patch

CCE Cluster Patch Version	Kubernetes Version	Feature Updates	Optimization	Vulnerability Fixing
v1.19.16-r20	v1.19.16	None	<ul style="list-style-type: none"> Enhanced the stability of the Docker runtime for pulling images after nodes are restarted. 	Fixed some security issues.
v1.19.16-r4	v1.19.16	<ul style="list-style-type: none"> Fault detection and isolation are supported on GPU nodes. Security groups can be customized by cluster. 	<ul style="list-style-type: none"> Scheduling is optimized on taint nodes. Enhanced the long-term running stability of containerd when cores are bound. Improved the stability of LoadBalancer Services/ingresses with a large number of connections. Optimized the memory usage of kube-apiserver when CRDs are frequently updated. 	Fixed some security issues and the following CVE vulnerabilities: <ul style="list-style-type: none"> CVE-2022-3294 CVE-2022-3162 CVE-2022-3172
v1.19.16-r0	v1.19.16	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"> CVE-2021-25741 CVE-2021-25737

CCE Cluster Patch Version	Kubernetes Version	Feature Updates	Optimization	Vulnerability Fixing
v1.19.10-r0	v1.19.10	CCE clusters of v1.19 are released for the first time. For more information, see Kubernetes 1.19 Release Notes .	None	None

6.2 Creating a Cluster

6.2.1 CCE Turbo Clusters and CCE Clusters

Comparison Between CCE Turbo Clusters and CCE Clusters

The following table lists the differences between CCE Turbo clusters and CCE clusters.

Table 6-7 Cluster types

Category	Subcategory	CCE Turbo Cluster	CCE Cluster
Cluster	Positioning	Next-gen container cluster designed for Cloud Native 2.0, with accelerated computing, networking, and scheduling	Standard cluster for common commercial use
	Node type	Deployment of VMs	Deployment of VMs
Networking	Model	Cloud Native Network 2.0: applies to large-scale and high-performance scenarios. Max networking scale: 2,000 nodes	Cloud Native Network 1.0: applies to common, smaller-scale scenarios. <ul style="list-style-type: none"> • Tunnel network model • VPC network model
	Performance	Flattens the VPC network and container network into one, achieving zero performance loss.	Overlays the VPC network with the container network, causing certain performance loss.

Category	Subcategory	CCE Turbo Cluster	CCE Cluster
	Container network isolation	Associates pods with security groups. Unifies security isolation in and out the cluster via security groups' network policies.	<ul style="list-style-type: none"> • Tunnel network model: supports network policies for intra-cluster communications. • VPC network model: supports no isolation.
Security	Isolation	<ul style="list-style-type: none"> • Physical machine: runs Kata containers, allowing VM-level isolation. • VM: runs common containers, isolated by cgroups. 	Runs common containers, isolated by cgroups.

6.2.2 Creating a 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.

Constraints

- During the node creation, software packages are downloaded from OBS using the domain name. Use a private DNS server to resolve the OBS domain name, and configure the DNS server address of 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:
 - Cluster type
 - 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 ([request forwarding](#)) settings.
 - Network model. For example, change **Tunnel network** to **VPC network**.

Procedure

Step 1 Log in to the CCE console.

Step 2 Choose **Clusters**. On the displayed page, select the type of the cluster to be created and click **Create**.

Step 3 Specify cluster parameters.

Basic Settings

- **Cluster Name:** indicates the name of the cluster to be created. The cluster name must be unique under the same account.
- **Enterprise Project:**
This parameter is displayed only for enterprise users who have enabled the enterprise project function.
After an enterprise project (for example, **default**) 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.
An enterprise project facilitates project-level management and grouping of cloud resources and users.
- **Cluster Version:** Select the Kubernetes version used by the cluster.
- **Cluster Scale:** maximum number of nodes that can be managed by the cluster.
- **HA:** distribution mode of master nodes. By default, master nodes are randomly distributed in different AZs to improve DR capabilities.
You can also expand advanced settings and customize the master node distribution mode. The following two modes are supported:
 - **Random:** Master nodes are created in different AZs for DR.
 - **Custom:** You can determine the location of each master node.
 - **Host:** Master nodes are created on different hosts in the same AZ.
 - **Custom:** You can determine the location of each master node.

Network Settings

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

- **Network Model:** CCE clusters support **VPC network** and **Tunnel network**. CCE Turbo clusters support **Cloud Native Network 2.0**. For details, see [Overview](#).
- **VPC:** Select the VPC to which the cluster belongs. If no VPC is available, click **Create VPC** to create one. The value cannot be changed after creation.
- **Master Node Subnet:** Select the subnet where the master node is deployed. If no subnet is available, click **Create Subnet** to create one. The subnet cannot be changed after creation.
- **Container CIDR Block (CCE Cluster):** Specify the CIDR block used by containers, which determines the maximum number of containers in the cluster.
- **Default Pod Subnet (CCE Turbo Cluster):** Select the subnet where the container is located. If no subnet is available, click **Create Subnet**. The pod subnet determines the maximum number of containers in the cluster. You can add pod subnets after creating the cluster.

- **IPv4 Service CIDR Block:** CIDR block for Services used by 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 creation.

Advanced Settings

- **Request Forwarding:** The IPVS and iptables modes are supported. For details, see [Comparing iptables and IPVS](#).
- **CPU Manager:** When enabled, CPU cores will be exclusively allocated to workload pods. For details, see [CPU Policy](#).
- **Certificate Authentication:**
 - **Default:** The X509-based authentication mode is enabled by default. X509 is a commonly used certificate format.
 - **Custom:** 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** of the client certificate.

CAUTION

- Upload a file **smaller than 1 MiB**. 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.**
- Starting from v1.25, Kubernetes no longer supports certificate authentication generated using the SHA1WithRSA or ECDSAWithSHA1 algorithm. You are advised to use the SHA256 algorithm.

-
- **Description:** The description cannot exceed 200 characters.

Step 4 Click **Next: Add-on Configuration**.

Domain Name Resolution:

- **Domain Name Resolution:** The [coredns](#) add-on is installed by default to resolve domain names and connect to the cloud DNS server.

Container Storage: The [everest](#) add-on is installed by default to provide container storage based on CSI and connect to cloud storage services such as EVS.

Fault Detection: The [npd](#) add-on is installed by default to provide node fault detection and isolation for the cluster, helping you identify node problems in a timely manner.

Data Plane Logs

- Using ICAgent:
A log collector provided by Application Operations Management (AOM), reporting logs to AOM and Log Tank Service (LTS) according to the log collection rules you configured.
You can collect stdout logs as required.

Overload Control: If enabled, concurrent requests are dynamically controlled based on the resource pressure of master nodes to keep them and the cluster available. For details, see [Cluster Overload Control](#).

Step 5 After the parameters are specified, click **Next: Confirm**. The cluster resource list is displayed. Confirm the information and click **Submit**.

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.

----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 nodes to the cluster. For details, see [Creating a Node](#).

6.2.3 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 IPVS forwarding modes.

- 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 there are a large number of short concurrent connections on the client. When there are more than 1,000 Services in the cluster, network delay may occur.

Constraints

- 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. You are advised to use different ELB load balancers for the ingress and Service.
- In iptables mode, the ClusterIP cannot be pinged. In IPVS mode, the ClusterIP can be pinged.

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.

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 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.

The complexity of the connection process of IPVS-based kube-proxy is $O(1)$. 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.

6.3 Connecting to a Cluster

6.3.1 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.

Permissions

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.

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, click the name of the

cluster to be connected, and view the access address and kubectl connection procedure on the cluster details page.

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 a public IP (EIP) to the cluster, go to the cluster details page and click **Bind** next to **EIP** in the **Connection Information** pane. In a cluster with an EIP bound, kube-apiserver will be exposed to public networks and may be attacked. You are advised to configure Advanced Anti-DDoS (AAD) for the EIP of the node where kube-apiserver resides.

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. Procedure:

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 number. Replace it as required.

2. Install kubectl.

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

Step 2 Obtain the kubectl configuration file (kubeconfig).

In the **Connection Information** pane on the cluster details page, click **Configure** next to **kubectl**. On the window displayed, download the configuration file.

 NOTE

- The kubectl configuration file **kubeconfig.json** is used for cluster authentication. If the file is leaked, your clusters may be attacked.
- By default, two-way authentication is disabled for domain names in the current cluster. You can run the **kubectl config use-context externalTLSVerify** command to enable two-way authentication. For details, see [Two-Way Authentication for Domain Names](#). 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.json** 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**.

Step 3 Configure kubectl.

Configure kubectl (A Linux OS is used).

1. Log in to your client and copy the kubeconfig.json configuration file 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.json $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.

- Two-way authentication is disabled for domain names by default. You can run the **kubectl config use-context externalTLSVerify** command to switch to the externalTLSVerify context to enable it.
- 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.json** again.
- If the domain name two-way authentication is not supported, **kubeconfig.json** contains the **"insecure-skip-tls-verify": true** field, as shown in [Figure 6-1](#). To use two-way authentication, you can download the **kubeconfig.json** file again and enable two-way authentication for the domain names.

Figure 6-1 Two-way authentication disabled for domain names

```

"clusters": [{
  "name": "mycluster",
  "cluster": {
    "server": "https://192.168.1.1:5443",
    "insecure-skip-tls-verify": true
  }
}]

```

Common Issues

- **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](#).

6.3.2 Connecting to a Cluster Using an X.509 Certificate

Scenario

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

Procedure

- Step 1** Log in to the CCE console and click the cluster name to access the cluster console.
- Step 2** Choose **Cluster Information** from the navigation pane and click **Download** next to **Certificate Authentication** in the **Connection Information** area.
- 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 view the pod information. In the following information, *192.168.***.***:5443* indicates the IP address of the API server in the cluster.

```
curl --cacert ./ca.crt --cert ./client.crt --key ./client.key https://192.168.***.***:5443/api/v1/namespaces/default/pods/
```

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

----End

6.3.3 Accessing a Cluster Using a Custom Domain Name

Scenario

A **Subject Alternative Name (SAN)** can be signed in to a cluster server certificate. 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 to enable two-way authentication on the client, which improves security. Typical use cases include DNAT access and domain name access.

Typical domain name access scenarios:

- Add the response domain name mapping when specifying 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.


Constraints

This feature is available only to clusters of v1.19 and later.

Customizing a SAN

Step 1 Log in to the CCE console.

Step 2 Click the target cluster in the cluster list to go to the cluster details page.

Step 3 In the **Connection Information** area, click  next to **Custom SAN**. In the dialog box displayed, add the IP address or domain name and click **Save**.

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

6.4 Upgrading a Cluster

6.4.1 Upgrade 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 tag will be displayed on the cluster card view if there is a new version for the cluster to upgrade.

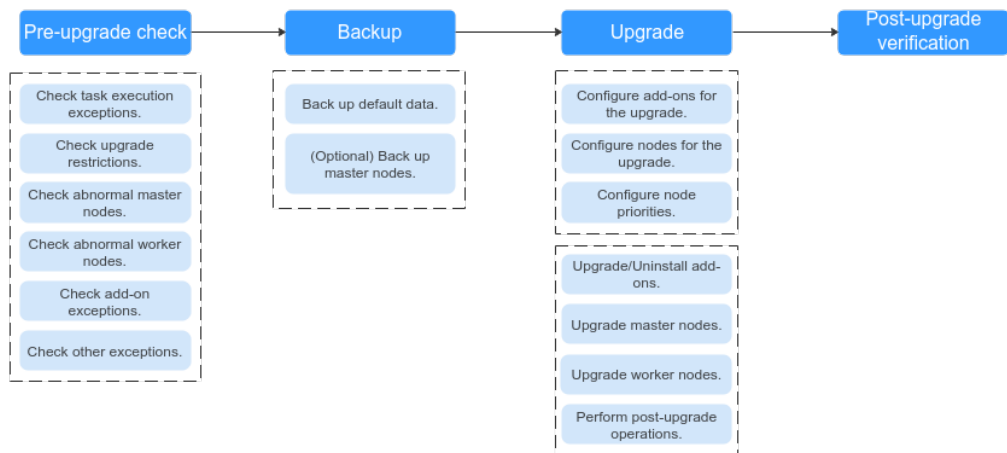
How to check:

Log in to the CCE console and check whether the message "New version available" is displayed in the lower left corner of the cluster. If yes, the cluster can be upgraded. View the release notes for the latest version. For details, see [Release Notes for CCE Cluster Versions](#). If no such a message is displayed, the cluster is of the latest version.

Cluster Upgrade Process

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

Figure 6-2 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 the compatibility of nodes, add-ons, and workloads in the cluster to reduce the probability of upgrade failures to the best extent. If any exception is detected, rectify the fault as prompted on the console.

2. Backup

During the upgrade, cluster data is backed up by default. You can also back up the entire master nodes as needed. Cloud Backup and Recovery (CBR) will be used for full-node backup. It takes about 20 minutes to back up one node.

3. Upgrade

During the upgrade, configure upgrade parameters, such as the step for add-on upgrade or node rolling upgrade. After the upgrade parameters are configured, the add-ons and nodes will be upgraded one by one.

4. Post-upgrade verification

After the upgrade, manually check services and ensure that services are not interrupted by the upgrade.

Cluster Upgrade

The following table describes the target version to which each cluster version can be upgraded and the supported upgrade modes.

Table 6-8 Cluster upgrade

Source Version	Target Version	Upgrade Mode
v1.25	v1.27	In-place upgrade
v1.23	v1.25	In-place upgrade

Source Version	Target Version	Upgrade Mode
v1.21	v1.25 v1.23	In-place upgrade
v1.19	v1.23 v1.21	In-place upgrade
v1.17	v1.19	In-place upgrade
v1.15	v1.19	In-place upgrade
v1.13	v1.15	Rolling upgrade

Upgrade Modes

Different upgrade modes have different advantages and disadvantages.

Table 6-9 Differences between upgrade modes and their advantages and disadvantages

Upgrade Mode	Method	Advantage	Disadvantage
In-place upgrade	Kubernetes components, network components, and CCE management components are upgraded on the node. During the upgrade, service pods and networks are not affected. The SchedulingDisabled 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 data after the node upgrade is complete and reset the node to upgrade the OS to a new version.

Upgrade Mode	Method	Advantage	Disadvantage
Rolling upgrade	<p>Only the Kubernetes components and certain network components are upgraded on the node. The SchedulingDisabled label will be added to all existing nodes to ensure that the running applications are not affected.</p> <p>NOTICE</p> <ul style="list-style-type: none"> • After the upgrade is complete, manually create nodes and gradually release the old nodes, thereby migrating your applications to the new nodes. In this mode, you can control the upgrade process. 	Services are not interrupted.	<ul style="list-style-type: none"> • After the upgrade is complete, manually create nodes and gradually release the old nodes. The new nodes are billed additionally. After services are migrated to the new nodes, the old nodes can be deleted. • After the rolling upgrade is complete, if you want to continue the upgrade to a later version, reset the old nodes first. Otherwise, the pre-upgrade check cannot be passed. Services may be interrupted during the upgrade.

6.4.2 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 [Upgrade Overview](#).

Precautions

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

- **Upgrading a cluster cannot be rolled back. Perform an upgrade at a proper time to minimize the impact on your services.** To ensure data security, you back up your data before an upgrade.
- Before upgrading a cluster, **ensure that 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.

- Before upgrading a cluster, ensure the cluster is working properly.
- 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. Do not **stop, restart, or delete nodes** during cluster upgrade. Otherwise, the upgrade will fail.
- During a cluster upgrade, the running workloads will not be interrupted, but access to the API server will be temporarily interrupted.
- During a cluster upgrade, the **node.kubernetes.io/upgrade** taint (equivalent to **NoSchedule**) will be added to the nodes in the cluster. The taint will be removed after the cluster is upgraded. Do not add taints with the same key name on a node. Even if the taints have different effects, they may be deleted by the system by mistake after the upgrade.

Constraints

- CCE clusters with VM nodes can be upgraded.
- If there are any nodes created using a private image, the cluster cannot be upgraded.
- After the cluster is upgraded, if the containerd vulnerability of the container engine is fixed in [Kubernetes Release Notes](#), manually restart containerd for the upgrade to take effect. The same applies to the existing pods.
- If you mount the **docker.sock** file on a node to a pod using the hostPath mode, that is, the Docker in Docker scenario, Docker will restart during the upgrade, but the **docker.sock** file does not change. As a result, your services may malfunction. You are advised to mount the **docker.sock** file by mounting the directory.
- 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.

Deprecated APIs

With the evolution of Kubernetes APIs, APIs are periodically reorganized or upgraded, and old 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 v1.25](#)
- [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 6-10 Deprecated APIs in Kubernetes v1.25

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

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

Resource Name	Deprecated API Version	Substitute API Version	Change Description
			<p>events.k8s.io/v1 event object).</p> <ul style="list-style-type: none"> Use reportingInstance instead of the deprecated source.host field (this field has been renamed deprecatedSource.host and is not allowed to appear in the new events.k8s.io/v1 event object).
HorizontalPod Autoscaler	autoscaling/v2beta1	autoscaling/v2 (This API is available since v1.23.)	None
PodDisruption Budget	policy/v1beta1	policy/v1 (This API is available since v1.21.)	If spec.selector is set to null ({}) in PodDisruptionBudget of policy/v1 , all pods in the namespace are selected. (In policy/v1beta1 , an empty spec.selector means that no pod will be selected.) If spec.selector 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 policy/v1beta1 version, and the PodSecurityPolicy access controller is deleted. Replace it with Configuring Pod Security Admission .
RuntimeClass	node.k8s.io/v1beta1	node.k8s.io/v1 (This API is available since v1.20.)	None

Table 6-11 Deprecated APIs in Kubernetes v1.22

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

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

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

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

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

Resource Name	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"> CSIDriver is available in storage.k8s.io/v1 since v1.19. CSINode is available in storage.k8s.io/v1 since v1.17. StorageClass is available in storage.k8s.io/v1 since v1.6. VolumeAttachment is available in storage.k8s.io/v1 since v1.13.

Table 6-12 Deprecated APIs in Kubernetes v1.16

Resource Name	Deprecated API Version	Substitute API Version	Change Description
NetworkPolicy	extensions/v1beta1	networking.k8s.io/v1 (This API is available since v1.8.)	None
DaemonSet	extensions/v1beta1 apps/v1beta2	apps/v1 (This API is available since v1.9.)	<ul style="list-style-type: none"> The spec.templateGeneration field is deleted. spec.selector 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. The default value of spec.updateStrategy.type is changed to RollingUpdate (the default value in the extensions/v1beta1 API version is OnDelete).

Resource Name	Deprecated API Version	Substitute API Version	Change Description
Deployment	extensions/v1beta1 apps/v1beta1 apps/v1beta2	apps/v1 (This API is available since v1.9.)	<ul style="list-style-type: none"> • The spec.rollbackTo field is deleted. • spec.selector 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. • The default value of spec.progressDeadlineSeconds is changed to 600 seconds (the default value in extensions/v1beta1 is unlimited). • The default value of spec.revisionHistoryLimit is changed to 10. (In the apps/v1beta1 API version, the default value of this field is 2. In the extensions/v1beta1 API version, all historical records are retained by default.) • The default values of maxSurge and maxUnavailable are changed to 25%. (In the extensions/v1beta1 API version, these fields default to 1.)
StatefulSet	apps/v1beta1 apps/v1beta2	apps/v1 (This API is available since v1.9.)	<ul style="list-style-type: none"> • spec.selector 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. • The default value of spec.updateStrategy.type is changed to RollingUpdate (the default value in the apps/v1beta1 API version is OnDelete).

Resource Name	Deprecated API Version	Substitute API Version	Change Description
ReplicaSet	extensions/v1beta1 apps/v1beta1 apps/v1beta2	apps/v1 (This API is available since v1.9.)	spec.selector 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 is available since v1.10.)	PodSecurityPolicy for the policy/v1beta1 API version will be removed in v1.25.

Version Differences

Upgrade Path	Version Difference	Self-Check
v1.23 to v1.25	Since Kubernetes v1.25, PodSecurityPolicy has been replaced by pod Security Admission (Configuring Pod Security Admission).	<ul style="list-style-type: none"> To migrate PodSecurityPolicy capabilities to pod Security Admission, perform the following steps: <ol style="list-style-type: none"> Ensure that the cluster is of the latest CCE v1.23 version. Migrate PodSecurityPolicy capabilities to pod Security Admission. For details, see Migrating from Pod Security Policy to Pod Security Admission. After confirming that the functions are running properly after the migration, upgrade the CCE cluster to v1.25. If you no longer need PodSecurityPolicy, delete PodSecurityPolicy from the cluster and upgrade the cluster to v1.25.

Upgrade Path	Version Difference	Self-Check
<p>v1.21 to v1.23</p> <p>v1.19 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 kubernetes.io/ingress.class: nginx is not set in the ingress annotations. 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>See nginx-ingress Upgrade Check.</p>
<p>v1.19 to v1.21</p>	<p>The bug of exec probe timeouts is fixed in Kubernetes 1.21. 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. If this field is not specified, the default value 1 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 1.19 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 CommonName is discarded in Go 1.15. kube-apiserver of CCE 1.19 is compiled using Go 1.15. If your webhook certificate does not have SANs, kube-apiserver does not process the CommonName 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"> • If you do not have your own webhook server, you can skip this check. • If the field is not set, you are advised to use the SAN field to specify the IP address and domain name supported by the certificate.
<p>v1.15 to v1.19</p>	<p>The control plane of CCE clusters of v1.19 is incompatible with kubelet v1.15. If a node fails to be upgraded or the node to be upgraded restarts after the master node is successfully upgraded, there is a high probability that the node is in the NotReady status.</p> <p>This is because the node failed to be upgraded restarts the kubelet and trigger the node registration. In clusters of v1.15, the default registration tags (failure-domain.beta.kubernetes.io/is-baremetal and kubernetes.io/availablezone) are regarded as invalid tags by the clusters of v1.19.</p> <p>The valid tags in the clusters of v1.19 are node.kubernetes.io/baremetal and failure-domain.beta.kubernetes.io/zone.</p>	<ol style="list-style-type: none"> 1. In normal cases, this scenario is not triggered. 2. After the master node is upgraded, do not suspend the upgrade so the node can be quickly upgraded. 3. If a node fails to be upgraded and cannot be restored, evict applications on the node as soon as possible. Contact technical support and skip the node upgrade. After the upgrade is complete, reset the node.

Upgrade Path	Version Difference	Self-Check
	<p>In CCE 1.15 and 1.19 clusters, the Docker storage driver file system is switched from XFS to Ext4. As a result, the import package sequence in the pods of the upgraded Java application may be abnormal, causing pod exceptions.</p>	<p>Before the upgrade, check the Docker configuration file /etc/docker/daemon.json on the node. Check whether the value of dm.fs is xfs.</p> <ul style="list-style-type: none"> • If the value is ext4 or the storage driver is Overlay, you can skip the next steps. • If the value is xfs, you are advised to deploy applications in the cluster of the new version in advance to test whether the applications are compatible with the new cluster version. <pre data-bbox="975 864 1430 1122"> { "storage-driver": "devicemapper", "storage-opts": ["dm.thinpooldev=/dev/mapper/vgpaas-thinpool", "dm.use_deferred_removal=true", "dm.fs=xfs", "dm.use_deferred_deletion=true"] } </pre>
	<p>kube-apiserver of CCE 1.19 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 CommonName is discarded in Go 1.15. kube-apiserver of CCE 1.19 is compiled using Go 1.15. The CommonName field is processed as the host name. 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"> • If you do not have your own webhook server, you can skip this check. • If the field is not set, you are advised to use the SAN field to specify the IP address and domain name supported by the certificate. <p>NOTICE To mitigate the impact of version differences on cluster upgrade, CCE performs special processing during the upgrade from 1.15 to 1.19 and still supports certificates without SANs. However, no special processing is required for subsequent upgrades. You are advised to rectify your certificate as soon as possible.</p>

Upgrade Path	Version Difference	Self-Check
	<p>In clusters of v1.17.17 and later, CCE automatically creates pod security policies (PSPs) for you, which restrict the creation of pods with unsafe configurations, for example, pods for which net.core.somaxconn under a sysctl is configured in the security context.</p> <p>If initContainer or Istio is used in the in-place upgrade of a cluster of v1.15, pay attention to the following restrictions:</p> <p>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.</p>	<p>After an upgrade, you can allow insecure system configurations as required. For details, see Configuring a Pod Security Policy.</p> <p>You are advised to modify the QoS class of the service container before the upgrade to avoid this problem. For details, see Table 6-13.</p>
v1.13 to v1.15	<p>After a VPC network cluster is upgraded, the master node occupies an extra CIDR block due to the upgrade of network components. If no container CIDR block is available for the new node, the pod scheduled to the node cannot run.</p>	<p>Generally, this problem occurs when the nodes in the cluster are about to fully occupy the container CIDR block. For example, the container CIDR block is 10.0.0.0/16, the number of available IP addresses is 65,536, and 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). If the upper limit is 128, the cluster supports a maximum of 512 (65536/128) nodes, including the three master nodes. After the cluster is upgraded, each of the three master nodes occupies one CIDR block. As a result, 506 nodes are supported.</p>

Table 6-13 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

Upgrade Backup

How to back up a node:

- 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.

6.4.3 Performing In-place Upgrade

You can upgrade your clusters to a newer 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 [Upgrade 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

The cluster upgrade goes through check, backup, configuration and upgrade, and verification.

Step 1 Log in to the CCE console and click the cluster name to access the cluster console.

Step 2 In the navigation pane, choose **Cluster Upgrade**.

Step 3 Perform the pre-upgrade check. Click **Start Check** and confirm the check. If there are abnormal or risky items in the cluster, handle the exceptions based on the check results displayed on the page and check again.

- **Exceptions:** View the solution displayed on the page, handle the exceptions and check again.
- **Risk Items:** may affect the cluster upgrade. Check the risk description and see whether you may be impacted. If no risk exists, click **OK** next to the risk item to manually skip this risk item and check again.

After the check is passed, click **Next: Back Up**.


Step 4 (Optional) Manually back up the cluster data. Data is backed up during the upgrade following a default policy. You can click **Back Up** to manually back up data. If you do not need to manually back up data, click **Next: Configure & Upgrade**.

Manual backup will back 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 may take longer. The cluster cannot be upgraded during the backup.

Step 5 Configure the upgrade parameters.

- **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:** You can set the maximum number of nodes to be upgraded in a batch.
- **Node Priority:** You can 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. Before setting the node upgrade priority, select a node pool. Nodes and node pools will be upgraded according to the priorities you specify.
 - **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.

Step 6 After the configuration is complete, click **Upgrade** and confirm the upgrade. The cluster starts to be upgraded. You can view the 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**. When the progress bar reaches 100%, the cluster upgrade is complete.

 **NOTE**

If an upgrade failure message is displayed during the cluster upgrade, rectify the fault as prompted and try again.

Step 7 After the upgrade is complete, click **Next: Verify**. Verify the upgrade based on the displayed check items. After confirming that all check items are normal, click **Complete** and confirm that the post-upgrade check is complete.

You can verify the cluster Kubernetes version on the **Clusters** page.

----End

6.4.4 Performing Post-Upgrade Verification

6.4.4.1 Service Verification

Check Item

After the cluster is upgraded, check whether the services are running normal.

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 are abnormal after the cluster upgrade, contact technical support.

6.4.4.2 Pod Check

Check Item

- Check whether there are unexpected pods in the cluster.
- Check whether there are any pods that ran properly originally in the cluster restart unexpectedly.

Procedure

Log in to the CCE console and access the cluster console. Choose **Workloads** in the navigation pane. On the displayed page, switch to the **Pods** tab page. Select all namespaces, click **Status**, and check whether there are any abnormal pods.

View the **Restarts** column to check whether there are pods that are restarted abnormally.

Solution

If there are abnormal pods in your cluster after the cluster upgrade, contact technical support.

6.4.4.3 Node and Container Network Check

Check Item

- Check whether the nodes are running properly.
- Check whether the node network is normal.
- Check whether the container network is normal.

Procedure

The node status reflects whether the node component or network is normal.

Go to the CCE console and access the cluster console. Choose **Nodes** in the navigation pane. You can filter node status by status to check whether there are abnormal nodes.

The container network affects services. Check whether your services are available.

Solution

If the node status is abnormal, contact technical support.

If the container network is abnormal and your services are affected, contact technical support and confirm the abnormal network access path.

Source	Destination	Destination Type	Possible Fault
<ul style="list-style-type: none"> • Pods (inside a cluster) • Nodes (inside a cluster) • Cloud servers outside the cluster but in the same VPC as the cluster • Outside the VPC to which the cluster belongs 	Public IP address of Service ELB	Cluster traffic load balancing entry	No record.
	Private IP address of Service ELB	Cluster traffic load balancing entry	No record.
	Public IP address of ingress ELB	Cluster traffic load balancing entry	No record.
	Private IP address of ingress ELB	Cluster traffic load balancing entry	No record.
	Public IP address of NodePort Service	Cluster traffic entry	The kube proxy configuration is overwritten. This fault has been rectified in the upgrade process.
	Private IP address of NodePort Service	Cluster traffic entry	No record.
	ClusterIP Service	Service network plane	No record.
	Non NodePort Service port	Container network	No record.
	Cross-node pods	Container network plane	No record.
	Pods on the same node	Container network plane	No record.
	Service and pod domain names are resolved by CoreDNS.	Domain name resolution	No record.

Source	Destination	Destination Type	Possible Fault
	External domain names are resolved based on the CoreDNS hosts configuration.	Domain name resolution	After CoreDNS is upgraded, the configuration is overwritten. This fault has been rectified in the add-on upgrade process.
	External domain names are resolved based on the CoreDNS upstream server.	Domain name resolution	After CoreDNS is upgraded, the configuration is overwritten. This fault has been rectified in the add-on upgrade process.
	External domain names are not resolved by CoreDNS.	Domain name resolution	No record.

6.4.4.4 Node Label and Taint Check

Check Item

- Check whether custom node labels are lost.
- Check whether there are any unexpected taints newly added on the node, which will affect workload scheduling.

Procedure

Go to the CCE console, access the cluster console, and choose **Nodes** in the navigation pane. On the displayed page, click the **Nodes** tab, select all nodes, and click **Manage Labels and Taints** to view the labels and taints of the current node.

Solution

Custom labels will not be changed during a cluster upgrade. If you find that labels are lost or added unexpectedly, contact technical support.

If you find a new taint (**node.kubernetes.io/upgrade**) on a node, the node may be skipped during the upgrade. For details, see [Node Skipping Check](#).

If you find that other taints are added to the node, contact technical support.

6.4.4.5 New Node Check

Check Item

Check whether nodes can be created in the cluster.

Procedure

Log in to the CCE console and access the cluster console. Choose **Nodes** in the navigation pane, and click **Create Node**. For details about node configurations, see [Creating a Node](#).

Solution

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

6.4.4.6 New Pod Check

Check Item

- 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.

Go to the CCE console, access the cluster console, and choose **Workloads** in the navigation pane. On the displayed page, switch to the **DaemonSets** tab page and click **Create Workload** or **Create from YAML** in the upper right corner. For details, see [Creating a DaemonSet](#).

You are advised to use the image for routine tests as the base image. You can deploy a pod 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 10 MB CPU 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
```

After the workload is created, check whether the pod status of the workload is normal.

After the check is complete, go to the CCE console and access the cluster console. Choose **Workloads** in the navigation pane. On the displayed page, switch to the **DaemonSets** tab page, choose **More > Delete** in the **Operation** column of the **post-upgrade-check** workload to delete the test workload.

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.

6.4.4.7 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.

6.4.5 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 6-14 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"> 1. Check whether the storage resources in use are provisioned by the cloud or by your organization. 2. Change the automatically created storage to the existing storage in the new cluster.
Network	<ol style="list-style-type: none"> 1. Pay special attention to the ELB and ingress. 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.
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 [Creating a Cluster](#).

Step 2 Add a node.

Add nodes 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 an existing storage volume 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 OBS buckets and SFS Turbo file systems. If non-shared storage is used, suspend the workloads in the old cluster to switch the storage resources. As a result, services will be unavailable.

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](#).

Step 5 Mount the storage again.

Mount the existing storage in the workload again. 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

6.4.6 Troubleshooting for Pre-upgrade Check Exceptions

6.4.6.1 Pre-upgrade Check

The system performs a comprehensive pre-upgrade check before the cluster upgrade. If the cluster does not meet the pre-upgrade check conditions, the upgrade cannot continue. To prevent upgrade risks, you can perform pre-upgrade check according to the check items provided by this section.

Table 6-15 Check items

No.	Check Item	Description
1	Node Restrictions	<ul style="list-style-type: none"> • Check whether the node is available. • Check whether the node OS supports the upgrade. • Check whether there are unexpected node pool tags in the node. • Check whether the Kubernetes node name is consistent with the ECS name.
2	Upgrade Management	Check whether the current user is in the upgrade blocklist.
3	Add-ons	<ul style="list-style-type: none"> • Check whether the add-on status is normal. • Check whether the add-on support the target version.
4	Helm Charts	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	SSH Connectivity of Master Nodes	Check whether CCE can connect to your master nodes.
6	Node Pools	Check the node pool status.
7	Security Groups	Check whether the security group allows the master node to access nodes using ICMP.
8	Arm Node Restrictions	<ul style="list-style-type: none"> • Check whether the cluster contains Arm nodes.
9	To-Be-Migrated Nodes	Check whether the node needs to be migrated.
10	Discarded Kubernetes Resources	Check whether there are discarded resources in the clusters.
11	Compatibility Risks	Read the version compatibility differences and ensure that they are not affected. The patch upgrade does not involve version compatibility differences.
12	Node CCE Agent Versions	Check whether cce-agent on the current node is of the latest version.
13	Node CPU Usage	Check whether the CPU usage of the node exceeds 90%.

No.	Check Item	Description
14	CRDs	<ul style="list-style-type: none"> • 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.
15	Node Disks	<ul style="list-style-type: none"> • Check whether the key data disks on the node meet the upgrade requirements. • Check whether the /tmp directory has 500 MiB available space.
16	Node DNS	<ul style="list-style-type: none"> • 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.
17	Node Key Directory File Permissions	Check whether the key directory /var/paas on the nodes contain files with abnormal owners or owner groups.
18	Kubelet	Check whether the kubelet on the node is running properly.
19	Node Memory	Check whether the memory usage of the node exceeds 90%.
20	Node Clock Synchronization Server	Check whether the clock synchronization server ntpd or chronyd of the node is running properly.
21	Node OS	Check whether the OS kernel version of the node is supported by CCE.
22	Node CPUs	Check whether the number of CPUs on the master node is greater than 2.
23	Node Python Commands	Check whether the Python commands are available on a node.
24	ASM Version	<ul style="list-style-type: none"> • Check whether ASM is used by the cluster. • Check whether the current ASM version supports the target cluster version.
25	Node Readiness	Check whether the nodes in the cluster are ready.
26	Node journald	Check whether journald of a node is normal.
27	containerd.sock	Check whether the containerd.sock file exists on the node. This file affects the startup of container runtime in the Euler OS.

No.	Check Item	Description
28	Internal Errors	Before the upgrade, check whether an internal error occurs.
29	Node Mount Points	Check whether inaccessible mount points exist on the node.
30	Kubernetes Node Taints	Check whether the taint needed for cluster upgrade exists on the node.
31	everest Restrictions	Check whether there are any compatibility restrictions on the current everest add-on.
32	cce-hpa-controller Restrictions	Check whether the current cce-controller-hpa add-on has compatibility restrictions.
33	Enhanced CPU Policies	Check whether the current cluster version and the target version support enhanced CPU policy.
34	Health of Worker Node Components	Check whether the container runtime and network components on the worker nodes are healthy.
35	Health of Master Node Components	Check whether the Kubernetes, container runtime, and network components of the master nodes are healthy.
36	Memory Resource Limit of Kubernetes Components	Check whether the resources of Kubernetes components, such as etcd and kube-controller-manager, exceed the upper limit.
37	Discarded Kubernetes APIs	<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>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.</p>
38	IPv6 Capabilities of a CCE Turbo Cluster	If IPv6 is enabled for a CCE Turbo cluster, check whether the target cluster version supports IPv6.
39	Node NetworkManager	Check whether NetworkManager of a node is normal.
40	Node ID File	Check the ID file format.
41	Node Configuration Consistency	When you upgrade a CCE cluster to v1.19 or later, the system checks whether the following configuration files have been modified in the background.

No.	Check Item	Description
42	Node Configuration File	Check whether the configuration files of key components exist on the node.
43	CoreDNS Configuration Consistency	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.
44	sudo Commands of a Node	Check whether the sudo commands and sudo-related files of the node are working.
45	Key Commands of Nodes	Check whether some key commands that the node upgrade depends on are working.
46	Mounting of a Sock File on a Node	The docker/containerd.sock file on the node is mounted to the pod through a hostPath. During the upgrade, Docker/containerd restarts, but the sock file in the container does not change. As a result, an error may occur in your services.
47	HTTPS Load Balancer Certificate Consistency	Check whether the certificate used by an HTTPS load balancer has been modified on ELB.
48	Node Mounting	Check whether the default mount directory and soft link on the node have been manually mounted or modified.
49	Login Permissions of User paas on a Node	Check whether user paas is allowed to log in to a node.
50	Private IPv4 Addresses of Load Balancers	Check whether the load balancer associated with a Service is allocated with a private IPv4 address.
51	Historical Upgrade Records	Check whether the source version of the cluster is earlier than v1.11 and the target version is later than v1.23.
52	CIDR Block of the Cluster Management Plane	Check whether the CIDR block of the cluster management plane is the same as that configured on the backbone network.
53	GPU Add-on	The GPU add-on is involved in the upgrade, which may affect the GPU driver installation during the creation of a GPU node.
54	Nodes' System Parameter Settings	Check whether the default system parameter settings on your nodes are modified.

No.	Check Item	Description
55	Residual Package Versions	Check whether there are residual package versions in the current cluster.
56	Node Commands	Check whether the commands required for the upgrade are available on the node.
57	Node Swap	Check whether swap has been enabled on cluster nodes.
58	nginx-ingress Upgrade	Check whether there are compatibility issues that may occur during nginx-ingress upgrade.

6.4.6.2 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 and 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.

2. **The container engine of the node does not support the upgrade.**

This issue typically occurs when a cluster of an earlier version is upgraded to v1.27 or later. Clusters of v1.27 or later support only the containerd runtime. If your node runtime is not containerd, the upgrade cannot be performed. In this case, reset the node and change the node runtime to containerd.

3. **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 6-16 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 check result shows that the upgrade is not supported due to regional restrictions, contact technical support. NOTE If the target version is v1.27 or later, only Ubuntu 22.04 supports the upgrade.

4. **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 node pool to the default node pool, the node pool label **cce.cloud.com/cce-nodepool** is retained, affecting 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.

5. **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.

6. **The Kubernetes node corresponding to the affected node does not exist.**

It is possible that the node is being deleted. Check again later.

6.4.6.3 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.

6.4.6.4 Add-ons

Check Item

Check the following aspects:

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

Solution

- **Scenario 1: The add-on status is abnormal.**

Log in to the CCE console and go to the target cluster. Choose **O&M > Add-ons** to view and handle the abnormal add-on.

- **Scenario 2: The target version does not support the current add-on.**

The add-on cannot be automatically upgraded with the cluster. Log in to the CCE console and go to the target cluster. Choose **O&M > Add-ons** to manually upgrade the add-on.

- **Scenario 3: After the add-on is upgraded to the latest version, the target cluster version is still not supported.**

Log in to the CCE console and go to the target cluster. Choose **O&M > Add-ons** to manually uninstall the add-on. For details about the supported add-on versions and replacement solutions, see the [Help](#) document.

- **Scenario 4: 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
```

Log in to the CCE console and go to the target cluster. Choose **O&M > Add-ons** to manually upgrade the add-on.

6.4.6.5 Helm Charts

Check Item

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.

6.4.6.6 SSH Connectivity of Master Nodes

Check Item

Check whether CCE can connect to your master nodes.

Solution

Contact technical support.

6.4.6.7 Node Pools

Check Item

Check the node pool status.

Solution

Scenario: The node pool malfunctions.

Log in to the CCE console, go to the target cluster and choose **Nodes**. On the displayed page, click **Node Pools** tab and check the node pool status. If the node pool is being scaled, wait until the node pool scaling is complete.

6.4.6.8 Security Groups

Check Item

Check whether the security group allows the master node to access nodes using ICMP.

 **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 displayed:

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

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

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.

6.4.6.9 Arm Node Restrictions

Check Item

Check the following aspects:

- Check whether the cluster contains Arm nodes.

Solution

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

6.4.6.10 To-Be-Migrated Nodes

Check Item

Check whether the node needs to be migrated.

Solution

For the 1.15 cluster that is upgraded from 1.13 in rolling mode, migrate (reset or create and replace) all nodes before performing the upgrade again.

Solution 1

Go to the CCE console and access the cluster console. Choose **Nodes** in the navigation pane and click **More > Reset Node** in the **Operation** column of the corresponding node. For details, see [Resetting a Node](#). After the node is reset, retry the check task.

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.

6.4.6.11 Discarded Kubernetes Resources

Check Item

Check whether there are discarded resources in the clusters.

Solution

Scenario 1: The PodSecurityPolicy resource object has been discarded since clusters of 1.25.

Run the `kubectl get psp -A` command in the cluster to obtain the existing PSP object.

If these two objects are not used, skip the check. Otherwise, upgrade the corresponding functions to PodSecurity by referring to [Pod Security](#).

Scenario 2: The Service in the clusters of 1.25 or later has discarded annotation: tolerate-unready-endpoints.

Check whether the Service provided in the log information contains the annotation of **tolerate-unready-endpoints**. If yes, replace the annotation with the following fields:

```
publishNotReadyAddresses: true
```

6.4.6.12 Compatibility Risks

Check Item

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 to v1.25	Since Kubernetes v1.25, PodSecurityPolicy has been replaced by pod Security Admission (Configuring Pod Security Admission).	<ul style="list-style-type: none"> To migrate PodSecurityPolicy capabilities to pod Security Admission, perform the following steps: <ol style="list-style-type: none"> Ensure that the cluster is of the latest CCE v1.23 version. Migrate PodSecurityPolicy capabilities to pod Security Admission. For details, see Migrating from Pod Security Policy to Pod Security Admission. After confirming that the functions are running properly after the migration, upgrade the CCE cluster to v1.25. If you no longer need PodSecurityPolicy, delete PodSecurityPolicy from the cluster and upgrade the cluster to v1.25.

Upgrade Path	Version Difference	Self-Check
<p>v1.21 to v1.23</p> <p>v1.19 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 kubernetes.io/ingress.class: nginx is not set in the ingress annotations. 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>See nginx-ingress Upgrade Check.</p>
<p>v1.19 to v1.21</p>	<p>The bug of exec probe timeouts is fixed in Kubernetes 1.21. 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. If this field is not specified, the default value 1 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 1.19 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 CommonName is discarded in Go 1.15. kube-apiserver of CCE 1.19 is compiled using Go 1.15. If your webhook certificate does not have SANs, kube-apiserver does not process the CommonName 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"> • If you do not have your own webhook server, you can skip this check. • If the field is not set, you are advised to use the SAN field to specify the IP address and domain name supported by the certificate.
v1.15 to v1.19	<p>The control plane of CCE clusters of v1.19 is incompatible with kubelet v1.15. If a node fails to be upgraded or the node to be upgraded restarts after the master node is successfully upgraded, there is a high probability that the node is in the NotReady status.</p> <p>This is because the node failed to be upgraded restarts the kubelet and trigger the node registration. In clusters of v1.15, the default registration tags (failure-domain.beta.kubernetes.io/is-baremetal and kubernetes.io/availablezone) are regarded as invalid tags by the clusters of v1.19.</p> <p>The valid tags in the clusters of v1.19 are node.kubernetes.io/baremetal and failure-domain.beta.kubernetes.io/zone.</p>	<ol style="list-style-type: none"> 1. In normal cases, this scenario is not triggered. 2. After the master node is upgraded, do not suspend the upgrade so the node can be quickly upgraded. 3. If a node fails to be upgraded and cannot be restored, evict applications on the node as soon as possible. Contact technical support and skip the node upgrade. After the upgrade is complete, reset the node.

Upgrade Path	Version Difference	Self-Check
	<p>In CCE 1.15 and 1.19 clusters, the Docker storage driver file system is switched from XFS to Ext4. As a result, the import package sequence in the pods of the upgraded Java application may be abnormal, causing pod exceptions.</p>	<p>Before the upgrade, check the Docker configuration file /etc/docker/daemon.json on the node. Check whether the value of dm.fs is xfs.</p> <ul style="list-style-type: none"> • If the value is ext4 or the storage driver is Overlay, you can skip the next steps. • If the value is xfs, you are advised to deploy applications in the cluster of the new version in advance to test whether the applications are compatible with the new cluster version. <pre data-bbox="975 864 1430 1122"> { "storage-driver": "devicemapper", "storage-opts": ["dm.thinpooldev=/dev/mapper/vgpaas- thinpool", "dm.use_deferred_removal=true", "dm.fs=xfs", "dm.use_deferred_deletion=true"] } </pre>
	<p>kube-apiserver of CCE 1.19 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 CommonName is discarded in Go 1.15. kube-apiserver of CCE 1.19 is compiled using Go 1.15. The CommonName field is processed as the host name. 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"> • If you do not have your own webhook server, you can skip this check. • If the field is not set, you are advised to use the SAN field to specify the IP address and domain name supported by the certificate. <p>NOTICE To mitigate the impact of version differences on cluster upgrade, CCE performs special processing during the upgrade from 1.15 to 1.19 and still supports certificates without SANs. However, no special processing is required for subsequent upgrades. You are advised to rectify your certificate as soon as possible.</p>

Upgrade Path	Version Difference	Self-Check
	<p>In clusters of v1.17.17 and later, CCE automatically creates pod security policies (PSPs) for you, which restrict the creation of pods with unsafe configurations, for example, pods for which net.core.somaxconn under a <code>sysctl</code> is configured in the security context.</p> <p>If <code>initContainer</code> or <code>Istio</code> is used in the in-place upgrade of a cluster of v1.15, pay attention to the following restrictions:</p> <p>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.</p>	<p>After an upgrade, you can allow insecure system configurations as required. For details, see Configuring a Pod Security Policy.</p> <p>You are advised to modify the QoS class of the service container before the upgrade to avoid this problem. For details, see Table 6-13.</p>
v1.13 to v1.15	<p>After a VPC network cluster is upgraded, the master node occupies an extra CIDR block due to the upgrade of network components. If no container CIDR block is available for the new node, the pod scheduled to the node cannot run.</p>	<p>Generally, this problem occurs when the nodes in the cluster are about to fully occupy the container CIDR block. For example, the container CIDR block is 10.0.0.0/16, the number of available IP addresses is 65,536, and 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). If the upper limit is 128, the cluster supports a maximum of 512 (65536/128) nodes, including the three master nodes. After the cluster is upgraded, each of the three master nodes occupies one CIDR block. As a result, 506 nodes are supported.</p>

6.4.6.13 Node CCE Agent Versions

Check Item

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.

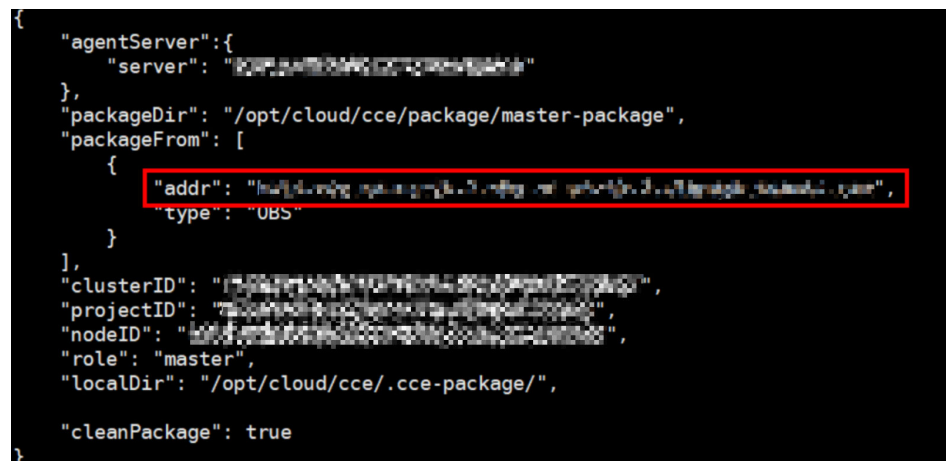
Solution

- a. Log in to a node where the check succeeded, obtain the path of the cce-agent configuration file, and obtain the OBS address.

```
cat `ps aux | grep cce-agent | grep -v grep | awk -F '|' '{print $2}'`
```

The OBS configuration address field in the configuration file is **packageFrom.addr**.

Figure 6-3 OBS address



```
{
  "agentServer": {
    "server": "https://obs.cn-north-1.amazonaws.com.cn"
  },
  "packageDir": "/opt/cloud/cce/package/master-package",
  "packageFrom": [
    {
      "addr": "https://obs.cn-north-1.amazonaws.com.cn",
      "type": "OBS"
    }
  ],
  "clusterID": "11111111111111111111111111111111",
  "projectID": "11111111111111111111111111111111",
  "nodeID": "11111111111111111111111111111111",
  "role": "master",
  "localDir": "/opt/cloud/cce/.cce-package/",
  "cleanPackage": true
}
```

- 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.

6.4.6.14 Node CPU Usage

Check Item

Check whether the CPU usage of the node exceeds 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.

6.4.6.15 CRDs

Check Item

Check the following aspects:

- 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.

6.4.6.16 Node Disks

Check Item

Check the following aspects:

- Check whether the key data disks on the node meet the upgrade requirements.
- Check whether the **/tmp** directory has 500 MiB available space.

Solution

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

- **Scenario 1: Check whether the disk meets the upgrade requirements.**

Run the following command to check the usage of each key disk. After ensuring that the available space meets the requirements and check again. If the space of the master node is insufficient, contact technical support.

- Disk partition of Docker: 2 GB for master nodes and 1 GB for worker nodes

```
df -h /var/lib/docker
```

- Disk partition of containerd: 2 GB for master nodes and 1 GB for worker nodes

```
df -h /var/lib/containerd
```

- Disk partition of kubelet: 2 GB for master nodes and 1 GB for worker nodes

```
df -h /mnt/paas/kubernetes/kubelet
```

- System disk: 10 GB for master nodes and 2 GB for worker nodes

```
df -h /
```

- **Scenario 2: The /tmp directory space 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
```

6.4.6.17 Node DNS

Check Item

Check the following aspects:

- 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.

6.4.6.18 Node Key Directory File Permissions

Check Item

Check whether the key directory `/var/paas` on the nodes contain files with abnormal owners or owner groups.

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 is stored in the `/var/paas` directory. If yes, do not use this directory, remove abnormal files from this directory, and check again. Otherwise, the upgrade is prohibited.

- **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.

6.4.6.19 Kubelet

Check Item

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

- **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.

6.4.6.20 Node Memory

Check Item

Check whether the memory usage of the node exceeds 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.

6.4.6.21 Node Clock Synchronization Server

Check Item

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 6-4 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 6-5 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.

6.4.6.22 Node OS

Check Item

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

Solution

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

Currently, this type of nodes should not be upgraded. You are advised to reset the node to the standard kernel version before the upgrade by following the instructions in [Resetting a Node](#).

6.4.6.23 Node CPUs

Check Item

Check whether the number of CPUs on the master node is greater than 2.

Solution

If the number of CPUs on the master node is 2, contact technical support to expand the number to 4 or more.

6.4.6.24 Node Python Commands

Check Item

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

Install Python before the upgrade.

6.4.6.25 ASM Version

Check Item

Check the following aspects:

- 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 6-17 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, v1.17, v1.19, or v1.21
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 v1.23, upgrade ASM to v1.13 first.

6.4.6.26 Node Readiness

Check Item

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 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.

6.4.6.27 Node journald

Check Item

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 6-6 Running status of journald

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

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

6.4.6.28 containerd.sock

Check Item

Check whether the containerd.sock file exists 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 **ls /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

6.4.6.29 Internal Errors

Check Item

Before the upgrade, check whether an internal error occurs.

Solution

If this check fails, contact technical support.

6.4.6.30 Node Mount Points

Check Item

Check whether inaccessible mount points exist on the node.

Solution

Scenario: There are inaccessible mount points on the node.

If network NFS (such as OBS, SFS, and SFS) is used by the node and the node is disconnected with the NFS server, the mount point would be inaccessible and all processes that access this mount point are suspended.

Step 1 Log in to the node.

Step 2 Run the following commands on the node in sequence:

```
- df -h
- for dir in `df -h | grep -v "Mounted on" | awk '{print \\$NF}';do cd $dir; done && echo "ok"
```

Step 3 If **ok** is returned, no problem occurs.

Otherwise, start another terminal and run the following command to check whether the previous command is in the D state:

```
- ps aux | grep "D "
```

Step 4 If a process is in the D state, the problem occurs. You can only reset the node to solve the problem. 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

6.4.6.31 Kubernetes Node Taints

Check Item

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

Table 6-18 Taint checklist

Taint Name	Impact
node.kubernetes.io/upgrade	NoSchedule

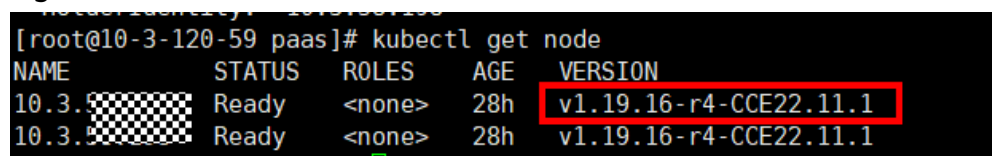
Solution

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

Step 1 For details about how to configure kubectl, see [Connecting to a Cluster Using kubectl](#).

Step 2 Check the kubelet version of the corresponding node. The following information is expected:

Figure 6-7 kubelet version



```
[root@10-3-120-59 paas]# kubectl get node
NAME          STATUS    ROLES    AGE    VERSION
10.3.5-120-59 Ready     <none>  28h   v1.19.16-r4-CCE22.11.1
10.3.5-120-59 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

6.4.6.32 everest Restrictions

Check Item

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

Table 6-19 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.

6.4.6.33 cce-hpa-controller Restrictions

Check Item

Check whether the current cce-controller-hpa add-on has compatibility restrictions.

Solution

The current cce-controller-hpa add-on has compatibility restrictions. An add-on that can provide metric APIs, for example, metric-server, must be installed in the cluster.

6.4.6.34 Enhanced CPU Policies

Check Item

Check whether the current cluster version and the target version support the 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 6-20 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
Clusters of v1.21	Not supported
Clusters of v1.23 and later	Supported

6.4.6.35 Health of Worker Node Components

Check Item

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

Solution

If a worker node component malfunctions, log in to the node to check the status of the component and rectify the fault.

6.4.6.36 Health of Master Node Components

Check Item

Check whether the Kubernetes, container runtime, and network components of the master nodes are healthy.

Solution

If a master node component malfunctions, contact technical support.

6.4.6.37 Memory Resource Limit of Kubernetes Components

Check Item

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

Solution

- Solution 1: Reduce Kubernetes resources.
- Solution 2: [Scale out the cluster](#).

6.4.6.38 Discarded Kubernetes APIs

Check Item

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

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

6.4.6.39 IPv6 Capabilities of a CCE Turbo Cluster

Check Item

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.

6.4.6.40 Node NetworkManager

Check Item

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 problem persists after NetworkManager is restarted, contact technical support.

6.4.6.41 Node ID File

Check Item

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.

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

6.4.6.42 Node Configuration Consistency

Check Item

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

- /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 6-21 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

Component	Configuration File	Parameter	Upgrade Version
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.

6.4.6.43 Node Configuration File

Check Item

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

Contact technical support to restore the configuration file and then perform the upgrade.

6.4.6.44 CoreDNS Configuration Consistency

Check Item

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 kubectl, see [Connecting to a Cluster Using kubectl](#).

Step 2 Obtain the Corefile that takes effect currently.

```
kubectl 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 record (depending on Python 3).

```
latest_release=`kubectl get secret -nkube-system -l owner=helm -l name=cceaddon-coredns --sort-by=.metadata.creationTimestamp | awk 'END{print $1}'`
kubectl get secret -nkube-system $latest_release -o jsonpath='{.data.release}' | base64 -d | base64 -d | gzip -d | python -m json.tool | python -c "
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 6-8 Viewing output differences

```
[root@paas]# diff corefile_now.txt corefile_record.txt -y;echo
.:5353 {
  bind {$POD_IP}
  cache 31
  errors
  health {$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 {$POD_IP}:9153
  forward . /etc/resolv.conf
  reload
}
.:5353 {
  bind {$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
}
```

Step 5 Return to the CCE console and click the cluster name to go to the cluster console. On the **Add-ons** page, select CoreDNS and click **Upgrade**.

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

- Set **parameterSyncStrategy** to **force**. Manually enter the differential configuration. For details, see [CoreDNS](#).
- If **parameterSyncStrategy** is set to **inherit**, differentiated configurations are automatically inherited. The system automatically parses, identifies, and inherits differentiated parameters.

```
{  
  "parameterSyncStrategy": "force",  
  "servers": [  
    {  
      "plugins": [  
        {  
          "name": "bind",  
          "parameters": "${POD_IP}"  
        },  
        {  
          "configBlock": "servfail 5s",  
          "name": "cache",  
          "parameters": 31  
        }  
      ]  
    }  
  ]  
}
```

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

```
kubectl 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, you are advised to use the parameter configuration function of CCE add-on management to modify the Corefile configuration to avoid differences.

----End

6.4.6.45 sudo Commands of a Node

Check Items

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
```
- 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
```

6.4.6.46 Key Commands of Nodes

Check Items

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

Solution

- Scenario 1: The package manager command fails to be executed.
The **rpm** or **dpkg** command fails to be executed. Log in to the node and check whether the following commands are available:
 - rpm:

```
rpm -qa
```
 - dpkg:

```
dpkg -l
```
- 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
```

6.4.6.47 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.

6.4.6.48 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.

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 **Networking** 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_id>` 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 **Configurations 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 **Configurations 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.

----End

6.4.6.49 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.
 - The soft link of **/var/lib/kubelet** to **/mnt/paas/kubernetes/kubelet** is created for CCE by default. Check whether it has been manually modified.
- Shared disk
 - By default, **/mnt/paas/** is mounted to CCE nodes. Check whether **/mnt** has been manually mounted.

- The soft link of `/var/lib/kubelet` to `/mnt/paas/kubernetes/kubelet`, or `/var/lib/docker` or `containerd` to `/mnt/paas/runtime` is created for CCE by default. Check whether the soft 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 6-9 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 link.

6.4.6.50 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
```

6.4.6.51 Private IPv4 Addresses of Load Balancers

Check Item

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 search for 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 **Networking** in the navigation pane, click the **Services** tab, and 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.

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

6.4.6.52 Historical Upgrade Records

Check Items

Check whether the source version of the cluster is earlier than v1.11 and the target version is later than v1.23.

Solution

If the source version of the cluster is earlier than v1.11, it is risky to upgrade the cluster to a version later than v1.23. In this case, contact technical support.

6.4.6.53 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

If the CIDR block of the cluster management plane is different from that configured on the backbone network, contact technical support.

6.4.6.54 GPU Add-on

Check Item

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 and click the cluster name to access the cluster console. Choose **Add-ons** in the navigation pane to view the GPU add-on.

 **NOTE**

gpu-beta is the same as **gpu-device-plugin**. **gpu-beta** is renamed **gpu-device-plugin** in versions later than 2.0.0.

- Step 2** Click **Upgrade** of the add-on to view the target version and driver configuration 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

6.4.6.55 Nodes' System Parameter Settings

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.

6.4.6.56 Residual Package Versions

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

6.4.6.57 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.

6.4.6.58 Node Swap

Check Items

Check whether swap has been enabled on cluster 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. Run the **swapoff -a** command to disable swap.

6.4.6.59 nginx-ingress Upgrade

Check Item

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.

Fault Locating

For an 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=[] newV
alue=[{IP: Hostname: Ports:[]}] {IP: Hostname: Ports:[]}]
```

If the fault persists, contact technical support personnel.

----End

Solution

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.

Possible Causes

The nginx-ingress add-on is developed based on the Nginx Ingress Controller template and image of the open source community.

For the Nginx Ingress Controller of an earlier version (community version v0.49 or earlier, corresponding to CCE nginx-ingress version v1.x.x), the Ingress type is not specified as nginx during Ingress creation, which is, **kubernetes.io/ingress.class: nginx** is not added to annotations. This Ingress can also be managed by Nginx Ingress Controller. For details, see the [GitHub code](#).

For the Nginx Ingress Controller of a later version (community version v1.0.0 or later, corresponding to CCE nginx-ingress version 2.x.x), if the Ingress type is not specified as nginx during Ingress creation, this Ingress will be ignored by the Nginx Ingress Controller and the Ingress rules become invalid. The services will be interrupted. For details, see the [GitHub code](#).

Related link: <https://github.com/kubernetes/ingress-nginx/pull/7341>

You can specify the Ingress type in either of the following ways:

- Add the **kubernetes.io/ingress.class: nginx** annotation to the Ingresses.
- Use spec. Set the **.spec.ingressClassName** field to **nginx**. IngressClass resources are required.

An example is as follows:

```
apiVersion: networking.k8s.io/v1beta1
kind: Ingress
metadata:
  name: test
  namespace: default
  annotations:
    kubernetes.io/ingress.class: nginx
spec:
  ingressClassName: nginx
  rules:
  ...
status:
  loadBalancer: {}
```

6.5 Managing a Cluster

6.5.1 Cluster Configuration Management

Scenario

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

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 **Clusters**.
- Step 2** Click  next to the target cluster.
- Step 3** On the **Manage Components** page on the right, change the values of the Kubernetes parameters listed in the following table.

Table 6-22 kube-apiserver parameters

Parameter	Description	Value
default-not-ready-toleration-seconds	Tolerance time when a node is in the NotReady state. By default, this tolerance is added to each pod.	Default: 300s
default-unreachable-toleration-seconds	Tolerance time when a node is in the unreachable state. By default, this tolerance is added to each pod.	Default: 300s
max-mutating-requests-inflight	Maximum number of concurrent mutating requests. When the value of this parameter is exceeded, the server rejects requests. The value 0 indicates no limitation. This parameter is related to the cluster scale. You are advised not to change the value.	Manual configuration is no longer supported since cluster v1.21. The value is automatically specified based on the cluster scale. <ul style="list-style-type: none"> ● 200 for clusters with 50 or 200 nodes ● 500 for clusters with 1,000 nodes ● 1000 for clusters with 2,000 nodes

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 0 indicates no limitation. 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"> ● 400 for clusters with 50 or 200 nodes ● 1000 for clusters with 1,000 nodes ● 2000 for clusters with 2,000 nodes
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>Default: 30000-32767</p> <p>Value range: Min > 20105 Max < 32768</p>
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 supported only by 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>Value range: Min ≥ 1s Max ≤ 1 hour</p>
feature-gates: ServerSideApply	<p>Whether to enable ServerSideApply of kube-apiserver. For details, see Server-Side Apply.</p> <p>This parameter is supported only by clusters of v1.19.16-r30, v1.21.10-r10, v1.23.8-r10, v1.25.3-r10, and later versions.</p>	<p>Default: true</p>
support-overload	<p>Cluster overload control. If enabled, concurrent requests are dynamically controlled based on the resource pressure of master nodes to keep them and the cluster available.</p> <p>This parameter is supported only by clusters of v1.23 or later.</p>	<ul style="list-style-type: none"> ● false: Overload control is disabled. ● true: Overload control is enabled.

Table 6-23 kube-scheduler parameters

Parameter	Description	Value
kube-api-qps	Query per second (QPS) to use while talking with kube-apiserver.	<ul style="list-style-type: none"> If the number of nodes in a cluster is less than 1000, the default value is 100. If a cluster contains 1000 or more nodes, the default value is 200.
kube-api-burst	Burst to use while talking with kube-apiserver.	<ul style="list-style-type: none"> If the number of nodes in a cluster is less than 1000, the default value is 100. If a cluster contains 1000 or more nodes, the default value is 200.
enable-gpu-share	<p>Whether to enable GPU sharing. This parameter is supported only by clusters of v1.23.7-r10, v1.25.3-r0, and later.</p> <ul style="list-style-type: none"> When disabled, ensure that pods in the cluster do not use the shared GPU (that is, the annotation of cce.io/gpu-decision does not exist in pods). When enabled, ensure that the annotation of cce.io/gpu-decision exists in pods that use GPU resources in the cluster. 	Default: true

Table 6-24 kube-controller-manager parameters

Parameter	Description	Value
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

Parameter	Description	Value
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
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. NOTE 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
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.	<ul style="list-style-type: none"> • If the number of nodes in a cluster is less than 1000, the default value is 100. • If a cluster contains 1000 or more nodes, the default value is 200.

Parameter	Description	Value
kube-api-burst	Burst to use while talking with kube-apiserver.	<ul style="list-style-type: none"> If the number of nodes in a cluster is less than 1000, the default value is 100. If a cluster contains 1000 or more nodes, the default value is 200.
terminated-pod-gc-threshold	<p>Number of terminated pods that can exist before the terminated pod garbage collector starts deleting terminated pods.</p> <p>If ≤ 0, the terminated pod garbage collector is disabled.</p>	Default: 1000

Table 6-25 eni parameters (supported only by CCE Turbo clusters)

Parameter	Description	Value
nic-minimum-target	Minimum number of ENIs bound to a node at the cluster level	Default: 10
nic-maximum-target	Maximum number of ENIs pre-bound to a node at the cluster level	Default: 0
nic-warm-target	Number of ENIs pre-bound to a node at the cluster level	Default: 2
nic-max-above-warm-target	Reclaim number of ENIs pre-bound to a node at the cluster level	Default: 2

Table 6-26 Extended controller configuration parameters (supported only by clusters of v1.21 and later)

Parameter	Description	Value
enable-resource-quota	<p>Whether to automatically create a resource quota object when creating a namespace.</p> <ul style="list-style-type: none"> false: no auto creation true: auto creation enabled For details about the resource quota defaults, see Setting a Resource Quota. 	Default: false

Step 4 Click **OK**.

----End

References

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

6.5.2 Cluster Overload Control

Scenario

If enabled, concurrent requests are dynamically controlled based on the resource pressure of master nodes to keep them and the cluster available.

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 go to an existing cluster whose version is v1.23 or later.

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

----End

Disabling Cluster Overload Control

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

Step 2 On the **Cluster Information** page, click **Manage** in the upper right corner.

Step 3 Set **support-overload** to **false** under **kube-apiserver**.

Step 4 Click **OK**.

----End

6.5.3 Changing Cluster Scale

Scenario


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

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.
- During the specifications change, master nodes will be powered off and on, and the cluster cannot run properly. Perform the change during off-peak hours.
- 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.
- Change failures will trigger a cluster rollback to the normal state. If the rollback fails, submit a service ticket.

Procedure

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

Step 2 Click  next to the cluster whose specifications need to be modified.

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 left corner to view the cluster change history. The status changes from **Executing** to **Successful**, indicating that the cluster specifications are successfully changed.

----End

6.5.4 Deleting a Cluster

Precautions

- 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)
- If you delete a cluster that is not running (for example, unavailable), associated resources, such as storage and networking resources, will remain.

Deleting a 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 Click  next to the cluster to be deleted.

Step 3 In the displayed **Delete Cluster** dialog box, select the resources to be released.

- 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 1,000 files in the OBS bucket, manually clear the files and then delete the cluster.
- Delete network resources such as load balancers in a cluster. (Only automatically created load balancers will be deleted).

Step 4 Click **Yes** to start deleting the cluster.

The delete operation takes 1 to 3 minutes to complete.

----End

6.5.5 Hibernating and Waking Up a Cluster

Scenario

If you do not need to use a cluster temporarily, you are advised to hibernate the cluster.

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.

Constraints

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 **Clusters**.

Step 2 Click  next to the cluster to be hibernated.

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


After a cluster is hibernated, the billing of master node resources will stop. 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](#).

Most nodes are no longer billed after they are stopped, excluding certain types of ECSs with a local disk attached (such as a disk-intensive or ultra-high I/O ECS)..

----End

Waking Up a Cluster

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

Step 2 Click  next to the cluster to be woken up.

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.

NOTE

After the cluster is woken up, the cluster management fee continues to be billed.

----End

7 Nodes

7.1 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**, 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.

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 Docker

- In clusters of v1.15.6 or earlier, the underlying file storage system uses the XFS format.
- In clusters of v1.15.11 or later, after a node is created or reset, the underlying file storage system uses 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 Docker underlying storage file used by the current node.

Paas User and User Group

When you create a node in a CCE cluster, a paas user or user group is created on the node by default. CCE components and CCE add-ons on a node run as a non-root user (paas user/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 7-1 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 running properly. Instances in this state no longer provide services. In this case, perform the operations in Resetting a Node .
Creating	Intermediate state	The node has been created but is not running.
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 occurred.

Status	Status Attribute	Description
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 Resetting a Node .

7.2 Container Engine

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 7-2 Comparison between container engines

Item	containerd	Docker
Tracing	kubelet --> CRI plugin (in the containerd process) --> containerd	<ul style="list-style-type: none"> Docker (Kubernetes 1.23 and earlier versions): kubelet --> dockershim (in the kubelet process) --> docker --> containerd Docker (community solution for Kubernetes 1.24 and later versions): kubelet --> cri-dockerd (kubelet uses CRI to connect to cri-dockerd) --> docker--> containerd
Command	crictl	docker
Kubernetes CRI	Native support	Support through dockershim or cri-dockerd
Pod startup delay	Low	High

Item	containerd	Docker
kubelet CPU/ memory usage	Low	High
Runtime's CPU/memory usage	Low	High

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 7-3 Image-related commands

No.	Docker Command	containerd Command	Remarks
1	docker images [Option] [Image name[:Tag]]	crictl images [Option] [Image name[:Tag]]	List local images.
2	docker pull [Option] <i>Image name</i> [:Tag] @DIGEST]	crictl pull [Option] <i>Image name</i> [:Tag]@DIGEST]	Pull images.
3	docker push	None	Pushing images.
4	docker rmi [Option] <i>Image...</i>	crictl rmi [Option] <i>Image ID...</i>	Delete a local image.
5	docker inspect <i>Image ID</i>	crictl inspecti <i>Image ID</i>	Check images.

Table 7-4 Container-related commands

No.	Docker Command	containerd Command	Remarks
1	docker ps [Option]	crictl ps [Option]	List containers.
2	docker create [Option]	crictl create [Option]	Create a container.
3	docker start [Option] <i>Container ID...</i>	crictl start [Option] <i>Container ID...</i>	Start a container.
4	docker stop [Option] <i>Container ID...</i>	crictl stop [Option] <i>Container ID...</i>	Stop a container.
5	docker rm [Option] <i>Container ID...</i>	crictl rm [Option] <i>Container ID...</i>	Delete a container.

No.	Docker Command	containerd Command	Remarks
6	docker attach [Option] <i>Container ID</i>	crictl attach [Option] <i>Container ID</i>	Connect to a container.
7	docker exec [Option] <i>Container ID Startup command [Parameter...]</i>	crictl exec [Option] <i>Container ID Startup command [Parameter...]</i>	Access the container.
8	docker inspect [Option] <i>Container name ID...</i>	crictl inspect [Option] <i>Container ID...</i>	Query container details.
9	docker logs [Option] <i>Container ID</i>	crictl logs [Option] <i>Container ID</i>	View container logs.
10	docker stats [Option] <i>Container ID...</i>	crictl stats [Option] <i>Container ID</i>	Check the resource usage of the container.
11	docker update [Option] <i>Container ID...</i>	crictl update [Option] <i>Container ID...</i>	Update container resource limits.

Table 7-5 Pod-related commands

No.	Docker Command	containerd Command	Remarks
1	None	crictl pods [Option]	List pods.
2	None	crictl inspectp [Option] <i>Pod ID...</i>	View pod details.
3	None	crictl start [Option] <i>Pod ID...</i>	Start a pod.
4	None	crictl runp [Option] <i>Pod ID...</i>	Run a pod.
5	None	crictl stopp [Option] <i>Pod ID...</i>	Stop a pod.
6	None	crictl rmp [Option] <i>Pod ID...</i>	Delete a pod.

 **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 1.23 and earlier versions):
kubenet --> docker shim (in the kubenet process) --> docker --> containerd
- Docker (community solution for Kubernetes v1.24 or later):
kubenet --> cri-dockerd (kubenet uses CRI to connect to cri-dockerd) --> docker--> containerd
- containerd:
kubenet --> 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.**

7.3 Creating a Node

Prerequisites

- At least one cluster has been created.
- A key pair has been created for identity authentication upon remote node login.

Constraints

- The node has at least 2 vCPUs and 4 GiB of memory.
- To ensure node stability, a certain number of CCE node resources will be reserved for Kubernetes components (such as kubenet, kube-proxy, and docker) based on the node specifications. 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. Otherwise, services may be unavailable. For example, the NIC named **gw_11cbf51a@eth0** on the node is the container network gateway and cannot be modified.
- During the node creation, software packages are downloaded from OBS using the domain name. Use a private DNS server to resolve the OBS domain name, and configure the DNS server address of 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.
- Once a node is created, its AZ cannot be changed.

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 on the left, choose **Nodes**. On the page displayed, click **Create Node**. In the **Node Settings** step, set node parameters by referring to the following table.

Compute Settings

You can configure the specifications and OS of a cloud server, on which your containerized applications run.

Table 7-6 Configuration parameters

Parameter	Description
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 Random 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>CCE cluster:</p> <ul style="list-style-type: none"> ECS (VM): Containers run on ECSs. ECS (physical): Containers run on servers using the QingTian architecture. <p>CCE Turbo cluster:</p> <ul style="list-style-type: none"> ECS (VM): Containers run on ECSs. Only the ECSs that can be bound with multiple NICs are supported. ECS (physical): Containers run on servers using the QingTian architecture.
Container Engine	<p>CCE clusters support Docker and containerd in some scenarios.</p> <ul style="list-style-type: none"> VPC network clusters of v1.23 and later versions support containerd. Tunnel network clusters of v1.23.2-r0 and later versions support containerd. For a CCE Turbo cluster, both Docker and containerd are supported.
Specifications	<p>Select node specifications that best fit your service needs.</p> <p>The available node flavors vary depending on AZs. Obtain the flavors displayed on the console.</p>

Parameter	Description
OS	<p>Select an OS type. Different types of nodes support different OSs.</p> <ul style="list-style-type: none"> • Public image: Select a public image for the node. <p>NOTE</p> <ul style="list-style-type: none"> • 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.
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>A node name must start with a lowercase letter and cannot end with a hyphen (-). Only digits, lowercase letters, and hyphens (-) are allowed.</p>
Login Mode	<ul style="list-style-type: none"> • Key Pair Select the key pair used to log in to the node. You can select a shared key. <p>A key pair is used for identity authentication when you remotely log in to a node. If no key pair is available, click Create Key Pair.</p>

Storage Settings

Configure storage resources on a node for the containers running on it. Set the disk size according to site requirements.

Table 7-7 Configuration parameters

Parameter	Description
System Disk	<p>System disk used by the node OS. The value ranges from 40 GiB to 1,024 GiB. The default value is 50 GiB.</p> <p>Encryption: System disk encryption safeguards your data. Snapshots generated from encrypted disks and disks created using these snapshots automatically inherit the encryption setting. This function is available only in certain regions.</p> <ul style="list-style-type: none"> • Encryption is not selected by default. • After selecting Encryption, you can select an existing key in the displayed dialog box. If no key is available, click View Key List and create a key. After the key is created, click the refresh icon next to the Encryption text box.

Parameter	Description
Data Disk	<p>At least one data disk is required for the container runtime and kubelet. The data disk cannot be deleted or uninstalled. Otherwise, the node will be unavailable.</p> <ul style="list-style-type: none"> • First data disk: used for container runtime and kubelet components. The value ranges from 20 GiB to 32,768 GiB. The default value is 100 GiB. • Other data disks: You can set the data disk size to a value ranging from 10 GB to 32,768 GiB. The default value is 100 GiB. <p>NOTE If the node flavor is disk-intensive or ultra-high I/O, one data disk can be a local disk. 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.</p> <p>Advanced Settings Click Expand to configure the following parameters:</p> <ul style="list-style-type: none"> • Data Disk Space Allocation: After selecting Set Container Engine Space, you can specify the proportion of the space for the container engine, image, and temporary storage on the data disk. The container engine space is used to store the working directory, container image data, and image metadata for the container runtime. The remaining space of the data disk is used for pod configuration files, keys, and EmptyDir. For details about how to allocate data disk space, see Data Disk Space Allocation. • Encryption: Data disk encryption safeguards your data. Snapshots generated from encrypted disks and disks created using these snapshots automatically inherit the encryption setting. This function is available only in certain regions. <ul style="list-style-type: none"> - Encryption is not selected by default. - After selecting Encryption, you can select an existing key in the displayed dialog box. If no key is available, click View Key List and create a key. After the key is created, click the refresh icon next to the Encryption text box. <p>Adding Multiple Data Disks A maximum of four data disks can be added. By default, raw disks are created without any processing. You can also click Expand and select any of the following options:</p> <ul style="list-style-type: none"> • Default: By default, a raw disk is created without any processing. • Mount Disk: The data disk is attached to a specified directory. • Use as PV: applicable to scenarios in which there is a high performance requirement on PVs. The node.kubernetes.io/local-storage-persistent label is added to the node with PV configured. The value is linear or striped.

Parameter	Description
	<ul style="list-style-type: none"> • Use as ephemeral volume: applicable to scenarios in which there is a high performance requirement on EmptyDir. <p>NOTE</p> <ul style="list-style-type: none"> • 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. • 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. <p>Local Persistent Volumes (Local PVs) and Local EVs support the following write modes:</p> <ul style="list-style-type: none"> • 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, allowing data to be concurrently read and written. A storage pool consisting of striped volumes cannot be scaled-out. This option can be selected only when multiple volumes exist.

Network Settings

Configure networking resources to allow node and containerized application access.

Table 7-8 Configuration parameters

Parameter	Description
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.
EIP	An ECS without a bound EIP cannot access the Internet or be accessed by public networks. The default value is Do not use . Use existing and Auto create are supported.

Advanced Settings

Configure advanced node capabilities such as labels, taints, and startup command.

Table 7-9 Advanced configuration parameters

Parameter	Description
Kubernetes Label	<p>A key-value pair added to a Kubernetes object (such as a pod). 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 Labels and Selectors.</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"> • 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. <p>For details, see Managing Node Taints.</p> <p>NOTE 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 Maximum Number of Pods That Can Be Created on a Node.</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 Add ECS Group to create one. After the ECS group is created, click the refresh button.</p>

Parameter	Description
Pre-installation Command	Enter commands. A maximum of 1,000 characters are allowed. 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 Command	Enter commands. A maximum of 1,000 characters are allowed. The script will be executed after Kubernetes software is installed and will not affect the installation. NOTE Do not run the reboot command in the post-installation script to restart the system immediately. To restart the system, run the shutdown -r 1 command to delay the restart for one minute.
Agency	An agency is created by the account 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. If no agency is available, click Create Agency on the right to create one.

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

7.4 Adding Nodes for Management

Scenario

In CCE, you can create a node ([Creating a Node](#)) or add existing nodes (ECSs or) to your cluster.

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.
- 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 management. Ensure that the information 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.

Constraints

- The cluster version must be 1.15 or later.
- If a password or key has been set when the original VM node was created, reset the password or key during management. The original password or key will become invalid.
- 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.

Prerequisites

A cloud server that meets the following conditions can be accepted:

- 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 specifications, AZ, and data disk configuration can be added in batches.

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 **Accept Node** in the upper right corner.

Step 3 Specify node parameters.

Compute Settings

Table 7-10 Configuration parameters

Parameter	Description
Specifications	Click Select Cloud Server 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.
Container Engine	CCE clusters support Docker and containerd in some scenarios. <ul style="list-style-type: none"> VPC network clusters of v1.23 and later versions support containerd. Tunnel network clusters of v1.23.2-r0 and later versions support containerd. For a CCE Turbo cluster, both Docker and containerd are supported.
OS	Public image: Select an OS for the node.
Login Mode	<ul style="list-style-type: none"> 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 Key Pair.

Storage Settings

Configure storage resources on a node for the containers running on it.

Table 7-11 Configuration parameters

Parameter	Description
System Disk	Directly use the system disk of the cloud server.

Parameter	Description
Data Disk	<p>At least one data disk is required for the container runtime and kubelet. The data disk cannot be deleted or uninstalled. Otherwise, the node will be unavailable.</p> <p>Click Expand and select Allocate Disk Space to define the disk space occupied by the container runtime to store the working directories, container image data, and image metadata. For details about how to allocate data disk space, see Data Disk Space Allocation.</p> <p>For other data disks, a raw disk is created without any processing by default. You can also click Expand and select Mount Disk to mount the data disk to a specified directory.</p>

Advanced Settings

Table 7-12 Advanced configuration parameters

Parameter	Description
Kubernetes Label	<p>Click Add 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 Labels and Selectors.</p>
Taint	<p>This field 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"> • 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. <p>NOTICE</p> <ul style="list-style-type: none"> • 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.

Parameter	Description
Max. Pods	Maximum number of pods that can run on the node, including the default system pods. This limit prevents the node from being overloaded with pods.
Pre-installation Command	Enter commands. A maximum of 1,000 characters are allowed. 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 Command	Enter commands. A maximum of 1,000 characters are allowed. The script will be executed after Kubernetes software is installed and will not affect the installation.

Step 4 Click **Next: Confirm**. Click **Submit**.

----End

7.5 Logging In to a Node

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 cloud 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 cloud.
- 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 7-13 Linux ECS login modes

EIP Binding	On-Premises OS	Connection Method
Yes	Windows	Use a remote login tool, such as PuTTY or Xshell. <ul style="list-style-type: none"> • SSH key authentication: Login Using an SSH Key
Yes	Linux	Run commands. <ul style="list-style-type: none"> • SSH key authentication: Login Using an SSH Key
Yes/No	Windows/ Linux	Remote login using the management console: Login Using VNC

7.6 Management Nodes

7.6.1 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.
- Affinity and anti-affinity between a workload and node:
 - Different workloads have different resource requirements such as CPU, memory, and I/O. If a workload consumes too many resources in a cluster, other workloads in the same cluster may fail to run properly. 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 efficient O&M, you can add a module label to each node so that each module can be deployed on the

corresponding node. In this way, modules do not interfere with each other and microservices can be easily maintained on their nodes.

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 7-14](#).

 **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 7-14 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 false indicates that the node is not a bare metal node.
node.kubernetes.io/instance-type	Node specifications
kubernetes.io/arch	Node processor architecture
kubernetes.io/hostname	Node name
kubernetes.io/os	OS type
node.kubernetes.io/subnetid	ID of the subnet where the node is located.
os.architecture	Node processor architecture For example, amd64 indicates a AMD64-bit processor.
os.name	Node OS name
os.version	Node OS kernel version
node.kubernetes.io/container- engine	Container engine used by the node.
accelerator	GPU node labels.

Key	Description
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.

Step 2 Click the cluster name, access the cluster details page, and choose **Nodes** in the navigation pane. On the page displayed, select a node and click **Labels and Taints** above the list.

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

7.6.2 Managing Node Taints

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 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, run the following command with a hyphen (-) added after **NoSchedule**:

```
$ 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>
...
```

On the CCE console, you can also manage taints of a node in batches.

Step 1 Log in to the CCE console.

Step 2 Click the cluster name, access the cluster details page, and choose **Nodes** in the navigation pane. On the page displayed, select a node and click **Labels and Taints** above the list.

Step 3 In the displayed dialog box, click **Add Operation** under **Batch Operation**, choose **Add/Update**, and select **Taint**.

Enter the key and value of the taint to be added, select the taint effect, and click **OK**.

Step 4 After the taint is added, check the added taint in node data.

----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.

Node Scheduling Settings

To configure scheduling, log in to the CCE console, click the cluster, choose **Nodes** in the navigation pane, and click **More > Disable Scheduling** in the **Operation** column of a node in the node list.

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, 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.

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"
```

7.6.3 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.

Constraints

- For CCE clusters and CCE Turbo clusters, the version must be v1.13 or later to support node resetting.

Precautions

- 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.
- Resetting a node will cause PVC/PV data loss for the **local PV** associated with the node. These PVCs and PVs cannot be restored or used again. In this scenario, the pod that uses the local PV is evicted from the reset node. A new pod is created and stays 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 is always in the creating state because the underlying logical volume corresponding to the PVC does not exist.

Procedure

The new console allows you to reset nodes in batches.

Step 1 Log in to the CCE console and click the cluster name to access the cluster console.

Step 2 Click the cluster name to access the cluster console. Choose **Nodes** in the navigation pane, and select one or multiple nodes to be reset in the list. Choose **More > Reset Node**.

Step 3 In the displayed dialog box, click **Next**.

- For nodes in the DefaultPool node pool, the parameter setting page is displayed. Set the parameters by referring to [Step 4](#).
- For a node you create in a node pool, resetting the node does not support parameter configuration. You can directly use the configuration image of the node pool to reset the node.

Step 4 Specify node parameters.

Compute Settings

Table 7-15 Configuration parameters

Parameter	Description
Specifications	Specifications cannot be modified when you reset a node.
Container Engine	CCE clusters support Docker and containerd in some scenarios. <ul style="list-style-type: none"> • VPC network clusters of v1.23 and later versions support containerd. Tunnel network clusters of v1.23.2-r0 and later versions support containerd. • For a CCE Turbo cluster, both Docker and containerd are supported.
OS	Public image: Select an OS for the node.
Login Mode	<ul style="list-style-type: none"> • 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 Key Pair.

Storage Settings

Configure storage resources on a node for the containers running on it.

Table 7-16 Configuration parameters

Parameter	Description
System Disk	Directly use the system disk of the cloud server.

Parameter	Description
Data Disk	<p>At least one data disk is required for the container runtime and kubelet. The data disk cannot be deleted or uninstalled. Otherwise, the node will be unavailable.</p> <p>Click Expand and select Allocate Disk Space to define the disk space occupied by the container runtime to store the working directories, container image data, and image metadata. For details about how to allocate data disk space, see Data Disk Space Allocation.</p> <p>For other data disks, a raw disk is created without any processing by default. You can also click Expand and select Mount Disk to mount the data disk to a specified directory.</p>

Advanced Settings

Table 7-17 Advanced configuration parameters

Parameter	Description
Kubernetes Label	<p>Click Add 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 Labels and Selectors.</p>
Taint	<p>This field 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"> • 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. <p>NOTICE</p> <ul style="list-style-type: none"> • 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.

Parameter	Description
Max. Pods	Maximum number of pods that can run on the node, including the default system pods. This limit prevents the node from being overloaded with pods.
Pre-installation Command	Enter commands. A maximum of 1,000 characters are allowed. 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 Command	Enter commands. A maximum of 1,000 characters are allowed. The script will be executed after Kubernetes software is installed and will not affect the installation.

Step 5 Click **Next: Confirm**.

Step 6 Click **Submit**.

----End

7.6.4 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.

Constraints

- Nodes can be removed only when the cluster is in the **Available** or **Unavailable** status.
- A CCE node can be removed only when it is in the **Active**, **Abnormal**, or **Error** status.
- A CCE node in the **Active** status 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](#).
- Removing a node will cause PVC/PV data loss for the **local PV** associated with the node. These PVCs and PVs cannot be restored or used again. In this scenario, the pod that uses the local PV is evicted from the node. A new pod

is created and stays 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.

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 and click the cluster name to access the cluster console.
- Step 2** Choose **Nodes** from the navigation pane and choose **More > Remove** in the **Operation** column of the target node.
- Step 3** 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

7.6.5 Synchronizing Data with 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.

Constraints

- Data, including the VM status, ECS names, number of CPUs, size of memory, ECS specifications, and public IP addresses, can be synchronized.
- Data, such as the OS and image ID, cannot be synchronized. (Such parameters cannot be modified on the ECS console.)

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.

Step 3 Choose **More > Sync Server Data** next to the node.

After the synchronization is complete, the **ECS data synchronization requested** message is displayed in the upper right corner.

----End

7.6.6 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.

Constraints

- VM nodes that are being used by CCE do not support deletion on the ECS page.
- Deleting a node will cause PVC/PV data loss for the **local PVs** associated with the node. These PVCs and PVs cannot be restored or used again. In this scenario, the pod that uses the local PV is evicted from the node. A new pod

is created and stays 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.

Precautions

- 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 and click the cluster name to access the cluster console.

Step 2 In the navigation pane, choose **Nodes**. In the same row as the node you will delete, choose **More > Delete**.

Step 3 In the **Delete Node** dialog box, click **Yes**.

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.

----End

7.6.7 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.

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 and click the cluster name to access the cluster console.

Step 2 In the navigation pane, choose **Nodes**. In the right pane, click the name of the node to be stopped.

Step 3 In the upper right corner of the ECS details page, click **Stop**. In the displayed dialog box, click **Yes**.

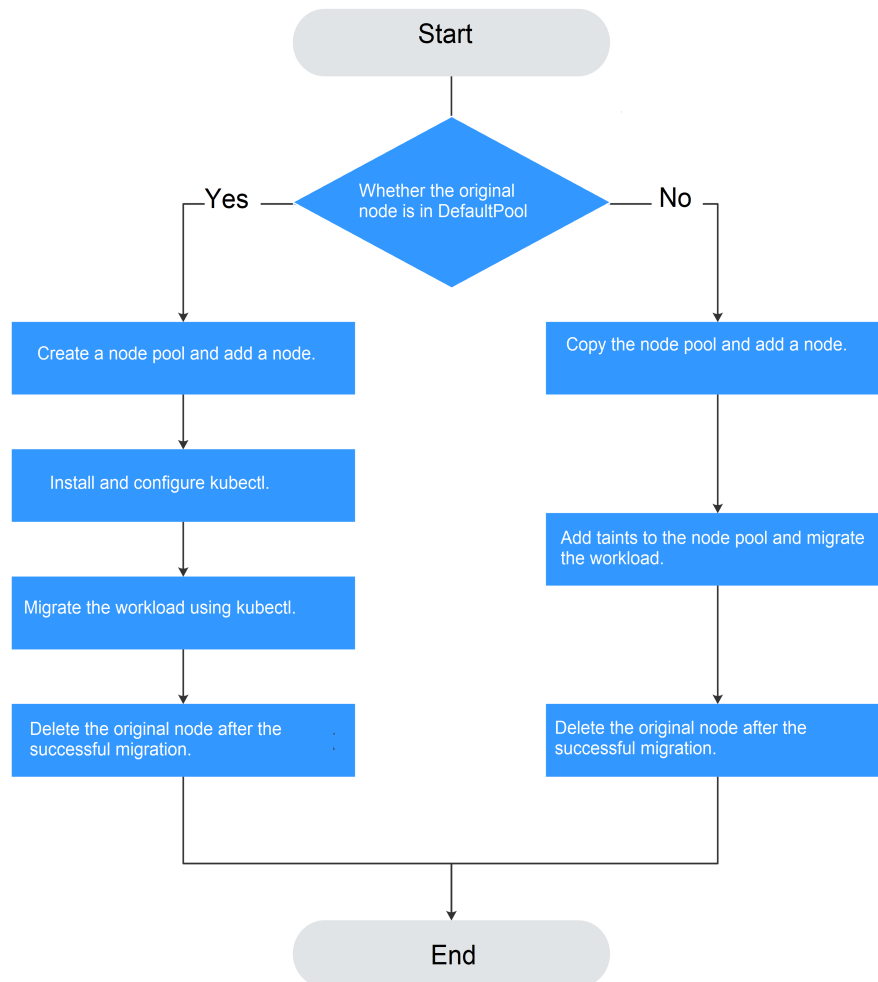
----End

7.6.8 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 7-1](#) shows the migration process.

Figure 7-1 Workload migration



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 pool. For details, see [Creating a Node Pool](#).

Step 2 On the node pool list page, click **View Node** in the **Operation** column of the target node pool. The IP address of the new node is displayed in the node list.

Step 3 Install and configure kubectl. For details, see [Connecting to a Cluster Using kubectl](#).

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 configure the affinity and anti-affinity policies of the new node. For details, see [Scheduling Policy \(Affinity/Anti-affinity\)](#).

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 runs properly, 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. For details, see [Copying a Node Pool](#).

Step 2 Click **View Node** in the **Operation** column 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 original node pool and configure **Taints**.
2. Enter the key and value of a taint. The options of **Effect** are **NoSchedule**, **PreferNoSchedule**, and **NoExecute**. Select **NoExecute** and click **Add**.
 - **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 the taint, delete the configured one.

3. Click **OK**.
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 configure the affinity and anti-affinity policies of the new node. For details, see [Scheduling Policy \(Affinity/Anti-affinity\)](#).

After the workload is 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 runs properly, delete the original node.

----End

7.7 Node O&M

7.7.1 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 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.

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 v2 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 7-18 Reservation rules for system components

Total Memory (TM)	Reserved Memory for System Components
$TM \leq 8 \text{ GB}$	0 MB
$8 \text{ GB} < TM \leq 16 \text{ GB}$	$[(TM - 8 \text{ GB}) \times 1024 \times 10\%]$ MB
$16 \text{ GB} < TM \leq 128 \text{ GB}$	$[8 \text{ GB} \times 1024 \times 10\% + (TM - 16 \text{ GB}) \times 1024 \times 6\%]$ MB
$TM > 128 \text{ GB}$	$(8 \text{ GB} \times 1024 \times 10\% + 112 \text{ GB} \times 1024 \times 6\% + (TM - 128 \text{ GB}) \times 1024 \times 2\%)$ MB

Table 7-19 Reservation rules for kubelet

Total Memory (TM)	Number of Pods	Reserved Memory for kubelet
$TM \leq 2 \text{ GB}$	None	$TM \times 25\%$
$TM > 2 \text{ GB}$	$0 < \text{Max. pods on a node} \leq 16$	700 MB
	$16 < \text{Max. pods on a node} \leq 32$	$[700 + (\text{Max. pods on a node} - 16) \times 18.75]$ MB
	$32 < \text{Max. pods on a node} \leq 64$	$[1024 + (\text{Max. pods on a node} - 32) \times 6.25]$ MB
	$64 < \text{Max. pods on a node} \leq 128$	$[1230 + (\text{Max. pods on a node} - 64) \times 7.80]$ MB
	$\text{Max. pods on a node} > 128$	$[1740 + (\text{Max. pods on a node} - 128) \times 11.20]$ 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 v2 for Reserving Node Memory

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 [Managing a Node Pool](#).

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 7-20 Rules for reserving node memory v2

Reserved for	Basic/Floating	Reservation	Used by
OS	Basic	400 MB (fixed)	OS service components such as sshd and systemd-journald.
	Floating (depending on the node memory)	25 MB/GB	Kernel
CCE	Basic	500 MB (fixed)	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 MB/pod containerd: 5 MB/pod	Container engine components when the number of pods increases NOTE 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 Table 7-24 .

Rules for Reserving Node CPU

Table 7-21 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.

7.7.2 Data Disk Space Allocation

This section describes how to allocate data disk space to nodes so that you can configure the data disk space accordingly.

Allocating Data Disk Space

When creating a node, configure data disks for the node. You can also click **Expand** and customize the data disk space allocation for the node.

- **Allocate 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 images, 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.
- **Allocate Pod Basesize:** 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 smaller 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.

Allocating Disk Space

For a node using a non-shared data disk (100 GB 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)**

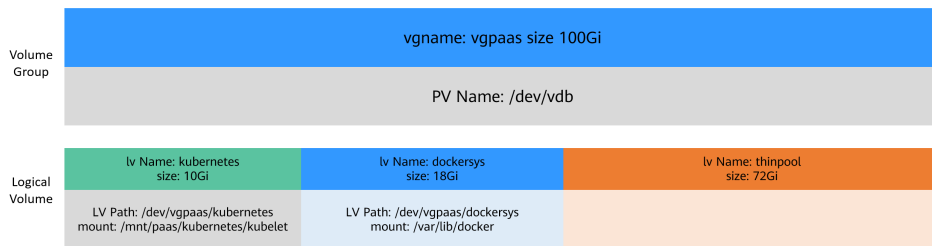
By default, the container engine and image space, occupying 90% of the data disk, can be divided into the following two parts:

 - 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.

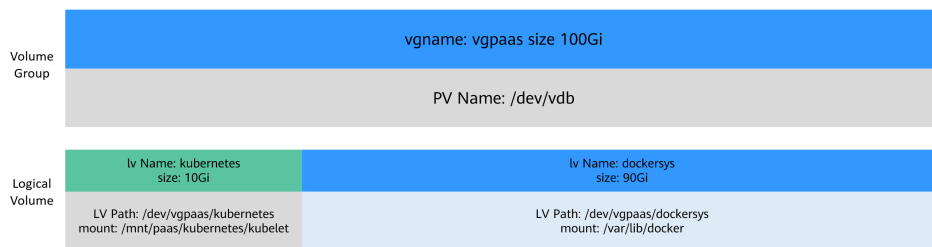
Figure 7-2 Space allocation for container engines of Device Mapper



- **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 7-3 Space allocation for container engines of OverlayFS



Allocating Basesize 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 GB.
- In OverlayFS mode, the pod container space is not limited by default.

NOTE

In the case of using Docker on EulerOS 2.9 nodes, **basesize** will not take effect if **CAP_SYS_RESOURCE** or **privileged** is configured for a container.

When configuring **basesize**, consider the maximum number of pods on a node. The container engine space should be greater than the total disk space used by containers. Formula: **the 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.

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 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.

Mapping Between OS and Container Storage Rootfs

Table 7-22 Node OSs and container engines in CCE clusters

OS	Container Storage Rootfs	Customized 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 container engine is Docker. The default value is 10 GB. Not supported when Rootfs is set to OverlayFS.
EulerOS 2.5	Device Mapper	Supported only when the container engine is Docker. The default value is 10 GB.
EulerOS 2.9	OverlayFS	Supported only by clusters of v1.19.16, v1.21.3, v1.23.3, and later. The container basesize is not limited by default. Not supported when the cluster versions are earlier than v1.19.16, v1.21.3, and v1.23.3.
Ubuntu 22.04	OverlayFS	Not supported.

Table 7-23 Node OSs and container engines in CCE Turbo clusters

OS	Container Storage Rootfs	Customized Basesize
CentOS 7.x	OverlayFS	Not supported.
Ubuntu 22.04	OverlayFS	Not supported.

OS	Container Storage Rootfs	Customized Basesize
EulerOS 2.9	ECS VMs use OverlayFS. ECS PMs use Device Mapper.	Supported only when Rootfs is set to OverlayFS and the container engine is Docker. The container basesize is not limited by default. Supported when Rootfs is set to Device Mapper and the container engine is Docker. The default value is 10 GB.

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 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.

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.

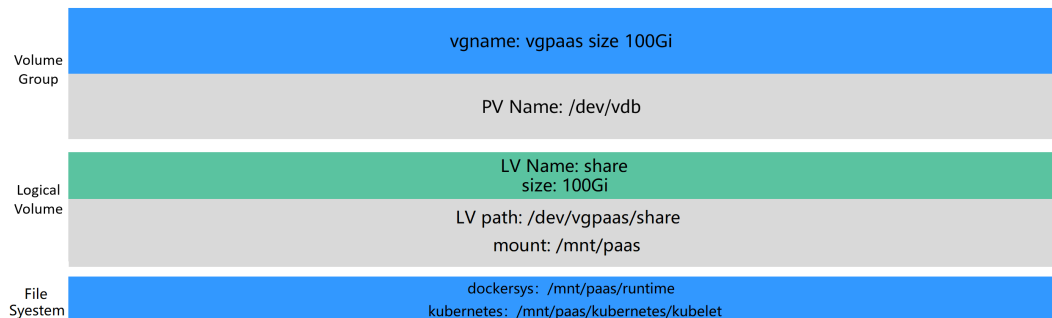
NOTICE

- 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 a 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.

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 GB) 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 7-4 Allocating the storage space of a shared data disk



7.7.3 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:

- For a cluster using the container tunnel network model, the value depends only on **the maximum number of pods on a node**.
- For clusters using the VPC network model, the value depends on **the maximum number of pods on a node** and **the minimum number of container IP addresses that can be allocated to a node**. It is recommended that the maximum number of pods on a node be less than or equal to the number of container IP addresses that can be allocated to the node. Otherwise, pods may fail to be scheduled.

- For CCE Turbo clusters using the Cloud Native Network 2.0 model, the value depends on **the maximum number of pods on a node** and **the minimum number of ENIs on a CCE Turbo cluster node**. It is recommended that the maximum number of pods on a node be less than or equal to the number of ENIs on the node. Otherwise, pods may fail to be scheduled.

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 (`alpha.cce/fixPoolMask`). If the pod uses the host network (**hostNetwork: true**), the pod does not occupy the IP address of the allocatable container network. For details, see [Container Network vs. Host Network](#).

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. If the pod uses the host network (**hostNetwork: true**), the pod does not occupy the IP address of the allocatable container network.

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.

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.

Table 7-24 lists the default maximum number of pods on a node based on node specifications.

Table 7-24 Default maximum number of pods on a node

Memory	Max. Pods
4 GiB	20
8 GiB	40
16 GiB	60

Memory	Max. Pods
32 GiB	80
64 GiB or above	110

Number of Node ENIs (CCE Turbo Clusters)

In a CCE Turbo cluster, ECSs 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.

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.

7.7.4 Migrating Nodes from Docker to containerd

Context

Kubernetes has removed dockershim from v1.24 and does not support Docker by default. CCE will continue to support Docker in v1.25 but just till v1.27. The following steps show you how to migrate nodes from Docker to containerd.

Prerequisites

- At least one cluster that supports containerd nodes has been created..
- 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 Engine](#).

Migrating a 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**. In the node list, select one or more nodes to be reset and choose **More > Reset Node**.
- Step 3** Set **Container Engine** to **containerd**. You can adjust other parameters as required or retain them as set during creation.
- Step 4** 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

Migrating a 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** On the **Compute Settings** area, set **Container Engine** to **containerd** and modify other parameters as required.
- 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

7.7.5 Node Fault Detection Policy

The node fault detection function depends on the [node-problem-detector \(npd\)](#) add-on. The add-on instances run on nodes and monitor nodes. This section describes how to enable node fault detection.

Prerequisites

The **npd** 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 on the left, choose **Nodes**. 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 **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.

----End

Customized Check Items

- Step 1** Log in to the CCE console and click the cluster name to access the cluster console.
- Step 2** Choose **Nodes** in the navigation pane. On the page displayed, 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%.
- **Check Period:** The default check period is 30 seconds. You can modify this parameter as required.
- **Troubleshooting Strategy:** After a fault occurs, you can select the strategies listed in the following table.

Table 7-25 Troubleshooting strategies

Troubleshooting Strategy	Effect
Prompting Exception	Kubernetes events are reported.
Disabling scheduling	Kubernetes events are reported and the NoSchedule taint is added to the node.
Evict Node Load	Kubernetes events are reported and the NoExecute 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 7-26 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: /dev/kmsg 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: /dev/kmsg Matching rule: "task \\S+:\\w+ blocked for more than \\w+ seconds\\."

Check Item	Function	Description
ReadOnlyFilesystem	<p>Check whether the Remount root filesystem read-only 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>Warning event</p> <p>Listening object: /dev/kmsg</p> <p>Matching rule: Remounting filesystem read-only</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 7-27 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"> • Default threshold: 10 restarts within 10 minutes • If Kubelet restarts for 10 times within 10 minutes, it indicates that the system restarts frequently and a fault alarm is generated. • Listening object: logs in the /run/log/journal directory
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 7-28 Checking system metrics

Check Item	Function	Description
Conntrack table full ConntrackFullProblem	Check whether the conntrack table is full.	<ul style="list-style-type: none"> • Default threshold: 90% • Usage: nf_conntrack_count • Maximum value: nf_conntrack_max
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"> • Default threshold: 90% • Source: <code>df -h</code> <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"> • Default threshold: 90% • Usage: the first value in /proc/sys/fs/file-nr • Maximum value: the third value in /proc/sys/fs/file-nr
Insufficient node memory MemoryProblem	Check whether memory is used up.	<ul style="list-style-type: none"> • Default threshold: 80% • Usage: MemTotal-MemAvailable in /proc/meminfo • Maximum value: MemTotal in /proc/meminfo
Insufficient process resources PIDProblem	Check whether PID process resources are exhausted.	<ul style="list-style-type: none"> • Default threshold: 90% • Usage: nr_threads in /proc/loadavg • Maximum value: smaller value between /proc/sys/kernel/pid_max and /proc/sys/kernel/threads-max.

Table 7-29 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"> • /mnt/paas/kubernetes/kubelet/ • /var/lib/docker/ • /var/lib/containerd/ • /var/paas/sys/log/cceaddon-npd/ <p>The temporary file npd-disk-write-ping is generated in the detection path.</p> <p>Currently, additional data disks are not supported.</p>

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"> • Default threshold: 90% • Source: df -h <p>Currently, additional data disks are not supported.</p>
emptyDir storage pool error EmptyDirVolumeGroupStatusError	<p>Check whether the ephemeral volume group on the node is normal.</p> <p>Impact: The pod that depends 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 a temporary volume storage pool. The user deletes some data disks by mistake. As a result, the storage pool becomes abnormal.</p>	<ul style="list-style-type: none"> • Detection period: 30s • Source: vgs -o vg_name, vg_attr • Principle: Check whether the VG (storage pool) is in the P state. If yes, some PVs (data disks) are lost. • 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. • 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 nodestatus.allocatable to 0. 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.
PV storage pool error LocalPvVolumeGroupStatusError	<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>	<p>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 nodestatus.allocatable to 0. 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.</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 cd 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 && echo "ok"</pre>
<p>Suspended disk I/O</p> <p>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"> • Check object: all data disks • Source: /proc/diskstat <p>Alternatively, you can run the following command:</p> <pre>iostat -xmt 1</pre> <ul style="list-style-type: none"> • Threshold: <ul style="list-style-type: none"> - Average usage: ioutil >= 0.99 - Average I/O queue length: avgqu-sz >= 1 - Average I/O transfer volume: iops (w/s) + ioth (wMB/s) <= 1 <p>NOTE</p> <p>In some OSs, no data changes during I/O. In this case, calculate the CPU I/O time usage. The value of iowait 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"> • Check object: all data disks • Source: /proc/diskstat Alternatively, you can run the following command: iostat -xmt 1 • Default threshold: Average I/O latency: await >= 5000 ms <p>NOTE If I/O requests are not responded and the await data is not updated, this check item is invalid.</p>

Table 7-30 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"> • /proc/{PID}/stat • Alternately, you can run the ps aux command. 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: /etc/resolv.conf
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"> • http://169.254.169.254/metadata/latest/events/scheduled 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 npd.

Table 7-31 Default kubelet check items

Check Item	Function	Description
Insufficient PID resources PIDPressure	Check whether PIDs are sufficient.	<ul style="list-style-type: none"> • Interval: 10 seconds • Threshold: 90% • Defect: In community version 1.23.1 and earlier versions, this check item becomes invalid when over 65535 PIDs are used. For details, see issue 107107. In community version 1.24 and earlier versions, thread-max is not considered in this check item.

Check Item	Function	Description
Insufficient memory MemoryPressure	Check whether the allocable memory for the containers is sufficient.	<ul style="list-style-type: none"> ● Interval: 10 seconds ● Threshold: max. 100 MiB ● Allocable = Total memory of a node – Reserved memory of a node ● Defect: This check item checks only the memory consumed by containers, and does not consider that consumed by other elements on the node.
Insufficient disk resources DiskPressure	Check the disk usage and inodes usage of the kubelet and Docker disks.	<ul style="list-style-type: none"> ● Interval: 10 seconds ● Threshold: 90%

8 Node Pools

8.1 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

Generally, all nodes in a node pool have the following same attributes:

- Node OS
- Node specifications
- Node login mode
- Node container runtime
- Startup parameters of Kubernetes components on a node
- User-defined 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 user-created 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.

Application 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 8-1 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	When a node pool is deleted, the nodes in the node pool are deleted first. Workloads on the original nodes are 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.	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.	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.
Changing node pool configurations	You can modify the node pool name, node quantity, Kubernetes labels (and their quantity), and taints and adjust the disk, OS, and container engine configurations 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.
Cloning 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"> • This function is supported only in clusters of v1.15 and later. It is not displayed for versions earlier than v1.15. • The default node pool DefaultPool does not support this type of configuration.

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 [Scheduling Policy \(Affinity/Anti-affinity\)](#).

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](#)
- [Scheduling Policy \(Affinity/Anti-affinity\)](#)

8.2 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](#).

Constraints

- The autoscaler add-on needs to be installed for node auto scaling. For details about the add-on installation and parameter configuration, see [CCE Cluster Autoscaler](#).


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 and click the **Node Pools** tab on the right.
- Step 3** In the upper right corner of the page, click **Create Node Pool**.

Basic Settings

Table 8-2 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.
Nodes	Number of nodes to be created in this node pool.

Parameter	Description
Auto Scaling	<p>By default, auto scaling is disabled.</p> <p>To enable auto scaling, install the autoscaler add-on.</p> <p>After you enable auto scaling by switching on , nodes in the node pool will be automatically created or deleted based on cluster loads.</p> <ul style="list-style-type: none"> • 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, these node pools are not prioritized and they will be scaled out by following the rule of maximizing resource utilization. <p>NOTE CCE selects a node pool for auto scaling based on the following policies:</p> <ol style="list-style-type: none"> 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. <ul style="list-style-type: none"> • Cooldown Period: Enter a period, in minutes. This field indicates the period during which the nodes added in the current node pool cannot be scaled in. Scale-in cooling intervals can be configured in the node pool settings and the autoscaler add-on settings. <p>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 setting takes effect in the entire node pool.</p> <p>Scale-in cooling interval configured in the autoscaler add-on</p>

Parameter	Description
	<p>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 setting takes effect in the entire cluster.</p> <p>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 setting takes effect in the entire cluster.</p> <p>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 setting takes effect in the entire cluster.</p> <p>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.</p>

Compute Settings

You can configure the specifications and OS of a cloud server, on which your containerized applications run.

Table 8-3 Configuration parameters

Parameter	Description
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>You are advised to select Random 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>CCE cluster:</p> <ul style="list-style-type: none"> ECS (VM): Containers run on ECSs. ECS (physical): Containers run on servers using the QingTian architecture. <p>CCE Turbo cluster:</p> <ul style="list-style-type: none"> ECS (VM): Containers run on ECSs. Only the ECSs that can be bound with multiple NICs are supported. ECS (physical): Containers run on servers using the QingTian architecture.

Parameter	Description
Container Engine	<p>CCE clusters support Docker and containerd in some scenarios.</p> <ul style="list-style-type: none"> VPC network clusters of v1.23 and later versions support containerd. Tunnel network clusters of v1.23.2-r0 and later versions support containerd. For a CCE Turbo cluster, both Docker and containerd are supported.
Specifications	<p>Select a node specification based on service requirements. The available node specifications vary depending on regions or AZs. For details, see the CCE console.</p>
OS	<p>Select an OS type. Different types of nodes support different OSs. For details, see Supported Node Specifications.</p> <p>Public image: Select an OS for the node.</p>
Login Mode	<ul style="list-style-type: none"> Key Pair Select the key pair used to log in to the node. You can select a shared key. <p>A key pair is used for identity authentication when you remotely log in to a node. If no key pair is available, click Create Key Pair.</p>

Storage Settings

Configure storage resources on a node for the containers running on it. Set the disk size according to site requirements.

Table 8-4 Configuration parameters

Parameter	Description
System Disk	<p>System disk used by the node OS. The value ranges from 40 GiB to 1,024 GiB. The default value is 50 GiB.</p> <p>Encryption: System disk encryption safeguards your data. Snapshots generated from encrypted disks and disks created using these snapshots automatically inherit the encryption setting. This function is available only in certain regions.</p> <ul style="list-style-type: none"> Encryption is not selected by default. After selecting Encryption, you can select an existing key in the displayed dialog box. If no key is available, click View Key List and create a key. After the key is created, click the refresh icon next to the Encryption text box.

Parameter	Description
Data Disk	<p>At least one data disk is required for the container runtime and kubelet. The data disk cannot be deleted or uninstalled. Otherwise, the node will be unavailable.</p> <ul style="list-style-type: none"> • First data disk: used for container runtime and kubelet components. The value ranges from 20 GiB to 32,768 GiB. The default value is 100 GiB. • Other data disks: You can set the data disk size to a value ranging from 10 GB to 32,768 GiB. The default value is 100 GiB. <p>NOTE If the node flavor is disk-intensive or ultra-high I/O, one data disk can be a local disk. 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.</p> <p>Advanced Settings Click Expand to configure the following parameters:</p> <ul style="list-style-type: none"> • Data Disk Space Allocation: After selecting Set Container Engine Space, you can specify the proportion of the space for the container engine, image, and temporary storage on the data disk. The container engine space is used to store the working directory, container image data, and image metadata for the container runtime. The remaining space of the data disk is used for pod configuration files, keys, and EmptyDir. For details about how to allocate data disk space, see Data Disk Space Allocation. • Encryption: Data disk encryption safeguards your data. Snapshots generated from encrypted disks and disks created using these snapshots automatically inherit the encryption setting. This function is available only in certain regions. <ul style="list-style-type: none"> - Encryption is not selected by default. - After selecting Encryption, you can select an existing key in the displayed dialog box. If no key is available, click View Key List and create a key. After the key is created, click the refresh icon next to the Encryption text box. <p>Adding Multiple Data Disks A maximum of four data disks can be added. By default, raw disks are created without any processing. You can also click Expand and select any of the following options:</p> <ul style="list-style-type: none"> • Default: By default, a raw disk is created without any processing. • Mount Disk: The data disk is attached to a specified directory. • Use as PV: applicable to scenarios in which there is a high performance requirement on PVs. The node.kubernetes.io/local-storage-persistent label is added to the node with PV configured. The value is linear or striped.

Parameter	Description
	<ul style="list-style-type: none"> • Use as ephemeral volume: applicable to scenarios in which there is a high performance requirement on EmptyDir. <p>NOTE</p> <ul style="list-style-type: none"> • 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. • 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. <p>Local Persistent Volumes (Local PVs) and Local EVs support the following write modes:</p> <ul style="list-style-type: none"> • 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, allowing data to be concurrently read and written. A storage pool consisting of striped volumes cannot be scaled-out. This option can be selected only when multiple volumes exist.

Network Settings

Configure networking resources to allow node and containerized application access.

Table 8-5 Configuration parameters

Parameter	Description
Node Subnet	The node subnet selected during cluster creation is used by default. You can choose another subnet instead.
Node IP	Random allocation is supported.
Associate Security Group	<p>Security group used by the nodes created in the node pool. A maximum of 5 security groups can be selected.</p> <p>When a cluster is created, a node security group named {Cluster name}-cce-node-{Random ID} 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>

Advanced Settings

Configure advanced node capabilities such as labels, taints, and startup command.

Table 8-6 Advanced configuration parameters

Parameter	Description
Kubernetes Label	<p>A key-value pair added to a Kubernetes object (such as a pod). 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 Labels and Selectors.</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"> • 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. <p>For details, see Managing Node Taints.</p> <p>NOTE 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 Maximum Number of Pods That Can Be Created on a Node.</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 Add ECS Group to create one. After the ECS group is created, click the refresh button.</p>

Parameter	Description
Pre-installation Command	Enter commands. A maximum of 1,000 characters are allowed. 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 Command	Enter commands. A maximum of 1,000 characters are allowed. The script will be executed after Kubernetes software is installed and will not affect the installation. NOTE Do not run the reboot command in the post-installation script to restart the system immediately. To restart the system, run the shutdown -r 1 command to delay the restart for one minute.
Agency	An agency is created by the account 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. If no agency is available, click Create Agency on the right to create one.

Step 4 Click **Next: Confirm**.

Step 5 Click **Submit**.

----End

8.3 Managing a Node Pool

8.3.1 Updating a Node Pool

Constraints

- When editing the resource tags of the node pool. The modified configuration takes effect only for new nodes. To synchronize the configuration to the existing nodes, manually reset the existing nodes.
- Updates of kubernetes labels and taints are automatically synchronized to existing nodes. You do not need to reset 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 and click the **Node Pools** tab on the right.

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 8-7 Basic settings

Parameter	Description
Node Pool Name	Name of the node pool.
Nodes	Modify the number of nodes based on service requirements.
Auto Scaling	<p>This function is disabled by default.</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"> • 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. • Node Pool Priority: indicates the priority of a node pool for a scale-out. A larger value indicates a higher priority. For example, the node pool with priority 4 is scaled out prior to the one with priority 1. If the priorities of multiple node pools are set to the same value, these node pools are not prioritized and they will be scaled out by following the rule of maximizing resource utilization. After the priority is changed, the modification takes effect within 1 minute. • Cooldown Period: Enter a period, in minutes. It specifies a period during which the nodes added in the current node pool cannot be scaled in. <p>To ensure the proper running of AS, install the autoscaler.</p>

Advanced Settings

Table 8-8 Advanced settings

Parameter	Description
Kubernetes Label	<p>A Kubernetes label is a key-value pair added to a Kubernetes object (such as a pod). After specifying a label, click Add. 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 Labels and Selectors.</p> <p>NOTE After a Kubernetes label is modified, the inventory nodes in the node pool are updated synchronously.</p>

Parameter	Description
Taint	<p>This field is left blank by default. You can add taints to configure node anti-affinity. A maximum of 20 taints are allowed for each node. Each taint contains the following parameters:</p> <ul style="list-style-type: none"> • 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. <p>For details, see Managing Node Taints.</p> <p>NOTE After a taint is modified, the existing nodes in the node pool are updated synchronously.</p>
Edit key pair	<p>Only node pools that use key pairs for login support key pair editing. You can select another key pair.</p> <p>NOTE The edited key pair automatically takes effect on newly added nodes. For existing nodes, manually reset the nodes for the modification to take effect.</p>

Step 4 After the configuration, click **OK**.

After the node pool parameters are updated, go to the **Nodes** page to check whether the node to which the node pool belongs is updated. You can reset the node to synchronize the configuration updates for the node pool.

----End

8.3.2 Configuring a Node Pool

Constraints

The default node pool DefaultPool 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 and later**. It is not displayed for clusters 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 and click the **Node Pools** tab on the right.
- Step 3** Choose **More > Manage** in the **Operation** column of the target node pool
- Step 4** On the **Manage Components** page on the right, change the values of the following Kubernetes parameters:

Table 8-9 kubelet

Parameter	Description	Default Value	Modification	Remarks
cpu-manager-policy	<p>CPU management policy configuration. For details, see CPU Scheduling.</p> <ul style="list-style-type: none"> • none: disables pods from exclusively occupying CPUs. Select this value if you want a large pool of shareable CPU cores. • static: enables pods to exclusively occupy CPUs. Select this value if your workload is sensitive to latency in CPU cache and scheduling. 	none	None	None
kube-api-qps	Query per second (QPS) for communicating with kube-apiserver.	100	None	None
kube-api-burst	Burst to use while talking with kube-apiserver.	100	None	None

Parameter	Description	Default Value	Modification	Remarks
max-pods	Maximum number of pods managed by kubelet.	<ul style="list-style-type: none"> For a CCE cluster, the maximum number of pods is determined based on the maximum number of pods on a node. For a CCE Turbo cluster, the maximum number of pods is determined based on the number of NICs on a CCE Turbo cluster node. 	None	None
pod-pids-limit	Limited number of PIDs in Kubernetes	-1	None	None
with-local-dns	Whether to use the local IP address as the ClusterDNS of the node.	false	None	None

Parameter	Description	Default Value	Modification	Remarks
event-qps	QPS limit for event creation	5	None	None
allowed-unsafe-sysctls	Insecure system configuration allowed. Starting from v1.17.17 , CCE enables pod security policies for kube-apiserver. Add corresponding configurations to allowedUnsafeSysctls 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 Example of Enabling Unsafe Sysctls in Pod Security Policy .	[]	None	None
over-subscription-resource	Whether to enable node oversubscription. If this parameter is set to true , node oversubscription is enabled.	false	None	None
colocation	Whether to enable hybrid deployment on nodes. If this parameter is set to true , hybrid deployment is enabled on nodes.	false	None	None

Parameter	Description	Default Value	Modification	Remarks
kube-reserved-mem system-reserved-mem	Reserved node memory.	Depends on node specifications. For details, see Node Resource Reservation Policy .	None	The sum of kube-reserved-mem and system-reserved-mem is less than half of the memory.
topology-manager-policy	<p>Set the topology management policy.</p> <p>Valid values are as follows:</p> <ul style="list-style-type: none"> • restricted: kubelet accepts only pods that achieve optimal NUMA alignment on the requested resources. • best-effort: kubelet preferentially selects pods that implement NUMA alignment on CPU and device resources. • none (default): The topology management policy is disabled. • single-numa-node: kubelet allows only pods that are aligned to the same NUMA node in terms of CPU and device resources. 	none	None	<p>NOTICE</p> <p>Modifying topology-manager-policy and topology-manager-scope 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>

Parameter	Description	Default Value	Modification	Remarks
topology-manager-scope	Set the resource alignment granularity of the topology management policy. Valid values are as follows: <ul style="list-style-type: none"> • container (default) • pod 	container		
resolv-conf	DNS resolution configuration file specified by the container	The default value is null.	None	None
runtime-request-timeout	Timeout interval of all runtime requests except long-running requests (pull, logs, exec, and attach).	The default value is 2m0s .	None	None
registry-pull-qps	Maximum number of image pulls per second.	The default value is 5 .	The value ranges from 1 to 50.	None
registry-burst	Maximum number of burst image pulls.	The default value is 10 .	The value ranges from 1 to 100 and must be greater than or equal to the value of registry-pull-qps .	None
serialize-image-pulls	When this function is enabled, kubelet is notified to pull only one image at a time.	The default value is true .	None	None

Parameter	Description	Default Value	Modification	Remarks
evictionHard : memory.available	A hard eviction signal. The threshold is memory.available .	The value is fixed at 100 MiB.	None	For details, see Node-pressure Eviction .
evictionHard : nodefs.available	A hard eviction signal. The threshold is nodefs.available .	10%	The value ranges from 1% to 99%.	NOTICE Exercise caution when modifying the eviction threshold configuration. Improper configuration may cause pods to be frequently evicted or fail to be evicted when the node is overloaded. nodefs and imagefs correspond to the file system partitions used by kubelet and container engines, respectively.
evictionHard : nodefs.inodesFree	A hard eviction signal. The threshold is nodefs.inodesFree .	5%	The value ranges from 1% to 99%.	
evictionHard : imagefs.available	A hard eviction signal. The threshold is imagefs.available .	10%	The value ranges from 1% to 99%.	
evictionHard : imagefs.inodesFree	A hard eviction signal. The threshold is imagefs.inodesFree .	This parameter is left blank by default.	The value ranges from 1% to 99%.	
evictionHard : pid.available	A hard eviction signal. The threshold is pid.available .	10%	The value ranges from 1% to 99%.	

Parameter	Description	Default Value	Modification	Remarks
evictionSoft:memory.available	A soft eviction signal. The threshold is memory.available .	This parameter is left blank by default.	The value ranges from 100 MiB to 1,000,000 MiB. Configure evictionSoftGracePeriod of the corresponding eviction signal to configure the eviction grace period. This value must be greater than the threshold of the corresponding hard eviction signal.	
evictionSoft:nodfs.available	A soft eviction signal. The threshold is nodfs.available .	This parameter is left blank by default.	The value ranges from 1% to 99%. Configure evictionSoftGracePeriod of the corresponding eviction signal to configure the eviction grace period. This value must be greater than the threshold of the corresponding hard eviction signal.	

Parameter	Description	Default Value	Modification	Remarks
evictionSoft:nodefs.inodesFree	A soft eviction signal. The threshold is nodefs.inodesFree .	This parameter is left blank by default.	The value ranges from 1% to 99%. Configure evictionSoftGracePeriod of the corresponding eviction signal to configure the eviction grace period. This value must be greater than the threshold of the corresponding hard eviction signal.	
evictionSoft:imagefs.available	A soft eviction signal. The threshold is imagefs.available .	This parameter is left blank by default.	The value ranges from 1% to 99%. Configure evictionSoftGracePeriod of the corresponding eviction signal to configure the eviction grace period. This value must be greater than the threshold of the corresponding hard eviction signal.	

Parameter	Description	Default Value	Modification	Remarks
evictionSoft: imagefs.inodesFree	A soft eviction signal. The threshold is imagefs.inodesFree .	This parameter is left blank by default.	The value ranges from 1% to 99%. Configure evictionSoftGracePeriod of the corresponding eviction signal to configure the eviction grace period. This value must be greater than the threshold of the corresponding hard eviction signal.	
evictionSoft: pid.available	A soft eviction signal. The threshold is pid.available .	This parameter is left blank by default.	The value ranges from 1% to 99%. Configure evictionSoftGracePeriod of the corresponding eviction signal to configure the eviction grace period. This value must be greater than the threshold of the corresponding hard eviction signal.	

Table 8-10 kube-proxy

Parameter	Description	Default Value	Modification
contrack-min	Maximum number of connection tracking entries To obtain the value, run the following command: sysctl -w net.nf_contrack_max	131072	None
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	1h0m0s	None

Table 8-11 Network components (available only for CCE Turbo clusters)

Parameter	Description	Default Value	Modification
nic-minimum-target	Minimum number of ENIs bound to the nodes in the node pool	Default: 10	None
nic-maximum-target	Maximum number of ENIs pre-bound to a node at the node pool level	Default: 0	None
nic-warm-target	Number of ENIs pre-bound to a node at the node pool level	Default: 2	None
nic-max-above-warm-target	Reclaim number of ENIs pre-bound to a node at the node pool level	Default: 2	None

Table 8-12 Pod security group in a node pool (available only for CCE Turbo clusters)

Parameter	Description	Default Value	Modification
security_group_ids_for_nodepool	<ul style="list-style-type: none"> Default security group used by pods in a node pool. You can enter the security group ID. If this parameter is not set, the default security group of the cluster container network is used. A maximum of five security group IDs can be specified at the same time, separated by semicolons (;). The priority of the security group is lower than that of the security group configured for Security Groups. 	None	None

Table 8-13 Docker (available only for node pools that use Docker)

Parameter	Description	Default Value	Modification
native-umask	`--exec-opt native.umask	normal	Cannot be changed.
docker-base-size	`--storage-opt dm.basesize	0	Cannot be changed.
insecure-registry	Address of an insecure image registry	false	Cannot be changed.
limitcore	Maximum size of a core file in a container. The unit is byte. If not specified, the value is infinity .	5368709120	None

Parameter	Description	Default Value	Modification
default-ulimit-nofile	Limit on the number of handles in a container	{soft}:{hard}	The value cannot exceed the value of the kernel parameter nr_open and cannot be a negative number. You can run the following command to obtain the kernel parameter nr_open : <code>sysctl -a grep nr_open</code>
image-pull-progress-timeout	If the image fails to be pulled before time outs, the image pull will be canceled.	The default value is 1m0s .	This parameter is supported in v1.25.3-r0 and later.

Table 8-14 containerd (available only for node pools that use containerd)

Parameter	Description	Default Value	Modification
devmapper-base-size	Available data space of a single container	None	Cannot be changed.
limitcore	Maximum size of a core file in a container. The unit is byte. If not specified, the value is infinity .	5368709120	None
default-ulimit-nofile	Limit on the number of handles in a container	1048576	The value cannot exceed the value of the kernel parameter nr_open and cannot be a negative number. You can run the following command to obtain the kernel parameter nr_open : <code>sysctl -a grep nr_open</code>
image-pull-progress-timeout	If the image fails to be pulled before time outs, the image pull will be canceled.	The default value is 1m0s .	This parameter is supported in v1.25.3-r0 and later.

Step 5 Click **OK**.

----End

8.3.3 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.

Step 2 Click the cluster name to access the cluster console. Choose **Nodes** in the navigation pane and click the **Node Pools** tab on the right.

Step 3 Choose **More > Copy** in the **Operation** column of the target node pool.

Step 4 The configurations of the selected node pool are replicated to the **Clone Node Pool** page. You can edit the configurations as required. For details about configuration items, see [Creating a Node Pool](#). After confirming the configuration, click **Next: Confirm**.

Step 5 On the **Confirm** page, confirm the node pool configuration and click **Submit**. Then, a new node pool is created based on the edited configuration.

----End

8.3.4 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:
 - When editing the resource tags of the node pool. The modified configuration takes effect only for new nodes. To synchronize the configuration to the existing nodes, manually reset the existing nodes.
 - Updates of kubernetes labels and taints are automatically synchronized to existing nodes. You do not need to reset 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 and click the **Nodes** tab on the right.
 - 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 and click the **Node Pools** tab on the right.
 - 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

8.3.5 Upgrading an OS

When CCE releases a new OS image, existing nodes cannot be automatically upgraded. You can manually upgrade them in batches.

NOTICE

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.

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 and click the **Node Pools** tab on the right.

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 upgrade 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:
 - **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 Key Pair**.
- **Pre-installation Command:** 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.
- **Post-installation Command:** Enter a maximum of 1,000 characters.
The script will be executed after Kubernetes software is installed and will 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 and click the **Node Pools** tab on the right.

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

8.3.6 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 and click the cluster name to access the cluster console.

Step 2 In the navigation pane, choose **Nodes** and switch to the **Node Pools** tab page.

Step 3 Click **View Node** in the **Operation** column of the node pool to be migrated.

Step 4 Click **More > Migrate** in the **Operation** column of the target node to migrate the node.

Step 5 In the displayed **Migrate Node** window, confirm the information.

NOTE

The migration has no impacts on the original resource tags, Kubernetes labels, and taints of the node.

----End

8.3.7 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.

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 and click the **Node Pools** tab on the right.
- 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, click **Yes** to confirm that you want to continue the deletion.
----End

9 Workloads

9.1 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.

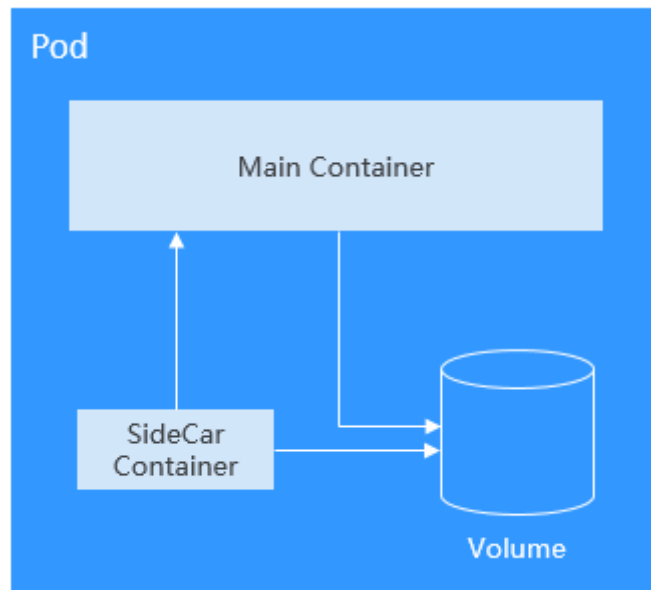
Pod

A pod is the smallest and simplest unit in the Kubernetes object model that you create or deploy. A pod encapsulates one or more containers, storage volumes, a unique network IP address, and options that govern how the containers should run.

Pods can be used in either of the following ways:

- A container is running in a pod. This is the most common usage of pods in Kubernetes. You can view the pod as a single encapsulated container, but Kubernetes directly manages pods instead of containers.
- Multiple containers that need to be coupled and share resources run in a pod. In this scenario, an application contains a main container and several sidecar containers, as shown in [Figure 9-1](#). For example, the main container is a web server that provides file services from a fixed directory, and a sidecar container periodically downloads files to the directory.

Figure 9-1 Pod

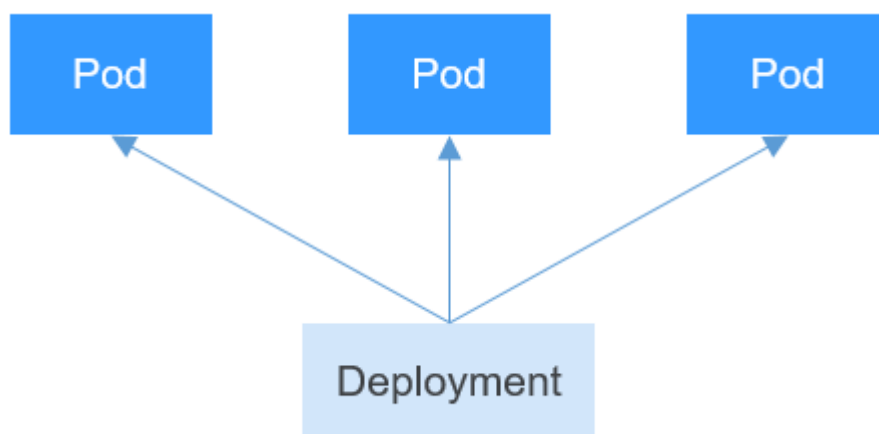


In Kubernetes, pods are rarely created directly. Instead, controllers such as Deployments and Jobs, are used to manage pods. Controllers can create and manage multiple pods, and provide replica management, rolling upgrade, and self-healing capabilities. A controller typically uses a pod template to create corresponding pods.

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 9-2 Deployment



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.

StatefulSet

All pods under a Deployment have the same characteristics except for the name and IP address. If required, a Deployment can use the pod template to create a new pod. If not required, the Deployment can delete any one of the pods.

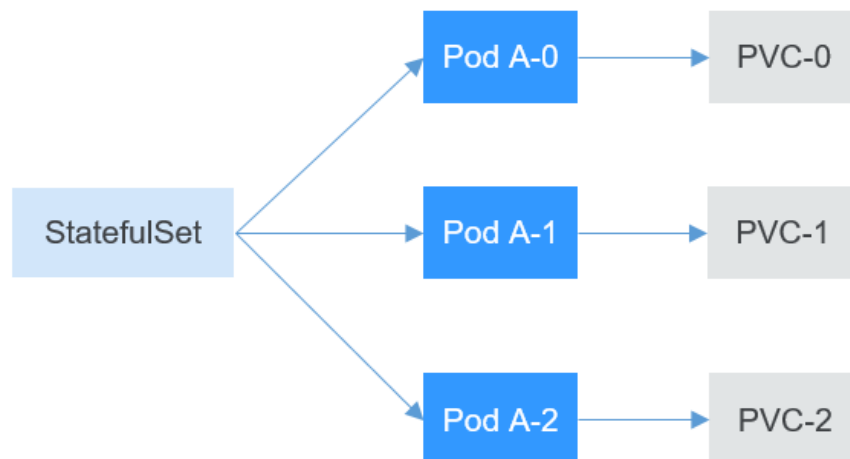
However, Deployments cannot meet the requirements in some distributed scenarios when each pod requires its own status or in a distributed database where each pod requires independent storage.

With detailed analysis, it is found that each part of distributed stateful applications plays a different role. For example, the database nodes are deployed in active/standby mode, and pods are dependent on each other. In this case, the pods need to meet the following requirements:

- A pod can be recognized by other pods. Therefore, a pod must have a fixed identifier.
- Each pod has an independent storage device. After a pod is deleted and then restored, the data read from the pod must be the same as the previous one. Otherwise, the pod status is inconsistent.

To address the preceding requirements, Kubernetes provides StatefulSets.

1. A StatefulSet provides 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 host name remain unchanged.
2. A StatefulSet provides a fixed access domain name for each pod through the headless Service (described in the following sections).
3. The StatefulSet creates PersistentVolumeClaims (PVCs) with fixed identifiers to ensure that pods can access the same persistent data after being rescheduled.

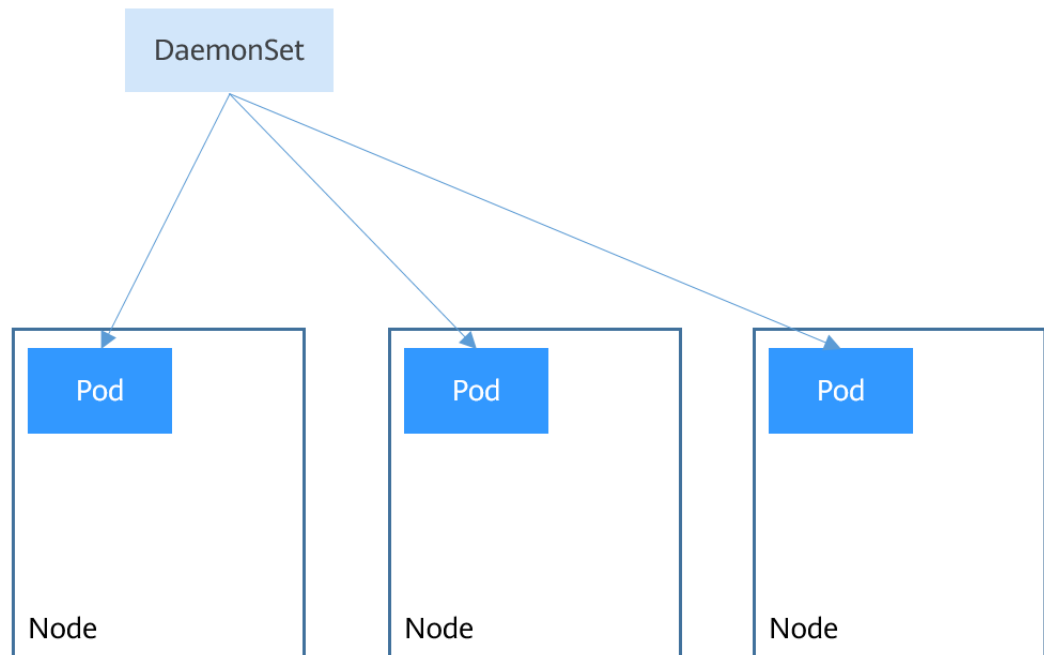


DaemonSet

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 9-3 DaemonSet



Job and Cron Job

Jobs and cron jobs 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 completing tasks based on user configurations.
- A cron job runs a job periodically on a specified schedule. A cron job 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 9-1 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.

9.2 Creating a Workload

9.2.1 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 [Creating a 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 **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 Always , the image is pulled from the image repository each time. If you do not select Always , 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 Select Image and select the image used by the container. To use a third-party image, see Using Third-Party Images .
Image Tag	Select the image tag to be deployed.

Parameter	Description
CPU Quota	<ul style="list-style-type: none"> ▪ Request: minimum number of CPU cores required by a container. The default value is 0.25 cores. ▪ Limit: maximum number of CPU cores available for a container. Do not leave Limit unspecified. Otherwise, intensive use of container resources will occur and your workload may exhibit unexpected behavior. <p>If Request and Limit are not specified, the quota is not limited. For more information and suggestions about Request and Limit, see Setting Container Specifications.</p>
Memory Quota	<ul style="list-style-type: none"> ▪ Request: minimum amount of memory required by a container. The default value is 512 MiB. ▪ Limit: maximum amount of memory available for a container. When memory usage exceeds the specified memory limit, the container will be terminated. <p>If Request and Limit are not specified, the quota is not limited. For more information and suggestions about Request and Limit, see Setting Container Specifications.</p>
(Optional) GPU Quota	<p>Configurable only when the cluster contains GPU nodes and the gpu-device-plugin add-on is installed.</p> <ul style="list-style-type: none"> ▪ All: The GPU is not used. ▪ Dedicated: GPU resources are exclusively used by the container. ▪ Shared: percentage of GPU resources used by the container. For example, if this parameter is set to 10%, the container uses 10% of GPU resources. <p>For details about how to use GPU in the cluster, see Default GPU Scheduling in Kubernetes.</p>
(Optional) Privileged Container	<p>Programs in a privileged container have certain privileges.</p> <p>If Privileged Container 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	<p>Indicates whether to use the container as an init container. The 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 Init Container.</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 [Setting Container Lifecycle Parameters](#).
- (Optional) **Health Check**: Set the liveness probe, ready probe, and startup probe as required. For details, see [Setting Health Check for a Container](#).
- (Optional) **Environment Variables**: Set 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 [Setting an Environment Variable](#).
- (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**: Set container permissions to protect the system and other containers from being affected. Enter the user ID to set container permissions and prevent systems and other containers from being affected.
- (Optional) **Logging**: Report container stdout streams to AOM by default and require no manual settings. You can manually configure the log collection path. For details, see [Using ICAgent to Collect Container Logs](#).

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 9-16](#).

- **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 with the specified GPU graphics card type.

(Optional) Service Settings

A Service provides external access for pods. With a static IP address, a Service forwards access traffic to pods and performs automatic load balancing 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 upgrade parameters of the workload. **Rolling upgrade** and **Replace upgrade** are supported. For details, see [Configuring the Workload Upgrade Policy](#).
- **Scheduling:** Configure affinity and anti-affinity policies for flexible workload scheduling. Node affinity, pod affinity, and pod anti-affinity are supported. For details, see [Scheduling Policy \(Affinity/Anti-affinity\)](#).
- **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 [Taints and Tolerations](#).
- **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 [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 Rate Limiting for Inter-Pod Access](#).

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 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 in My Images, obtain the image path from SWR.
        imagePullPolicy: Always
        name: nginx
      imagePullSecrets:
        - name: default-secret

```

For details about these parameters, see [Table 9-2](#).

Table 9-2 Deployment YAML parameters

Parameter	Description	Mandatory/Optional
apiVersion	API version. NOTE Set this parameter based on the cluster version. <ul style="list-style-type: none"> For clusters of v1.17 or later, the apiVersion format of Deployments is apps/v1. For clusters of v1.15 or earlier, the apiVersion format of Deployments is extensions/v1beta1. 	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"> RollingUpdate ReplaceUpdate By default, rolling update is used.	Optional
template	Detailed description of a created container pod.	Mandatory
metadata	Metadata.	Mandatory
labels	metadata.labels: Container labels.	Optional

Parameter	Description	Mandatory/Optional
spec: containers	<ul style="list-style-type: none"> • image (mandatory): Name of a container image. • imagePullPolicy (optional): Policy for obtaining an image. The options include Always (attempting to download images each time), Never (only using local images), and IfNotPresent (using local images if they are available; downloading images if local images are unavailable). The default value is Always. • name (mandatory): Container name. 	Mandatory
imagePullSecrets	<p>Name of the secret used during image pulling. If a private image is used, this parameter is mandatory.</p> <ul style="list-style-type: none"> • To pull an image from the Software Repository for Container (SWR), set this parameter to default-secret. • To pull an image from a third-party image repository, set this parameter to the name of the created secret. 	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:** 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, add the corresponding Service. For details, see [Network](#).

----End

9.2.2 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.

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 Service](#).
- 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 [Creating a 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 **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 Always , the image is pulled from the image repository each time. If you do not select Always , 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 Select Image and select the image used by the container. To use a third-party image, see Using Third-Party Images .
Image Tag	Select the image tag to be deployed.
CPU Quota	<ul style="list-style-type: none"> ▪ Request: minimum number of CPU cores required by a container. The default value is 0.25 cores. ▪ Limit: maximum number of CPU cores available for a container. Do not leave Limit unspecified. Otherwise, intensive use of container resources will occur and your workload may exhibit unexpected behavior. <p>If Request and Limit are not specified, the quota is not limited. For more information and suggestions about Request and Limit, see Setting Container Specifications.</p>

Parameter	Description
Memory Quota	<ul style="list-style-type: none"> ▪ Request: minimum amount of memory required by a container. The default value is 512 MiB. ▪ Limit: maximum amount of memory available for a container. When memory usage exceeds the specified memory limit, the container will be terminated. <p>If Request and Limit are not specified, the quota is not limited. For more information and suggestions about Request and Limit, see Setting Container Specifications.</p>
(Optional) GPU Quota	<p>Configurable only when the cluster contains GPU nodes and the gpu-device-plugin add-on is installed.</p> <ul style="list-style-type: none"> ▪ All: The GPU is not used. ▪ Dedicated: GPU resources are exclusively used by the container. ▪ Shared: percentage of GPU resources used by the container. For example, if this parameter is set to 10%, the container uses 10% of GPU resources. <p>For details about how to use GPU in the cluster, see Default GPU Scheduling in Kubernetes.</p>
(Optional) Privileged Container	<p>Programs in a privileged container have certain privileges.</p> <p>If Privileged Container 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>Indicates whether to use the container as an init container. The 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 Init Container.</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 [Setting Container Lifecycle Parameters](#).
- (Optional) **Health Check:** Set the liveness probe, ready probe, and startup probe as required. For details, see [Setting Health Check for a Container](#).

- (Optional) **Environment Variables:** Set 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 [Setting an Environment Variable](#).
- (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](#) and [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:** Set container permissions to protect the system and other containers from being affected. Enter the user ID to set container permissions and prevent systems and other containers from being affected.
 - (Optional) **Logging:** Report container stdout streams to AOM by default and require no manual settings. You can manually configure the log collection path. For details, see [Using ICAgent to Collect Container Logs](#).
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 9-16](#).
- **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 with the specified GPU graphics card 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 Service](#).

(Optional) Service Settings

A Service provides external access for pods. With a static IP address, a Service forwards access traffic to pods and performs automatic load balancing 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 upgrade parameters of the workload. **Rolling upgrade** and **Replace upgrade** are supported. For details, see [Configuring the Workload Upgrade Policy](#).
- **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. Node affinity, pod affinity, and pod anti-affinity are supported. For details, see [Scheduling Policy \(Affinity/Anti-affinity\)](#).
- **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 [Taints and Tolerations](#).
- **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 [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 Rate Limiting for Inter-Pod Access](#).

Step 4 Click **Create Workload** in the lower right corner.

----End

Using kubectl

In this example, an nginx workload is used and the EVS volume is dynamically mounted to it using the **volumeClaimTemplates** field.

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-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-0 # 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 # Storage class 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, set the corresponding workload access type. For details, see [Network](#).

----End

9.2.3 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

You must have one cluster available before creating a DaemonSet. For details on how to create a cluster, see [Creating a 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 **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 Always , the image is pulled from the image repository each time. If you do not select Always , 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 Select Image and select the image used by the container. To use a third-party image, see Using Third-Party Images .
Image Tag	Select the image tag to be deployed.

Parameter	Description
CPU Quota	<ul style="list-style-type: none"> ▪ Request: minimum number of CPU cores required by a container. The default value is 0.25 cores. ▪ Limit: maximum number of CPU cores available for a container. Do not leave Limit unspecified. Otherwise, intensive use of container resources will occur and your workload may exhibit unexpected behavior. <p>If Request and Limit are not specified, the quota is not limited. For more information and suggestions about Request and Limit, see Setting Container Specifications.</p>
Memory Quota	<ul style="list-style-type: none"> ▪ Request: minimum amount of memory required by a container. The default value is 512 MiB. ▪ Limit: maximum amount of memory available for a container. When memory usage exceeds the specified memory limit, the container will be terminated. <p>If Request and Limit are not specified, the quota is not limited. For more information and suggestions about Request and Limit, see Setting Container Specifications.</p>
(Optional) GPU Quota	<p>Configurable only when the cluster contains GPU nodes and the gpu-device-plugin add-on is installed.</p> <ul style="list-style-type: none"> ▪ All: The GPU is not used. ▪ Dedicated: GPU resources are exclusively used by the container. ▪ Shared: percentage of GPU resources used by the container. For example, if this parameter is set to 10%, the container uses 10% of GPU resources. <p>For details about how to use GPU in the cluster, see Default GPU Scheduling in Kubernetes.</p>
(Optional) Privileged Container	<p>Programs in a privileged container have certain privileges.</p> <p>If Privileged Container 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	<p>Indicates whether to use the container as an init container. The 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 Init Container.</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 [Setting Container Lifecycle Parameters](#).
- (Optional) **Health Check**: Set the liveness probe, ready probe, and startup probe as required. For details, see [Setting Health Check for a Container](#).
- (Optional) **Environment Variables**: Set 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 [Setting an Environment Variable](#).
- (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**: Set container permissions to protect the system and other containers from being affected. Enter the user ID to set container permissions and prevent systems and other containers from being affected.
- (Optional) **Logging**: Report container stdout streams to AOM by default and require no manual settings. You can manually configure the log collection path. For details, see [Using ICAgent to Collect Container Logs](#).

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 9-16](#).

- **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 with the specified GPU graphics card type.

(Optional) Service Settings

A Service provides external access for pods. With a static IP address, a Service forwards access traffic to pods and performs automatic load balancing 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 upgrade parameters of the workload. **Rolling upgrade** and **Replace upgrade** are supported. For details, see [Configuring the Workload Upgrade Policy](#).
- **Scheduling:** Configure affinity and anti-affinity policies for flexible workload scheduling. Node affinity, pod affinity, and pod anti-affinity are supported. For details, see [Scheduling Policy \(Affinity/Anti-affinity\)](#).
- **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 [Taints and Tolerations](#).
- **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 [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 Rate Limiting for Inter-Pod Access](#).

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 connect to 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 documents](#).

```
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.

DaemonSet only creates pods on nodes with specific labels. In the preceding pod template, **nodeSelector** specifies that a pod is created only on nodes with the **daemon: need** label. 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 Query 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, set the corresponding workload access type. For details, see [Network](#).

----End

9.2.4 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 **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 Always , the image is pulled from the image repository each time. If you do not select Always , 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 Select Image and select the image used by the container. To use a third-party image, see Using Third-Party Images .
Image Tag	Select the image tag to be deployed.

Parameter	Description
CPU Quota	<ul style="list-style-type: none"> ▪ Request: minimum number of CPU cores required by a container. The default value is 0.25 cores. ▪ Limit: maximum number of CPU cores available for a container. Do not leave Limit unspecified. Otherwise, intensive use of container resources will occur and your workload may exhibit unexpected behavior. <p>If Request and Limit are not specified, the quota is not limited. For more information and suggestions about Request and Limit, see Setting Container Specifications.</p>
Memory Quota	<ul style="list-style-type: none"> ▪ Request: minimum amount of memory required by a container. The default value is 512 MiB. ▪ Limit: maximum amount of memory available for a container. When memory usage exceeds the specified memory limit, the container will be terminated. <p>If Request and Limit are not specified, the quota is not limited. For more information and suggestions about Request and Limit, see Setting Container Specifications.</p>
(Optional) GPU Quota	<p>Configurable only when the cluster contains GPU nodes and the gpu-device-plugin add-on is installed.</p> <ul style="list-style-type: none"> ▪ All: The GPU is not used. ▪ Dedicated: GPU resources are exclusively used by the container. ▪ Shared: percentage of GPU resources used by the container. For example, if this parameter is set to 10%, the container uses 10% of GPU resources. <p>For details about how to use GPU in the cluster, see Default GPU Scheduling in Kubernetes.</p>
(Optional) Privileged Container	<p>Programs in a privileged container have certain privileges.</p> <p>If Privileged Container 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	<p>Indicates whether to use the container as an init container. The 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 Init Container.</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 [Setting Container Lifecycle Parameters](#).
- (Optional) **Environment Variables**: Set 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 [Setting an Environment Variable](#).
- (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 container stdout streams to AOM by default and require no manual settings. You can manually configure the log collection path. For details, see [Using ICAgent to Collect Container Logs](#).

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 9-16](#).

- **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 with the specified GPU graphics card 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 [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.
- **Network Configuration:**
 - Pod ingress/egress bandwidth limitation: You can set ingress/egress bandwidth limitation for pods. For details, see [Configuring QoS Rate Limiting for Inter-Pod Access](#).

Step 4 Click **Create Workload** in the lower right corner.

----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 9-3 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 completions count.	Work queue processing pod
Parallel jobs with a fixed completion count	Multiple pods run until reaching the specified completions 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 π till the 2000th 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
      imagePullSecrets:
      - name: default-secret
```

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
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 9-4](#).

Table 9-4 Related operations

Operation	Description
Editing a YAML file	Click More > Edit YAML next to the job name to edit the YAML file corresponding to the current job.
Deleting a job	<ol style="list-style-type: none"> 1. Select the job to be deleted and choose More > Delete in the Operation column. 2. Click Yes. Deleted jobs cannot be restored. Exercise caution when deleting a job.

9.2.5 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 **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.
Pull Policy	Image update or pull policy. If you select Always , the image is pulled from the image repository each time. If you do not select Always , 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 Select Image and select the image used by the container. To use a third-party image, see Using Third-Party Images .
Image Tag	Select the image tag to be deployed.

Parameter	Description
CPU Quota	<ul style="list-style-type: none"> ▪ Request: minimum number of CPU cores required by a container. The default value is 0.25 cores. ▪ Limit: maximum number of CPU cores available for a container. Do not leave Limit unspecified. Otherwise, intensive use of container resources will occur and your workload may exhibit unexpected behavior. <p>If Request and Limit are not specified, the quota is not limited. For more information and suggestions about Request and Limit, see Setting Container Specifications.</p>
Memory Quota	<ul style="list-style-type: none"> ▪ Request: minimum amount of memory required by a container. The default value is 512 MiB. ▪ Limit: maximum amount of memory available for a container. When memory usage exceeds the specified memory limit, the container will be terminated. <p>If Request and Limit are not specified, the quota is not limited. For more information and suggestions about Request and Limit, see Setting Container Specifications.</p>
(Optional) GPU Quota	<p>Configurable only when the cluster contains GPU nodes and the gpu-device-plugin add-on is installed.</p> <ul style="list-style-type: none"> ▪ All: The GPU is not used. ▪ Dedicated: GPU resources are exclusively used by the container. ▪ Shared: percentage of GPU resources used by the container. For example, if this parameter is set to 10%, the container uses 10% of GPU resources. <p>For details about how to use GPU in the cluster, see Default GPU Scheduling in Kubernetes.</p>
(Optional) Privileged Container	<p>Programs in a privileged container have certain privileges.</p> <p>If Privileged Container 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	<p>Indicates whether to use the container as an init container. The 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 Init Container.</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 [Setting Container Lifecycle Parameters](#).
- (Optional) **Environment Variables**: Set 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 [Setting an Environment Variable](#).
- **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 with the specified GPU graphics card type.

Execution Settings

- **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 is executed by hour as an example. As **/2, /3, /4, /6, /8, and /12** can exactly divide 24 hours, an accurate period can be represented. If another period is used, the last period will be reset at the beginning of a new day. For example, if the cron expression is ****/12 * * ***, the execution time is **00:00:00** and **12:00:00** every day. If the cron expression is ****/13 * * ***, the execution time is **00:00:00** and **13:00:00** every day. At 00:00 on the next day, the execution time is updated even if the period does not reach 13 hours.

- **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 [Labels and Annotations](#).
- **Network Configuration:**
 - Pod ingress/egress bandwidth limitation: You can set ingress/egress bandwidth limitation for pods. For details, see [Configuring QoS Rate Limiting for Inter-Pod Access](#).

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 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.

 **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 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	*/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
```

kubectl 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 9-5](#).

Table 9-5 Related operations

Operation	Description
Editing a YAML file	Click More > Edit YAML next to the cron job name to edit the YAML file of the current job.
Stopping a CronJob	<ol style="list-style-type: none"> 1. Select the job to be stopped and click Stop in the Operation column. 2. Click Yes.
Deleting a CronJob	<ol style="list-style-type: none"> 1. Select the CronJob to be deleted and click More > Delete in the Operation column. 2. Click Yes. Deleted jobs cannot be restored. Therefore, exercise caution when deleting a job.

9.3 Configuring a Container

9.3.1 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
```

9.3.2 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
```

You can also set the image pull policy when creating a workload on the CCE console. 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.

9.3.3 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, 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, you can enter a private image path in the format of *domainname/namespace/imagename:tag* for **Image Name** and select the key created in [Step 1](#) for **Image Access Credential**.

Step 3 Set other parameters and click **Create Workload**.

----End

Using kubectl

Step 1 Use kubectl to connect to 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

9.3.4 Setting 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 Description

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 9-6 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 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 9-7 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.

Parameter	Description
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.

9.3.5 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:

- **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 during image start. To run a specific command or rewrite the default image value, you must perform specific settings:

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 9-8 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 9-9 Container startup command

Configuration Item	Procedure
Command	<p>Enter an executable command, for example, /run/server.</p> <p>If there are multiple executable commands, write them in different lines.</p> <p>NOTE In the case of multiple commands, you are advised to run /bin/sh or other shell commands. Other commands are used as parameters.</p>
Args	<p>Enter the argument that controls the container running command, for example, --port=8080.</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 9-10 Post-start processing parameters

Parameter	Description
CLI	<p>Set commands to be executed in the container for post-start processing. The command format is Command Args[1] Args[2]... Command 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. Commands that are executed in the background or asynchronously are not supported.</p> <p>Example command:</p> <pre>exec: command: - /install.sh - install_agent</pre> <p>Enter /install install_agent in the script. This command indicates that install.sh 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"> • Path: (optional) request URL. • Port: (mandatory) request port. • Host: (optional) requested host IP address. The default value is the IP address of the pod.

----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 9-11 Pre-stop processing parameters

Parameter	Description
CLI	<p>Set commands to be executed in the container for pre-stop processing. The command format is Command Args[1] Args[2]... Command 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 /uninstall uninstall_agent in the script. This command indicates that the uninstall.sh 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"> • Path: (optional) request URL. • Port: (mandatory) request port. • Host: (optional) requested host IP address. The default value is the IP address of the pod.

----End

Example YAML

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
```

9.3.6 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 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, 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.

- **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. The reason is that the cluster is not in the terminal environment when executing programs in a container.

Common Parameters

Table 9-12 Common parameter description

Parameter	Description
Period (periodSeconds)	Indicates the probe detection period, in seconds. For example, if this parameter is set to 30 , the detection is performed every 30 seconds.
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 30 , 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 10 , 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 0 , 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 1 , 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 1 , which is also the minimum value. The value of this parameter is fixed to 1 in Liveness Probe and Startup Probe .

Parameter	Description
Failure Threshold (failureThreshold)	<p>Number of retry times when the detection fails.</p> <p>Giving up in case of liveness probe means to restart the container. In case of readiness probe the pod will be marked Unready.</p> <p>The default value is 3. The minimum value is 1.</p>

YAML Example

```

apiVersion: v1
kind: Pod
metadata:
  labels:
    test: liveness
  name: liveness-http
spec:
  containers:
  - name: liveness
    image: nginx:alpine
    args:
    - /server
    livenessProbe:
      httpGet:
        path: /healthz
        port: 80
        httpHeaders:
        - name: Custom-Header
          value: Awesome
      initialDelaySeconds: 3
      periodSeconds: 3
    readinessProbe:
      exec:
        command:
        - cat
        - /tmp/healthy
      initialDelaySeconds: 5
      periodSeconds: 5
    startupProbe:
      httpGet:
        path: /healthz
        port: 80
      failureThreshold: 30
      periodSeconds: 10
  
```

9.3.7 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.

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 key:** Import all keys in a ConfigMap as environment variables.
- **Added from ConfigMap:** Import a key in a ConfigMap as the value of an environment variable.
- **Added from secret:** Import all keys 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.
- **Variable value/reference:** Use the field defined by a pod as the value of the environment variable.
- **Resource Reference:** The value of **Request** or **Limit** defined by the container is used as the value of the environment variable.

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 the **Container Settings** area and click the **Environment Variables** tab.

Step 4 Configure environment variables.

----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: key1
      - name: key2              # Added from secret key
        valueFrom:
          secretKeyRef:
            name: secret-example
            key: key2
      - 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
ey1=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 key is directly imported.
secret_key=secret_value # Added from key. The key value in the original secret is directly imported.
```

9.3.8 Configuring the Workload Upgrade Policy

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.

Upgrade Parameters

Parameter	Description	Constraint
Max. Surge (maxSurge)	Specifies the maximum number of pods that can exist compared with spec.replicas . The default value is 25% . For example, if spec.replicas is set to 4 , 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.
Max. Unavailable Pods (maxUnavailable)	Specifies the maximum number of pods that can be unavailable compared with spec.replicas . The default value is 25% . For example, if spec.replicas is set to 4 , 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.

Parameter	Description	Constraint
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 0 (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 kubectl get rs . 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 .spec.minReadySeconds .	None
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 terminationGracePeriodSeconds , 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-tmqk  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 the earlier version when a fault occurs during the 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**.

9.3.9 Scheduling Policy (Affinity/Anti-affinity)

Kubernetes supports node affinity and pod affinity/anti-affinity. You can configure custom rules to achieve affinity and anti-affinity scheduling. For example, you can deploy frontend pods and backend pods together, deploy the same type of applications on a specific node, or deploy different applications on different nodes.

Kubernetes affinity applies to nodes and pods.

- **nodeAffinity**: similar to pod nodeSelector, and they both schedule pods only to the nodes with specified labels. The difference between nodeAffinity and nodeSelector lies in that nodeAffinity features stronger expression than nodeSelector and allows you to specify preferentially selected soft constraints. The two types of node affinity are as follows:

- `requiredDuringSchedulingIgnoredDuringExecution`: hard constraint that **must be met**. The scheduler can perform scheduling only when the rule is met. This function is similar to `nodeSelector`, but it features stronger syntax expression. For details, see [Node Affinity \(nodeAffinity\)](#).
- `preferredDuringSchedulingIgnoredDuringExecution`: soft constraint that is **met as much as possible**. The scheduler attempts to find the node that meets the rule. If no matching node is found, the scheduler still schedules the pod. For details, see [Node Preference Rule](#).
- **Workload Affinity (podAffinity)/Workload Anti-affinity (podAntiAffinity)**: The nodes to which a pod can be scheduled are determined based on the label of the pod running on a node, but not the label of the node. Similar to node affinity, workload affinity and anti-affinity are also of `requiredDuringSchedulingIgnoredDuringExecution` and `preferredDuringSchedulingIgnoredDuringExecution` types.

 **NOTE**

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.

You can create the preceding affinity policies on the console. For details, see [Configuring Scheduling Policies](#).


Configuring Scheduling Policies

Step 1 Log in to the CCE console.

Step 2 When creating a workload, click **Scheduling** in the **Advanced Settings** area.

Table 9-13 Node affinity settings

Parameter	Description
Required	Hard constraint, which corresponds to <code>requiredDuringSchedulingIgnoredDuringExecution</code> for specifying the conditions that must be met. If multiple rules that must be met are added, scheduling will be performed when only one rule is met.
Preferred	Soft constraint, which corresponds to <code>preferredDuringSchedulingIgnoredDuringExecution</code> for specifying the conditions that must be met as many as possible. If multiple rules that must be met as much as possible are added, scheduling will be performed even if one or none of the rules is met.

Step 3 Click  under **Node Affinity**, **Workload Affinity**, or **Workload Anti-Affinity** to add scheduling policies. In the dialog box that is displayed, directly add policies. Alternatively, you can specify nodes or AZs to be scheduled on the console.

Specifying nodes and AZs 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 9-14 Parameters for configuring the scheduling policy

Parameter	Description
Label	Node label. You can use the default label or customize a label.
Operator	<p>The following relations are supported: In, NotIn, Exists, DoesNotExist, Gt, and Lt.</p> <ul style="list-style-type: none"> • In: The label of the affinity or anti-affinity object is in the label value list (values field). • NotIn: The label of the affinity or anti-affinity object is not in the label value list (values field). • Exists: The affinity or anti-affinity object has a specified label name. • DoesNotExist: The affinity or anti-affinity object does not have the specified label name. • Gt: (available only for node affinity) The label value of the scheduled node is greater than the list value (string comparison). • Lt: (available only for node affinity) The label value of the scheduling node is less than the list value (string comparison).
Label Value	Label value.
Namespace	<p>This parameter is available only in a workload affinity or anti-affinity scheduling policy.</p> <p>Namespace for which the scheduling policy takes effect.</p>
Topology Key	<p>This parameter can be used only in a workload affinity or anti-affinity scheduling policy.</p> <p>Select the scope specified by topologyKey and then select the content defined by the policy.</p>
Weight	This parameter can be set only in a Preferred scheduling policy.

----End

Node Affinity (nodeAffinity)

Workload node affinity rules are implemented using node labels. When a node is created in a CCE cluster, certain labels are automatically added. You can run the **kubectrl describe node** command to view the labels. The following is an example:

```
$ kubectrl describe node 192.168.0.212
Name:          192.168.0.212
Roles:        <none>
```

```
Labels:      beta.kubernetes.io/arch=amd64
             beta.kubernetes.io/os=linux
             failure-domain.beta.kubernetes.io/is-baremetal=false
             failure-domain.beta.kubernetes.io/region=*****
             failure-domain.beta.kubernetes.io/zone=*****
             kubernetes.io/arch=amd64
             kubernetes.io/availablezone=*****
             kubernetes.io/eniquota=12
             kubernetes.io/hostname=192.168.0.212
             kubernetes.io/os=linux
             node.kubernetes.io/subnetid=fd43acad-33e7-48b2-a85a-24833f362e0e
             os.architecture=amd64
             os.name=EulerOS_2.0_SP5
             os.version=3.10.0-862.14.1.5.h328.eulerosv2r7.x86_64
```

In workload scheduling, common node labels are as follows:

- **failure-domain.beta.kubernetes.io/region**: region where the node is located.
- **failure-domain.beta.kubernetes.io/zone**: availability zone to which the node belongs.
- **kubernetes.io/hostname**: host name of the node.

Kubernetes provides the **nodeSelector** field. When creating a workload, you can set this field to specify that the pod can be deployed only on a node with the specific label. The following example shows how to use a nodeSelector to deploy the pod only on the node with the **gpu=true** label.

```
apiVersion: v1
kind: Pod
metadata:
  name: nginx
spec:
  nodeSelector:      # Node selection. A pod is deployed on a node only when the node has the
                    # gpu=true label.
  gpu: true
...
```

Node affinity rules can achieve the same results. Compared with nodeSelector, node affinity rules seem more complex, but with a more expressive syntax. You can use the **spec.affinity.nodeAffinity** field to set node affinity. There are two types of node affinity:

- **requiredDuringSchedulingIgnoredDuringExecution**: Kubernetes cannot schedule the pod unless the rule is met.
- **PreferredDuringSchedulingIgnoredDuringExecution**: Kubernetes tries to find a node that meets the rule. If a matching node is not available, Kubernetes still schedules the pod.

 **NOTE**

In these two types of node affinity, **requiredDuringScheduling** or **preferredDuringScheduling** indicates that the pod can be scheduled to a node only when all the defined rules are met (required). **IgnoredDuringExecution** indicates that if the node label changes after Kubernetes schedules the pod, the pod continues to run and will not be rescheduled.

The following is an example of setting 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:
        nodeAffinity:
          requiredDuringSchedulingIgnoredDuringExecution:
            nodeSelectorTerms:
            - matchExpressions:
              - key: gpu
                operator: In
                values:
                - "true"
```

In this example, the scheduled node must contain a label with the key named **gpu**. The value of **operator** is to **In**, indicating that the label value must be in the **values** list. That is, the key value of the **gpu** label of the node is **true**. For details about other values of **operator**, see [Operator Value Description](#). Note that there is no such thing as **nodeAntiAffinity** because operators **NotIn** and **DoesNotExist** provide the same function.

The following describes how to check whether the rule takes effect. Assume that a cluster has three nodes.

```
$ kubectl get node
NAME          STATUS  ROLES  AGE  VERSION
192.168.0.212 Ready   <none> 13m  v1.15.6-r1-20.3.0.2.B001-15.30.2
192.168.0.94  Ready   <none> 13m  v1.15.6-r1-20.3.0.2.B001-15.30.2
192.168.0.97  Ready   <none> 13m  v1.15.6-r1-20.3.0.2.B001-15.30.2
```

Add the **gpu=true** label to the **192.168.0.212** node.

```
$ kubectl label node 192.168.0.212 gpu=true
node/192.168.0.212 labeled

$ kubectl get node -L gpu
NAME          STATUS  ROLES  AGE  VERSION          GPU
192.168.0.212 Ready   <none> 13m  v1.15.6-r1-20.3.0.2.B001-15.30.2  true
192.168.0.94  Ready   <none> 13m  v1.15.6-r1-20.3.0.2.B001-15.30.2
192.168.0.97  Ready   <none> 13m  v1.15.6-r1-20.3.0.2.B001-15.30.2
```

Create the Deployment. You can find that all pods are deployed on the **192.168.0.212** node.

```
$ kubectl create -f affinity.yaml
deployment.apps/gpu created

$ kubectl get pod -o wide
NAME          READY  STATUS  RESTARTS  AGE  IP          NODE
gpu-6df65c44cf-42xw4  1/1    Running  0         15s  172.16.0.37  192.168.0.212
```


gpu-6df65c44cf-jzjvs	1/1	Running	0	15s	172.16.0.36	192.168.0.212
gpu-6df65c44cf-zv5cl	1/1	Running	0	15s	172.16.0.38	192.168.0.212

Node Preference Rule

The preceding **requiredDuringSchedulingIgnoredDuringExecution** rule is a hard selection rule. There is another type of selection rule, that is, **preferredDuringSchedulingIgnoredDuringExecution**. It is used to specify which nodes are preferred during scheduling.

To achieve this effect, add a node attached with SAS disks to the cluster, add the **DISK=SAS** label to the node, and add the **DISK=SSD** label to the other three nodes.

```
$ kubectl get node -L DISK,gpu
NAME          STATUS    ROLES    AGE   VERSION          DISK    GPU
192.168.0.100 Ready    <none>   7h23m v1.15.6-r1-20.3.0.2.B001-15.30.2 SAS
192.168.0.212 Ready    <none>   8h    v1.15.6-r1-20.3.0.2.B001-15.30.2 SSD    true
192.168.0.94  Ready    <none>   8h    v1.15.6-r1-20.3.0.2.B001-15.30.2 SSD
192.168.0.97  Ready    <none>   8h    v1.15.6-r1-20.3.0.2.B001-15.30.2 SSD
```

Define a Deployment. Use the **preferredDuringSchedulingIgnoredDuringExecution** rule to set the weight of nodes with the SSD disk installed as **80** and nodes with the **gpu=true** label as **20**. In this way, pods are preferentially deployed on the nodes with the SSD disk installed.

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: gpu
  labels:
    app: gpu
spec:
  selector:
    matchLabels:
      app: gpu
  replicas: 10
  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:
        nodeAffinity:
          preferredDuringSchedulingIgnoredDuringExecution:
            - weight: 80
              preference:
                matchExpressions:
                  - key: DISK
                    operator: In
                    values:
                      - SSD
            - weight: 20
```

```

preference:
  matchExpressions:
  - key: gpu
    operator: In
    values:
    - "true"
    
```

After the deployment, there are five pods deployed on the node **192.168.0.212** (label: **DISK=SSD** and **GPU=true**), three pods deployed on the node **192.168.0.97** (label: **DISK=SSD**), and two pods deployed on the node **192.168.0.100** (label: **DISK=SAS**).

From the preceding output, you can find that no pods of the Deployment are scheduled to node **192.168.0.94** (label: **DISK=SSD**). This is because the node already has many pods on it and its resource usage is high. This also indicates that the **preferredDuringSchedulingIgnoredDuringExecution** rule defines a preference rather than a hard requirement.

```

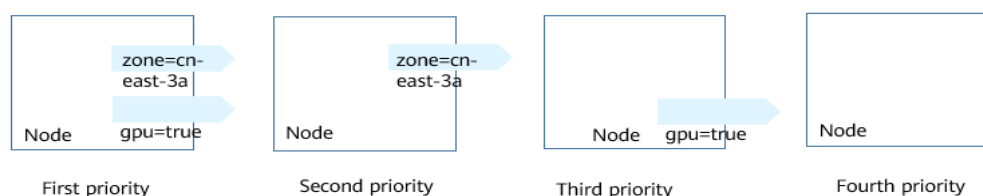
$ kubectl create -f affinity2.yaml
deployment.apps/gpu created
    
```

```

$ kubectl get po -o wide
NAME                READY  STATUS   RESTARTS  AGE  IP           NODE
gpu-585455d466-5bmcz 1/1    Running  0         2m29s 172.16.0.44 192.168.0.212
gpu-585455d466-cg2l6 1/1    Running  0         2m29s 172.16.0.63 192.168.0.97
gpu-585455d466-f2bt2 1/1    Running  0         2m29s 172.16.0.79 192.168.0.100
gpu-585455d466-hdb5n 1/1    Running  0         2m29s 172.16.0.42 192.168.0.212
gpu-585455d466-hkgvz 1/1    Running  0         2m29s 172.16.0.43 192.168.0.212
gpu-585455d466-mngvn 1/1    Running  0         2m29s 172.16.0.48 192.168.0.97
gpu-585455d466-s26qs 1/1    Running  0         2m29s 172.16.0.62 192.168.0.97
gpu-585455d466-sxtzm 1/1    Running  0         2m29s 172.16.0.45 192.168.0.212
gpu-585455d466-t56cm 1/1    Running  0         2m29s 172.16.0.64 192.168.0.100
gpu-585455d466-t5w5x 1/1    Running  0         2m29s 172.16.0.41 192.168.0.212
    
```

In the preceding example, the node scheduling priority is as follows. Nodes with both **SSD** and **gpu=true** labels have the highest priority. Nodes with the **SSD** label but no **gpu=true** label have the second priority (weight: 80). Nodes with the **gpu=true** label but no **SSD** label have the third priority. Nodes without any of these two labels have the lowest priority.

Figure 9-4 Scheduling priority



Workload Affinity (podAffinity)

Node affinity rules affect only the affinity between pods and nodes. Kubernetes also supports configuring inter-pod affinity rules. For example, the frontend and backend of an application can be deployed together on one node to reduce access latency. There are also two types of inter-pod affinity rules: **requiredDuringSchedulingIgnoredDuringExecution** and **preferredDuringSchedulingIgnoredDuringExecution**.

 NOTE

For workload affinity, `topologyKey` cannot be left blank when `requiredDuringSchedulingIgnoredDuringExecution` and `preferredDuringSchedulingIgnoredDuringExecution` are used.

Assume that the backend of an application has been created and has the **app=backend** label.

```
$ kubectl get po -o wide
NAME                 READY STATUS  RESTARTS  AGE   IP           NODE
backend-658f6cb858-dlrz8 1/1   Running  0         2m36s 172.16.0.67 192.168.0.100
```

You can configure the following pod affinity rule to deploy the frontend pods of the application to the same node as the backend pods.

```
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:
        podAffinity:
          requiredDuringSchedulingIgnoredDuringExecution:
            - topologyKey: kubernetes.io/hostname
              labelSelector:
                matchExpressions:
                  - key: app
                    operator: In
                    values:
                      - backend
```

Deploy the frontend and you can find that the frontend is deployed on the same node as the backend.

```
$ kubectl create -f affinity3.yaml
deployment.apps/frontend created

$ kubectl get po -o wide
NAME                 READY STATUS  RESTARTS  AGE   IP           NODE
backend-658f6cb858-dlrz8 1/1   Running  0         5m38s 172.16.0.67 192.168.0.100
frontend-67ff9b7b97-dsqzn 1/1   Running  0         6s    172.16.0.70 192.168.0.100
frontend-67ff9b7b97-hxm5t 1/1   Running  0         6s    172.16.0.71 192.168.0.100
frontend-67ff9b7b97-z8pdb 1/1   Running  0         6s    172.16.0.72 192.168.0.100
```

The **topologyKey** field is used to divide topology domains to specify the selection range. If the label keys and values of nodes are the same, the nodes are considered to be in the same topology domain. Then, the contents defined in the following rules are selected. The effect of **topologyKey** is not fully demonstrated in the preceding example because all the nodes have the **kubernetes.io/hostname** label, that is, all the nodes are within the range.

To see how **topologyKey** works, assume that the backend of the application has two pods, which are running on different nodes.

```
$ kubectl get po -o wide
NAME                READY STATUS RESTARTS AGE IP          NODE
backend-658f6cb858-5bpd6 1/1 Running 0      23m 172.16.0.40 192.168.0.97
backend-658f6cb858-dlrz8 1/1 Running 0      2m36s 172.16.0.67 192.168.0.100
```

Add the **prefer=true** label to nodes **192.168.0.97** and **192.168.0.94**.

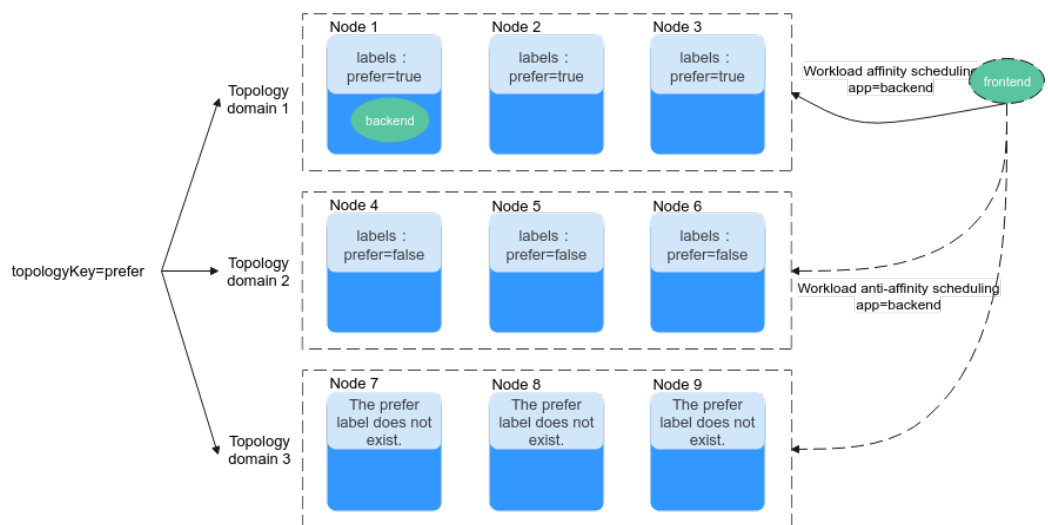
```
$ kubectl label node 192.168.0.97 prefer=true
node/192.168.0.97 labeled
$ kubectl label node 192.168.0.94 prefer=true
node/192.168.0.94 labeled

$ kubectl get node -L prefer
NAME           STATUS ROLES AGE VERSION                                PREFER
192.168.0.100 Ready <none> 44m v1.15.6-r1-20.3.0.2.B001-15.30.2
192.168.0.212 Ready <none> 91m v1.15.6-r1-20.3.0.2.B001-15.30.2
192.168.0.94  Ready <none> 91m v1.15.6-r1-20.3.0.2.B001-15.30.2 true
192.168.0.97  Ready <none> 91m v1.15.6-r1-20.3.0.2.B001-15.30.2 true
```

If the **topologyKey** of **podAffinity** is set to **prefer**, the node topology domains are divided as shown in [Figure 9-5](#).

```
affinity:
  podAffinity:
    requiredDuringSchedulingIgnoredDuringExecution:
      - topologyKey: prefer
        labelSelector:
          matchExpressions:
            - key: app
              operator: In
              values:
                - backend
```

Figure 9-5 Topology domains



During scheduling, node topology domains are divided based on the **prefer** label. In this example, **192.168.0.97** and **192.168.0.94** are divided into the same topology domain. If a pod with the **app=backend** label runs in the topology domain, even if not all nodes in the topology domain run the pod with the **app=backend** label (in this example, only the **192.168.0.97** node has such a pod), **frontend** is also deployed in this topology domain (**192.168.0.97** or **192.168.0.94**).

```
$ kubectl create -f affinity3.yaml
deployment.apps/frontend created
```

```
$ kubectl get po -o wide
NAME                READY STATUS RESTARTS AGE IP          NODE
backend-658f6cb858-5bpd6  1/1   Running 0      26m 172.16.0.40 192.168.0.97
backend-658f6cb858-dlrz8  1/1   Running 0      5m38s 172.16.0.67 192.168.0.100
frontend-67ff9b7b97-dsqzn 1/1   Running 0      6s    172.16.0.70 192.168.0.97
frontend-67ff9b7b97-hxm5t 1/1   Running 0      6s    172.16.0.71 192.168.0.97
frontend-67ff9b7b97-z8pdb 1/1   Running 0      6s    172.16.0.72 192.168.0.97
```

Workload Anti-Affinity (podAntiAffinity)

Unlike the scenarios in which pods are preferred to be scheduled onto the same node, sometimes, it could be the exact opposite. For example, if certain pods are deployed together, they will affect the performance.

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.

The following is an example of defining an anti-affinity rule. This rule divides node topology domains by the **kubernetes.io/hostname** label. If a pod with the **app=frontend** label already exists on a node in the topology domain, pods with the same label cannot be scheduled to other nodes in the topology domain.

```
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:
    requiredDuringSchedulingIgnoredDuringExecution:
      - topologyKey: kubernetes.io/hostname # Topology domain of the node
        labelSelector: # Pod label matching rule
          matchExpressions:
            - key: app
              operator: In
          values:
            - frontend
```

Create an anti-affinity rule and view the deployment result. In the example, node topology domains are divided by the **kubernetes.io/hostname** label. The label values of nodes with the **kubernetes.io/hostname** label are different, so there is only one node in a topology domain. If a **frontend** pod already exists in a topology domain, pods with the same label will not be scheduled to the topology domain. In this example, there are only four nodes. Therefore, there is one pod which is in the **Pending** state and cannot be scheduled.

```
$ kubectl create -f affinity4.yaml
deployment.apps/frontend created

$ kubectl get po -o wide
NAME                                READY STATUS RESTARTS AGE IP          NODE
frontend-6f686d8d87-8dlsc          1/1   Running 0       18s 172.16.0.76 192.168.0.100
frontend-6f686d8d87-d6l8p          0/1   Pending 0       18s <none>      <none>
frontend-6f686d8d87-hgcq2          1/1   Running 0       18s 172.16.0.54 192.168.0.97
frontend-6f686d8d87-q7cfq          1/1   Running 0       18s 172.16.0.47 192.168.0.212
frontend-6f686d8d87-xl8hx          1/1   Running 0       18s 172.16.0.23 192.168.0.94
```

Operator Value Description

You can use the **operator** field to set the logical relationship of the usage rule. The value of **operator** can be:

- **In**: The label of the affinity or anti-affinity object is in the label value list (**values** field).
- **NotIn**: The label of the affinity or anti-affinity object is not in the label value list (**values** field).
- **Exists**: The affinity or anti-affinity object has a specified label name.
- **DoesNotExist**: The affinity or anti-affinity object does not have the specified label name.
- **Gt**: (available only for node affinity) The label value of the scheduled node is greater than the list value (string comparison).
- **Lt**: (available only for node affinity) The label value of the scheduling node is less than the list value (string comparison).

9.3.10 Taints and Tolerations

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.

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"> Pods running on the node will be evicted immediately. Inactive pods will not be scheduled to the node. 	<ul style="list-style-type: none"> If the tolerance time window tolerationSeconds is not specified, pods can run on this node all the time. If the tolerance time window tolerationSeconds is specified, pods still run on the node with taints within the time window. After the time expires, the pods will be evicted.
PreferNoSchedule	<ul style="list-style-type: none"> Pods running on the node will not be evicted. Inactive pods will not be scheduled to the node to the best extend. 	Pods can run on this node all the time.
NoSchedule	<ul style="list-style-type: none"> Pods running on the node will not be evicted. Inactive pods will not be scheduled to the node. 	Pods can run on this node all the time.

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 9-15 Parameters for configuring a taint tolerance policy

Parameter	Description
Taint key	Key of a node taint
Operator	<ul style="list-style-type: none"> Equal: Exact match 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. Exists: matches only 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.
Taint value	Taint value specified if the operator is set to Equal .

Parameter	Description
Taint Policy	<ul style="list-style-type: none"> • All: All taint policies are matched. • NoSchedule: Only the NoSchedule taint is matched. • PreferNoSchedule: Only the PreferNoSchedule taint is matched. • NoExecute: Only the NoExecute taint is matched.
Toleration Time Window	<p>tolerationSeconds, which is configurable only when Taint Policy is set to NoExecute.</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
```

9.3.11 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 9-16 Pod annotations

Annotation	Description	Default Value
kubernetes.AOM.log.stdout	<p>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.</p> <p>Example:</p> <ul style="list-style-type: none"> Collecting none of the stdout logs: kubernetes.AOM.log.stdout: '[]' Collecting stdout logs of container-1 and container-2: kubernetes.AOM.log.stdout: '["container-1","container-2"]' 	None
metrics.alpha.kubernetes.io/custom-endpoints	<p>Parameter for reporting AOM monitoring metrics that you specify.</p> <p>For details, see Monitoring Custom Metrics on AOM.</p>	None
kubernetes.io/ingress-bandwidth	<p>Ingress bandwidth of a pod.</p> <p>For details, see Configuring QoS Rate Limiting for Inter-Pod Access.</p>	None
kubernetes.io/egress-bandwidth	<p>Egress bandwidth of a pod.</p> <p>For details, see Configuring QoS Rate Limiting for Inter-Pod Access.</p>	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:

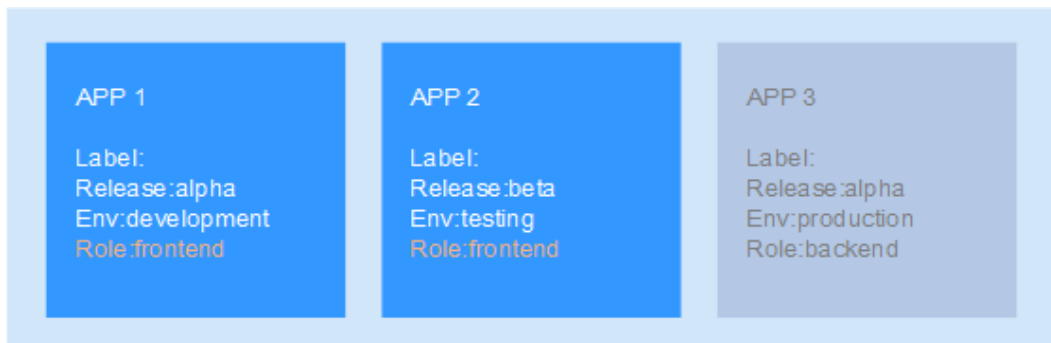
```
...
spec:
  selector:
    matchLabels:
      app: nginx
      version: v1
  template:
    metadata:
      labels:
        app: nginx
        version: v1
```

```
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 9-6 Label example



For example, if **key/value** is set to **role/backend**, APP 3 will be selected for affinity scheduling. For details, see [Workload Affinity \(podAffinity\)](#).

9.4 Accessing a Container

Scenario

If you encounter unexpected problems when using a container, you can log in to the container to debug it.

Logging In to a Container Using kubectl

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 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

9.5 Managing Workloads and Jobs

Scenario

After a workload is created, you can upgrade, monitor, roll back, or delete the workload, as well as edit its YAML file.

Table 9-17 Workload/Job management

Operation	Description
Monitor	You can view the CPU and memory usage of workloads and pods on the CCE console.
View Log	You can view the logs of workloads.
Upgrade	You can replace images or image tags to quickly upgrade Deployments, StatefulSets, and DaemonSets without interrupting services.
Edit YAML	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.
Roll Back	Only Deployments can be rolled back.
Redeploy	You can redeploy a workload. After the workload is redeployed, all pods in the workload will be restarted.
Enabling/ Disabling the Upgrade	Only Deployments support this operation.
Manage Label	Labels are attached to workloads as key-value pairs to manage and select workloads. Jobs and Cron Jobs do not support this operation.
Delete	You can delete a workload or job that is no longer needed. Deleted workloads or jobs cannot be recovered.
View Events	You can view event names, event types, number of occurrences, Kubernetes events, first occurrence time, and last occurrence time.
Stop/Start	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.
- 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 the **View Log** of the target workload.
On the displayed **View Log** window, you can view logs.

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 [Using ICAgent to Collect Container Logs](#).

----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 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, 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, 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**.

----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.
- If a Deployment is being upgraded, 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.

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 **Add**, enter a key and a value, and click **OK**.

 **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, 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 Deployments as an example to illustrate how to view events of a workload. To view the event of a job or cron job, 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

10 Scheduling

10.1 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	Reference
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 thus 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.	CPU Policy

GPU Scheduling

CCE schedules heterogeneous GPU resources in clusters and allows GPUs to be used in containers.

Function	Description	Reference
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.	Default GPU Scheduling in Kubernetes

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	Reference
NUMA affinity scheduling	Volcano targets to lift the limitation to make scheduler NUMA topology aware so that: <ul style="list-style-type: none"> Pods are not scheduled to the nodes that NUMA topology does not match. Pods are scheduled to the best node for NUMA topology. 	NUMA Affinity Scheduling

10.2 CPU Scheduling

10.2.1 CPU Policy

Scenarios

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. 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 and CPU cache affinity and scheduling latency significantly affect workload performance, kubelet allows alternative CPU management policies (CPU binding) to determine some placement preferences on the node. The CPU manager preferentially allocates resources on a socket and full physical cores to avoid interference.

Enabling the CPU Management Policy

A **CPU management policy** is specified by the kubelet flag `--cpu-manager-policy`. By default, Kubernetes supports the following policies:

- 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 GPU requests to be granted increased CPU affinity and exclusivity on the node.

When creating a cluster, you can configure the CPU management policy in **Advanced Settings**.

You can also configure the policy in a node pool. The configuration will change the kubelet flag `--cpu-manager-policy` on the node. Log in to the CCE console, click the cluster name, access the cluster details page, and choose **Nodes** in the navigation pane. On the page displayed, click the **Node Pools** tab. Choose **More > Manage** in the **Operation** column of the target node pool, and change the value of `cpu-manager-policy` to **static**.

Allowing Pods to Exclusively Use the CPU Resources

Prerequisites:

- Enable the **static** policy on the node. For details, see [Enabling the CPU Management Policy](#).
- 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 [Scheduling Policy \(Affinity/Anti-affinity\)](#) 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
```

10.3 GPU Scheduling

10.3.1 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 gpu-device-plugin (previously gpu-beta add-on) has been installed. During the installation, select the GPU driver on the node. For details, see [gpu-device-plugin](#).
- gpu-device-plugin mounts the driver directory to `/usr/local/nvidia/lib64`. To use GPU resources in a container, add `/usr/local/nvidia/lib64` to the `LD_LIBRARY_PATH` environment variable.

Generally, you can use any of the following methods to add a file:

- Configure the `LD_LIBRARY_PATH` environment variable in the Dockerfile used for creating an image. (Recommended)

```
ENV LD_LIBRARY_PATH /usr/local/nvidia/lib64:$LD_LIBRARY_PATH
```
- Configure the `LD_LIBRARY_PATH` environment variable in the image startup command.

```
/bin/bash -c "export LD_LIBRARY_PATH=/usr/local/nvidia/lib64:$LD_LIBRARY_PATH && ..."
```
- 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 GPUs on the CCE console, select the GPU quota and specify the percentage of GPUs reserved for the container when creating a workload.

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
```

10.4 Volcano Scheduling

10.4.1 NUMA Affinity Scheduling

Background

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. However, in workloads where CPU cache affinity and scheduling latency significantly affect workload performance, the kubelet allows alternative CPU management policies to determine some placement preferences on the node.

Both the CPU Manager and Topology Manager are kubelet components, but they have the following limitations:

- The scheduler is not topology-aware. Therefore, the workload may be scheduled on a node and then fail on the node due to the Topology Manager. This is unacceptable for TensorFlow jobs. If any worker or ps failed on node, the job will fail.
- The managers are node-level that results in an inability to match the best node for NUMA topology in the whole cluster.

For more information, see <https://github.com/volcano-sh/volcano/blob/master/docs/design/numa-aware.md>.

Volcano targets to resolve the limitation to make scheduler NUMA topology aware so as to achieve the following:

- Do not schedule pods to the nodes which NUMA topology does not match.
- Schedule pods to the best node for NUMA topology.

Application Scope

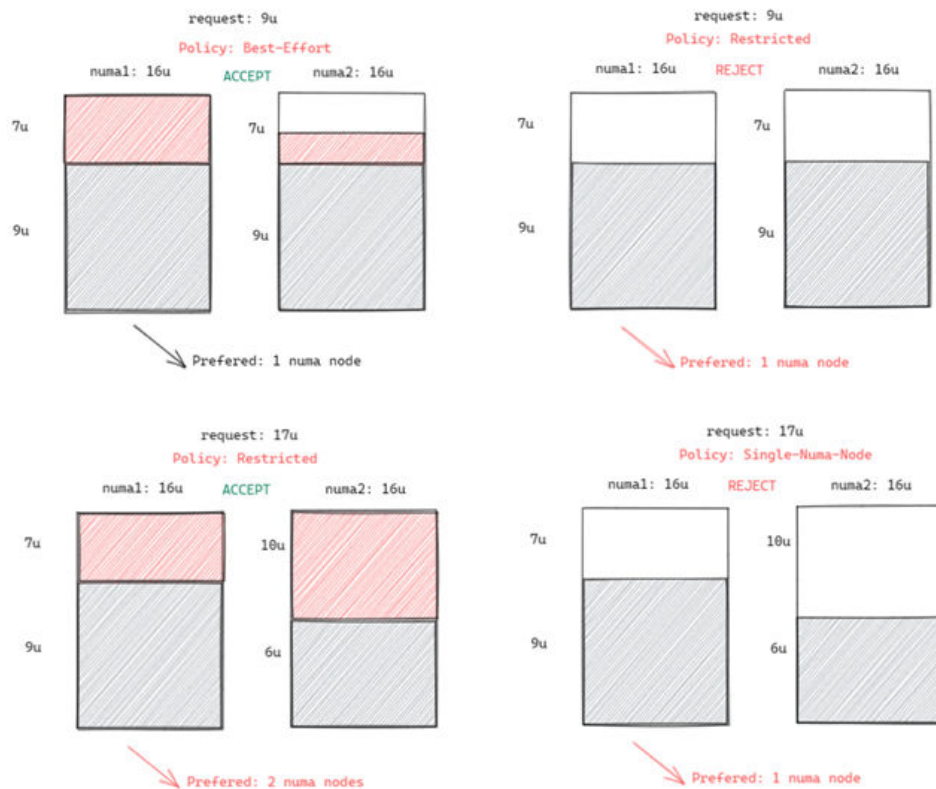
- Support CPU resource topology scheduling
- Support pod-level topology policies

Scheduling Prediction

For pods with the topology policy, predicate the matched node list.

policy	action
none	1. No filter action
best-effort	1. Filter out the node with the topology policy best-effort .
restricted	1. Filter out the node with the topology policy restricted . 2. Filter out the node that the CPU topology meets the CPU requirements for restricted .
single-numa-node	1. Filter out the node with the topology policy single-numa-node . 2. Filter out the node that the CPU topology meets the CPU requirements for single-numa-node .

Figure 10-1 Comparison of NUMA scheduling policies



Scheduling Priority

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:

$$\text{score} = \text{weight} * (100 - 100 * \text{numaNodeNum} / \text{maxNumaNodeNum})$$

Parameter description:

- **weight**: indicates the weight of NUMA Aware Plugin.
- **numaNodeNum**: indicates the number of NUMA nodes required for running the pod on the worker node.
- **maxNumaNodeNum**: indicates the maximum number of NUMA nodes in a pod of all worker nodes.

Enabling Volcano to Support NUMA Affinity Scheduling

Step 1 Enable the CPU management policy. For details, see [Enabling the CPU Management Policy](#).

Step 2 Configure a CPU topology policy.

1. Log in to the CCE console, click the cluster name, access the cluster details page, and choose **Nodes** in the navigation pane. On the page displayed, click the **Node Pools** tab. Choose **More > Manage** in the **Operation** column of the target node pool.
2. Change the value of **topology-manager-policy** under **kubelet** to the required CPU topology policy.

The valid topology policies are **none**, **best-effort**, **restricted**, and **single-numa-node**. For details about these policies, see [Scheduling Prediction](#).

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 access the cluster console. In the navigation pane, choose **Add-ons**. On the right of the page, locate the **volcano** add-on and click **Edit**. In the **Parameters** area, configure Volcano scheduler parameters.

```
{
  "ca_cert": "",
  "default_scheduler_conf": {
    "actions": "allocate, backfill",
    "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"
    },
    {
      // add this also enable resource_exporter
      "name": "numa-aware",
      // the weight of the NUMA Aware Plugin
      "arguments": {
        "weight": "10"
      }
    }
  ]
},
{
  "plugins": [
    {
      "name": "nodelocalvolume"
    },
    {
      "name": "nodeemptydirvolume"
    },
    {
      "name": "nodeCSIScheduling"
    },
    {
      "name": "networkresource"
    }
  ]
}
],
"server_cert": "",
"server_key": ""
}

```

volcano earlier than 1.7.1

1. The **resource_exporter_enable** parameter is enabled for the volcano add-on to collect node NUMA information.

```

{
  "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

```

2. 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"
    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

Using Volcano to Support NUMA Affinity Scheduling

- Step 1** Configure NUMA affinity for Deployments. The following is an example:

```

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 # set 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
```

Step 2 Create a volcano job and use NUMA affinity.

```
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 3 Check the NUMA usage.

```
# 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

# 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}

----End
```

11 Network

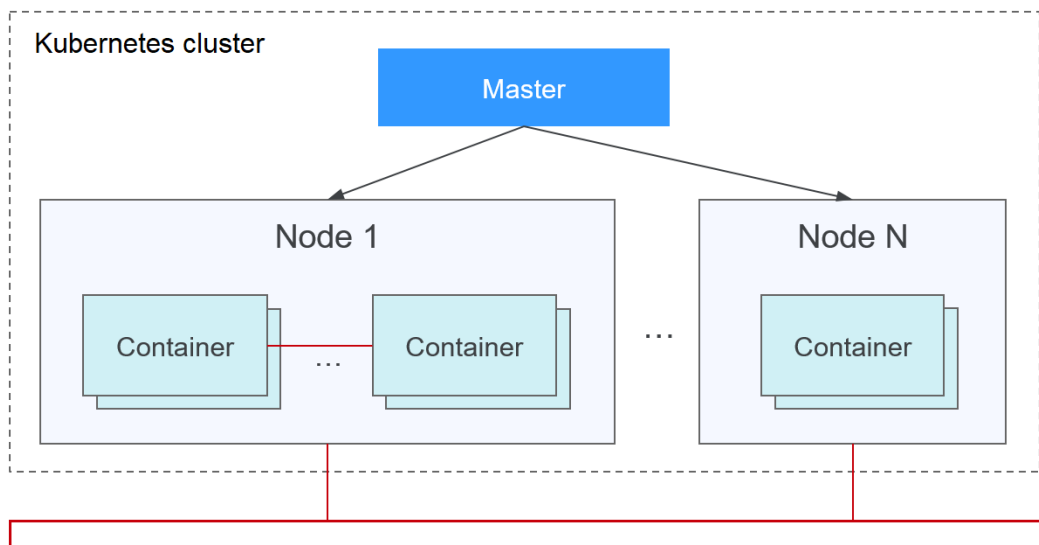
11.1 Overview

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 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 applies 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 a cluster can be directly accessed from outside the cluster.
- Developed by CCE, Cloud Native Network 2.0 deeply integrates Elastic Network Interfaces (ENIs) and Sub Network Interfaces (sub-ENIs) of VPC. Container IP addresses are allocated from the VPC CIDR block. ELB passthrough networking is supported to direct access requests to containers. Security groups and elastic IPs (EIPs) are bound to deliver high performance.

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](#).

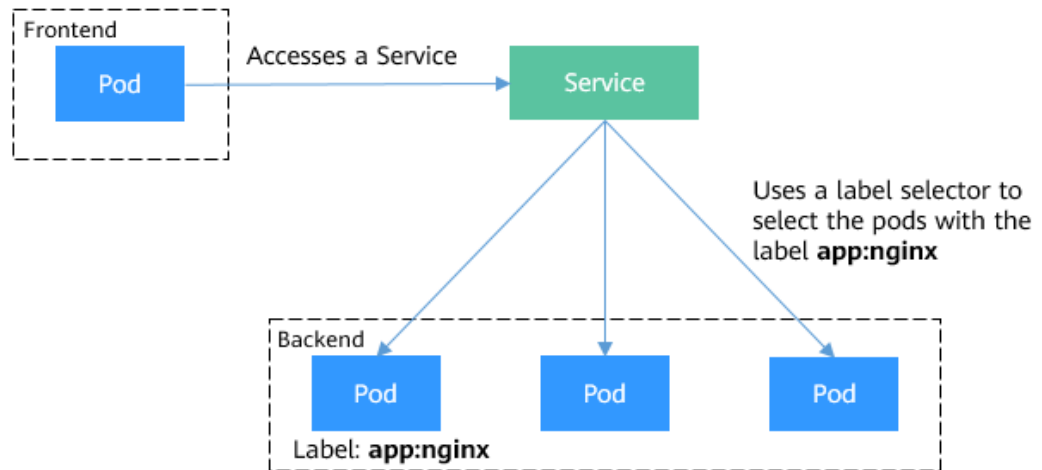
- **Service Network**

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 11-1 Accessing pods through a Service



You can configure the following types of Services:

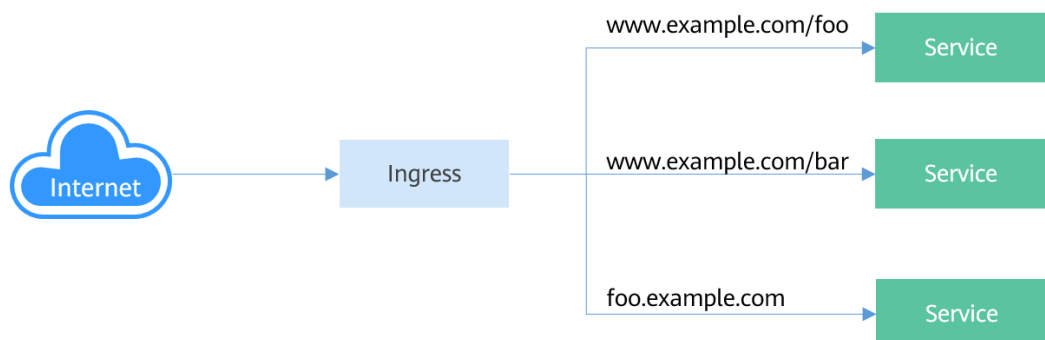
- ClusterIP: used to make the Service only reachable from within a cluster.
- NodePort: used for access from outside a cluster. A NodePort Service is accessed through the port on the node.
- LoadBalancer: used for access from outside a cluster. It is an extension of NodePort, to which a load balancer 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 11-2 Ingress-Service



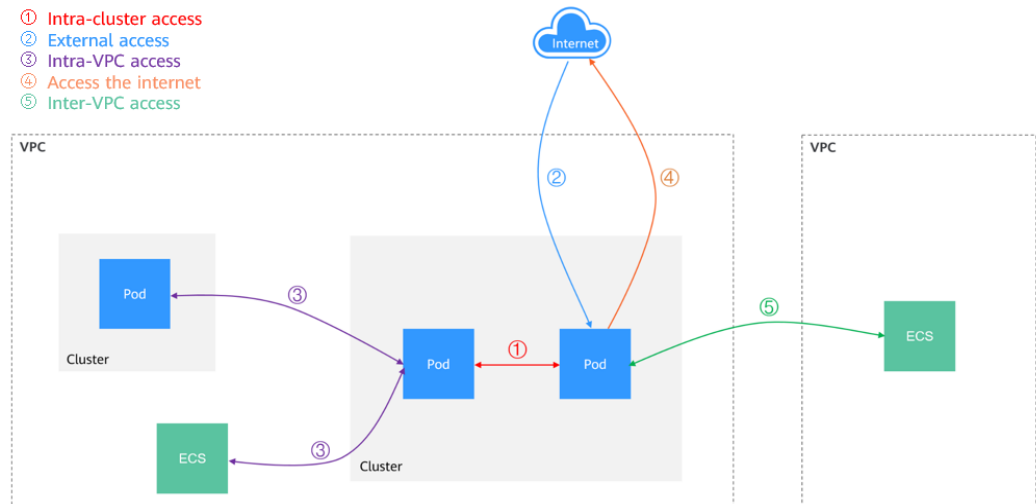
For details about the ingress, see [Overview](#).

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 Public Networks from a Container](#).

Figure 11-3 Network access diagram



11.2 Container Network Models

11.2.1 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:

- [Tunnel network](#)
- [VPC network](#)
- [Cloud Native Network 2.0](#)

Network Model Comparison

Table 11-1 describes the differences of network models supported by CCE.

 **CAUTION**

After a cluster is created, the network model cannot be changed.

Table 11-1 Network model comparison

Dimension	Tunnel Network	VPC Network	Cloud Native Network 2.0
Application scenarios	<ul style="list-style-type: none"> • Common container service scenarios • Scenarios that do not have high requirements on network latency and bandwidth 	<ul style="list-style-type: none"> • Scenarios that have high requirements on network latency and bandwidth • Containers can communicate with VMs using a microservice registration framework, such as Dubbo and CSE. 	<ul style="list-style-type: none"> • Scenarios that have high requirements on network latency, bandwidth, and performance • Containers can communicate with VMs using a microservice registration framework, such as Dubbo and CSE.
Core technology	OVS	IPvlan and VPC route	VPC ENI/sub-ENI
Applicable clusters	CCE cluster	CCE cluster	CCE Turbo cluster
Network isolation	Kubernetes native NetworkPolicy for pods	No	Pods support security group isolation.
Passthrough networking	No	No	Yes

Dimension	Tunnel Network	VPC Network	Cloud Native Network 2.0
IP address management	<ul style="list-style-type: none"> The container CIDR block is allocated separately. CIDR blocks are divided by node and can be dynamically allocated (CIDR blocks can be dynamically added after being allocated.) 	<ul style="list-style-type: none"> The container CIDR block is allocated separately. CIDR blocks are divided by node and statically allocated (the CIDR block cannot be changed after a node is created). 	The container CIDR block is divided from the VPC subnet and does not need to be allocated separately.
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.	The container network is integrated with the VPC network, eliminating performance loss.
Networking scale	A maximum of 2,000 nodes are supported.	<p>A maximum of 2000 nodes are supported, which is restricted by the VPC routing capability.</p> <p>Each time a node is added to the cluster, a route is added to the VPC route tables. Therefore, the cluster scale is limited by the VPC route tables.</p>	A maximum of 2,000 nodes are supported.

NOTICE

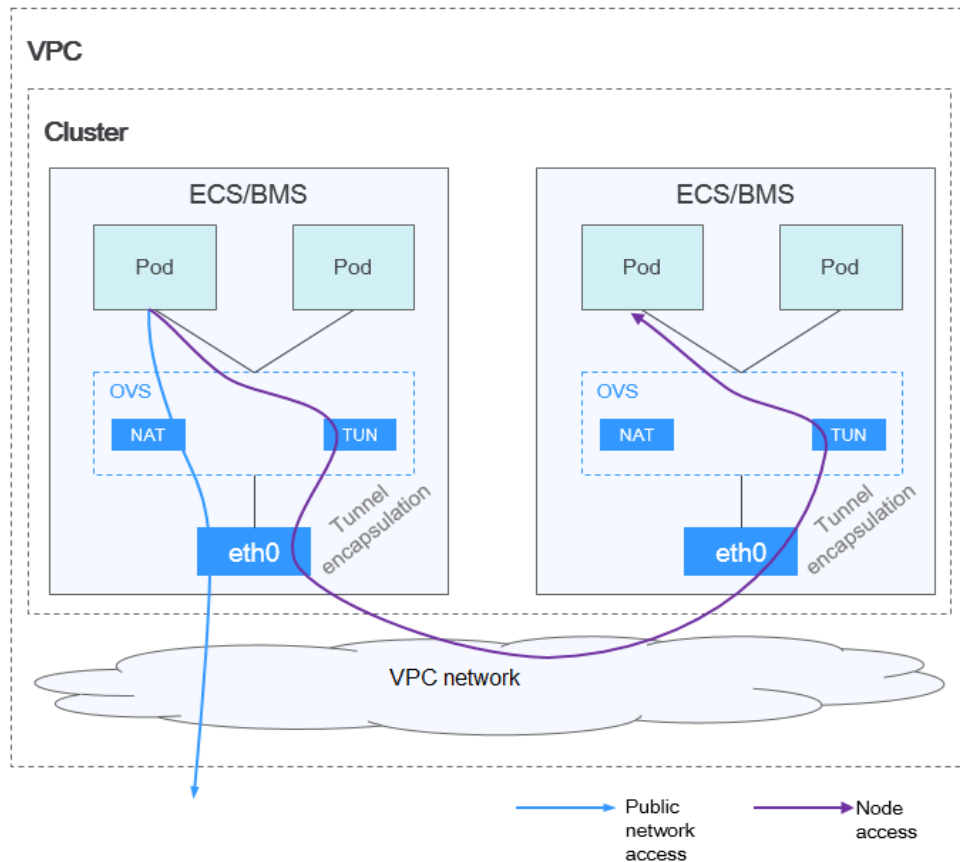
1. The scale of a cluster that uses the VPC network model is limited by the custom routes of the VPC. Therefore, estimate the number of required nodes before creating a cluster.
 2. The scale of a cluster that uses the Cloud Native Network 2.0 model depends on the size of the VPC subnet CIDR block selected for the network attachment definition. Before creating a cluster, evaluate the scale of your cluster.
 3. 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.
 4. Do not change the mask of the primary CIDR block on the VPC after a cluster is created. Otherwise, the network will be abnormal.
-

11.2.2 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 11-4 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 [Network Policies](#).
- Bandwidth limits are supported.
- Large-scale networking is supported.

Disadvantages

- High encapsulation overhead, complex networking, and low performance
- Pods cannot directly use functionalities such as EIPs and security groups.
- External networks cannot be directly connected to container IP addresses.

Applicable Scenarios

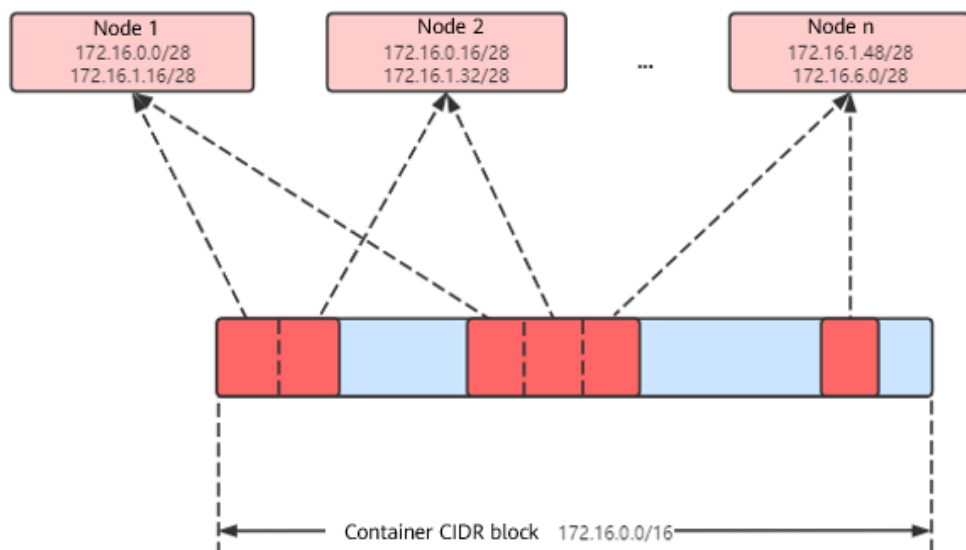
- Low requirements on performance: As the container tunnel network requires additional VXLAN tunnel encapsulation, it has about 5% to 15% of performance loss when compared with the other two container network models. Therefore, the container tunnel network applies to the scenarios that do not have high 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:

- 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 11-5 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.

Example of Container Tunnel Network Access

Create a cluster that uses the container tunnel network model. Create a Deployment in 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>

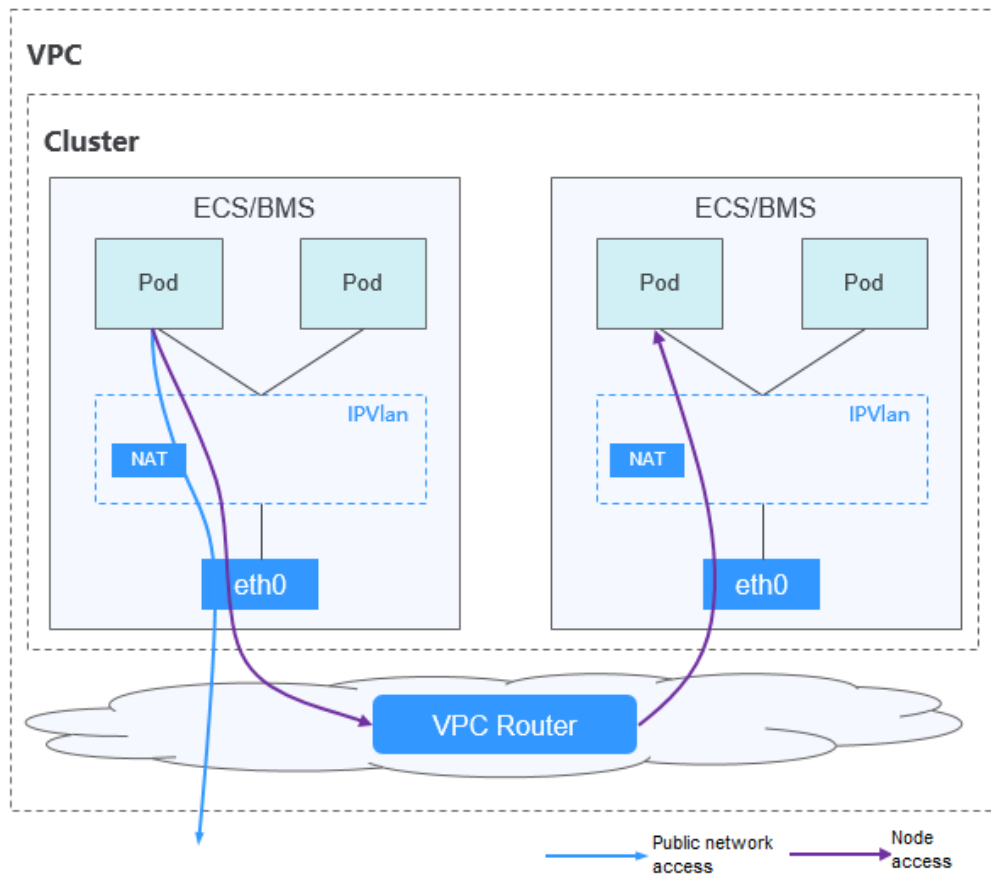
<p><em>Thank you for using nginx.</em></p>
</body>
</html>
```

11.2.3 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 ECSs in the same VPC outside the cluster.

Figure 11-6 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.
- In the same VPC, the external network of the cluster can be directly connected to the container IP address.

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

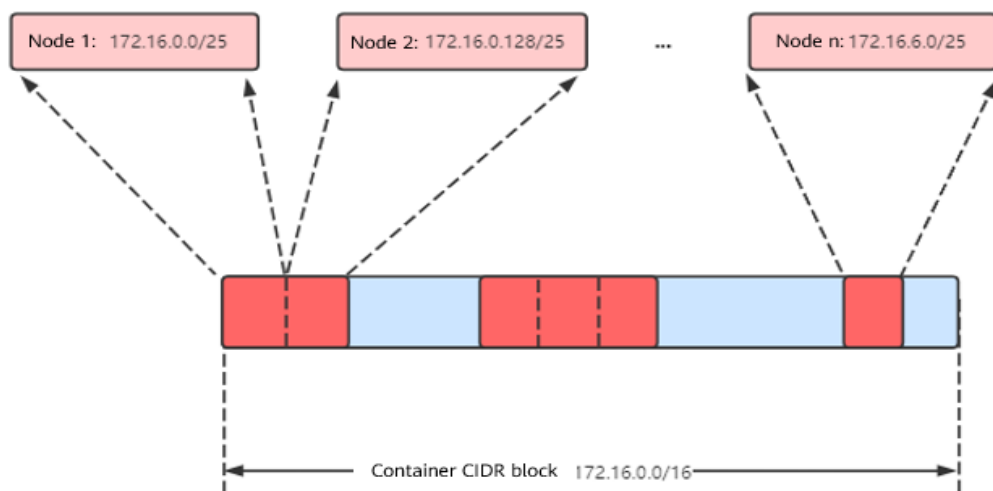
- High performance requirements: As no tunnel encapsulation is required, the VPC network model delivers the performance close to that of a VPC network when compared with the container tunnel network model. Therefore, the VPC network model applies to scenarios that have high requirements on performance, 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 11-7 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. 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 in 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, if you access the IP address of the pod from an ECS (outside the cluster) in the same VPC, the access is successful. This is a feature of VPC networking. Pods can be directly accessed from any node locating outside of the cluster and in the same VPC as that of the pods using the pods' IP addresses.

Pods can be accessed from nodes or pods in the same cluster. In the following example, you can directly access the pods in 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>

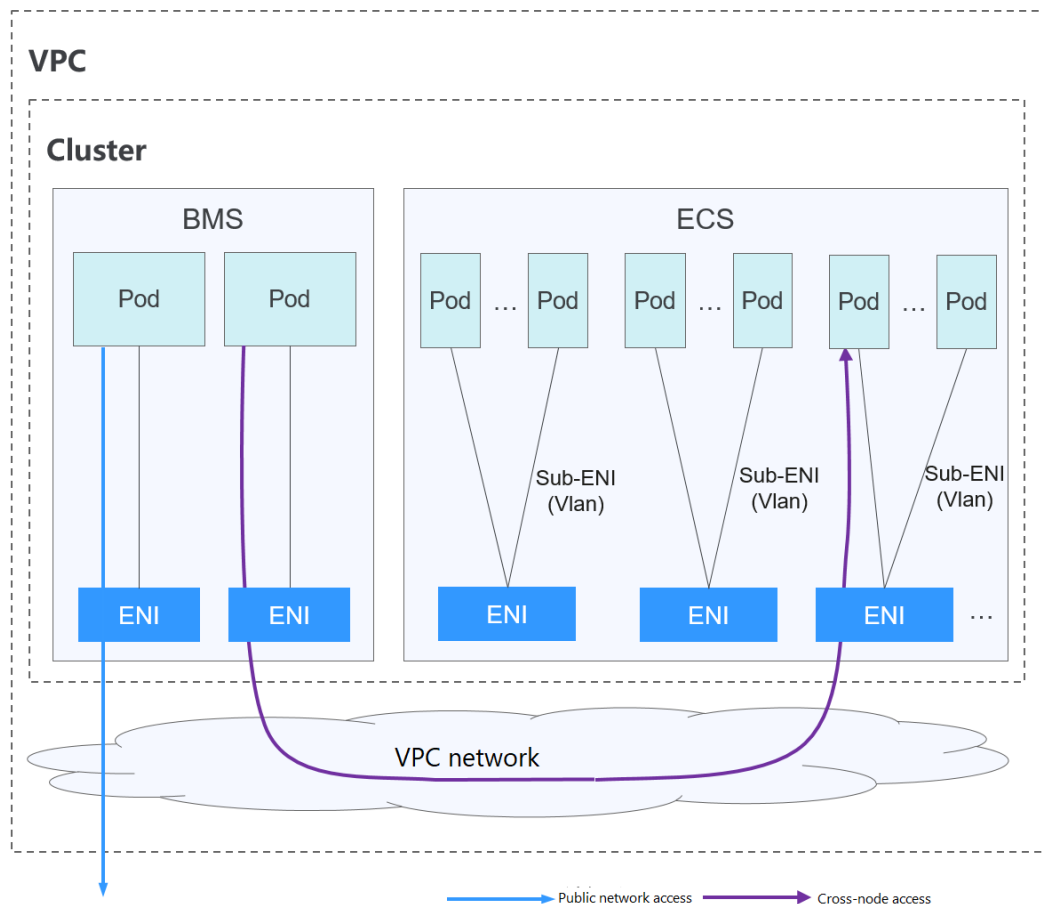
<p><em>Thank you for using nginx.</em></p>
</body>
</html>
```

11.2.4 Cloud Native 2.0 Network

Model Definition

Developed by CCE, Cloud Native 2.0 network deeply integrates Elastic Network Interfaces (ENIs) and sub-ENIs of Virtual Private Cloud (VPC). Container IP addresses are allocated from the VPC CIDR block. ELB passthrough networking is supported to direct access requests to containers. Security groups and elastic IPs (EIPs) are bound to deliver high performance.

Figure 11-8 Cloud Native 2.0 network model



Pod-to-pod communication

- On the same node: Packets are forwarded through the VPC ENI or sub-ENI.
- Across nodes: Packets are forwarded through the VPC ENI or sub-ENI.

Advantages and Disadvantages

Advantages

- As the container network directly uses VPC, it is easy to locate network problems and provide the highest performance.
- External networks in a VPC can be directly connected to container IP addresses.
- The load balancing, security group, and EIP capabilities provided by VPC can be directly used by pods.

Disadvantages

The container network directly uses VPC, which occupies the VPC address space. Therefore, you must properly plan the container CIDR block before creating a cluster.

Application Scenarios

- High performance requirements and use of other VPC network capabilities: Cloud Native Network 2.0 directly uses VPC, which delivers almost the same performance as the VPC network. Therefore, it applies to scenarios that have high requirements on bandwidth and latency.
- Large-scale networking: Cloud Native Network 2.0 supports a maximum of 2000 ECS nodes and 100,000 containers.

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. All subnets (including those created from the secondary CIDR block) in the VPC where the cluster resides cannot conflict with the container and Service CIDR blocks.
- 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.

In the Cloud Native Network 2.0 model, the container CIDR block and node CIDR block share the network 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 resources.

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 2.0 Network Access

Create a CCE Turbo cluster, which contains three ECS nodes.

Access the details page of one node. You can see that the node has one primary ENI and one extended ENI, and both of them are ENIs. The extended ENI belongs to the container CIDR block and is used to mount a sub-ENI to the pod.

Create a Deployment in the cluster.

```
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
```

View the created pod.

```
$ kubectl get pod -owide
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-5bdc5699b7-6dzt5  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 mounted to the ENI (extended ENI) of the node.

For example, the extended ENI of node 10.1.0.167 is 10.1.17.172. On the **Network Interfaces** page of the Network Console, you can see that three sub-ENIs are mounted to the extended ENI 10.1.17.172, which is the IP address of the pod.

In the VPC, the IP address of the pod can be successfully accessed.

11.3 Service

11.3.1 Overview

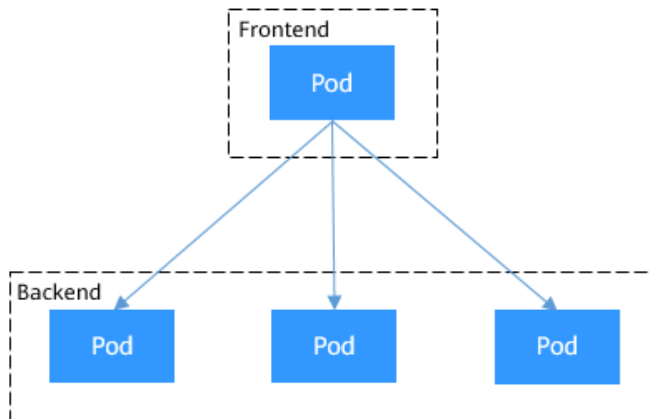
Direct Access to a Pod

After a pod is created, the following problems may occur if you directly access to the pod:

- The pod can be deleted and created again 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 11-9](#). Three pods are running in the backend, which are independent and replaceable. When a backend pod is created again, the new pod is assigned with a new IP address, of which the frontend pod is unaware.

Figure 11-9 Inter-pod access

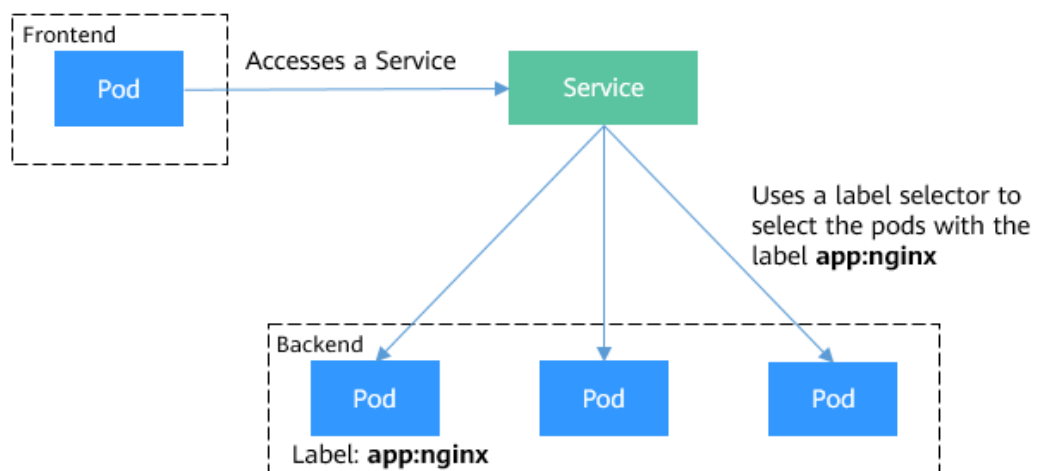


Using Services for Pod Access

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, performs 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 11-10](#).

Figure 11-10 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 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.

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.

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.

You can set this parameter when creating a Service of the NodePort type on the CCE console.

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. In this scenario, Services may fail to be accessed from within the cluster. For details, see [Why a Service Fail to Be Accessed from Within the Cluster](#).

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.
- CCE Turbo clusters using Cloud Native 2.0 networking do not support node-level service affinity.

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)
		<p>Other containers on the same node as the service pod</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 is successful.</p>	<p>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>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)
		Other containers on different nodes from the service pod	Access the IP address and NodePort on the node where the server is located: The access is successful. Access the IP address and NodePort on a node other than the node where the server is located: The access failed.	Access the IP address and NodePort on the node where the server is located: The access is successful. Access the IP address and NodePort on a node other than the node where the server is located: The access failed.	Access the IP address and NodePort on the node where the server is located: The access is successful. Access the IP address and NodePort on a node other than the node where the server is located: The access failed.	Access the IP address and NodePort on the node where the server is located: The access is successful. Access the IP address and NodePort on a node other than the node where the server is located: The access failed.
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.
		Other containers on the 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)
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
```

11.3.2 ClusterIP

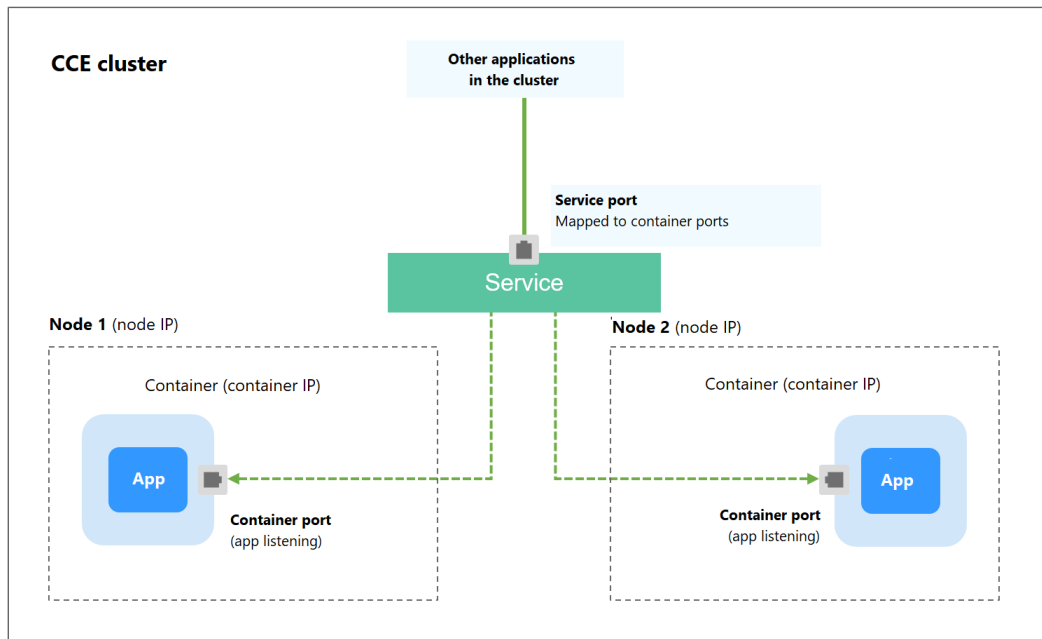
Scenario

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 11-11 shows the mapping relationships between access channels, container ports, and access ports.

Figure 11-11 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** Choose **Networking** in the navigation pane and click **Create Service** in the upper right corner.
- Step 3** Set intra-cluster access parameters.
 - **Service Name:** Service name, which can be the same as the workload name.

- **Service Type:** Select **ClusterIP**.
- **Namespace:** Namespace to which the workload belongs.
- **Selector:** Add a label and click **Confirm**. A Service selects a pod based on the added label. You can also click **Reference Workload Label** to reference the label of an existing workload. In the dialog box that is displayed, select a workload and click **OK**.
- **Port**
 - **Protocol:** protocol used by the Service.
 - **Service Port:** port used by the Service. The port number ranges from 1 to 65535.
 - **Container Port:** port on which the workload listens. For example, Nginx uses port 80 by default.

Step 4 Click **OK**.

----End

Setting the Access Type Using kubectl

You can run kubectl commands to set the access type (Service). This section uses an 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 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><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>  
...
```

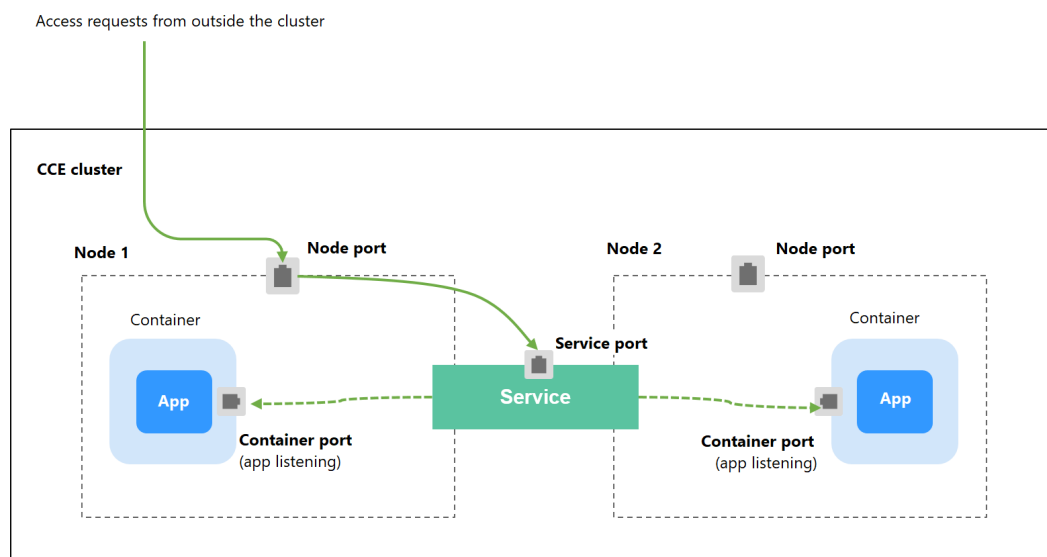
----End

11.3.3 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 11-12 NodePort access



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.
- CCE Turbo clusters support only cluster-level service affinity.
- 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 Choose **Networking** in the navigation pane and click **Create Service** in the upper right corner.

Step 3 Set 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 to which the workload belongs.
- **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**. A Service selects a pod based on the added label. You can also click **Reference Workload Label** to reference the label of an existing workload. In the dialog box that is displayed, select a workload and click **OK**.
- **Port**
 - **Protocol:** protocol used by the Service.
 - **Service Port:** port used by the Service. The port number ranges from 1 to 65535.

- **Container Port:** port on which the workload listens. 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 run kubectl commands to set the access type. This section uses an 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 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

11.3.4 LoadBalancer

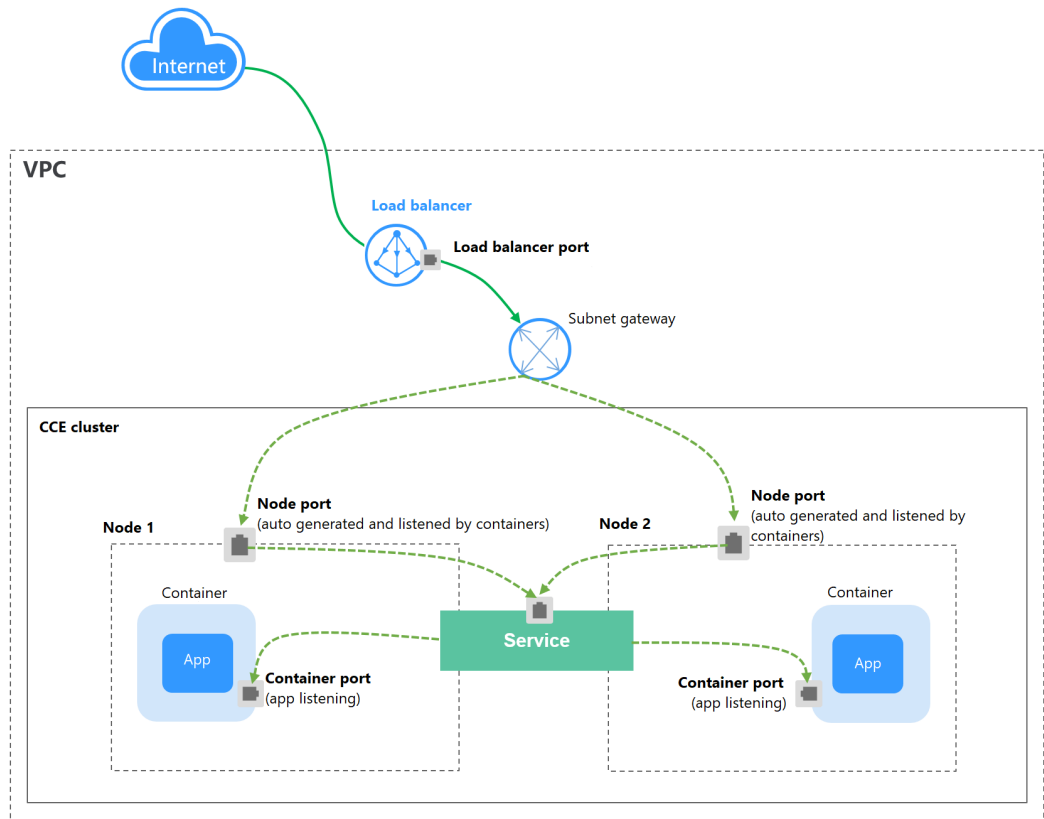
11.3.4.1 Creating a LoadBalancer Service

Scenario

LoadBalancer Services can access workloads from the public network through ELB, 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.

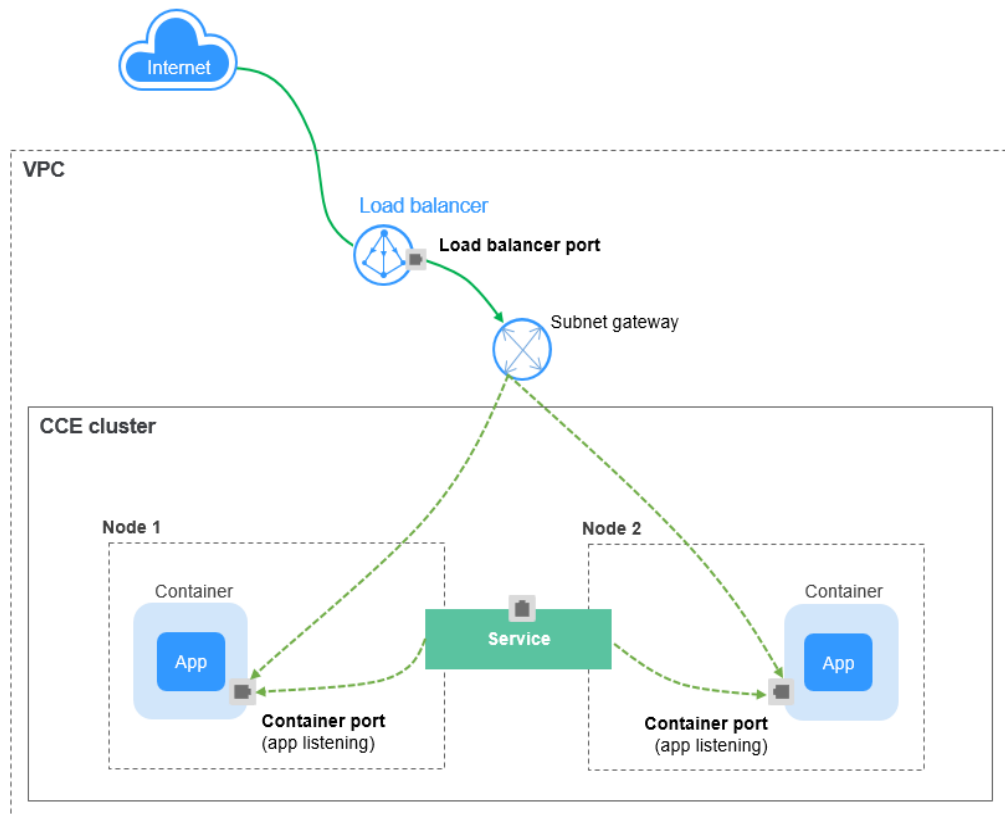
Figure 11-13 LoadBalancer



When **CCE Turbo clusters and dedicated load balancers** are used, passthrough networking is supported to reduce service latency and ensure zero performance loss.

External access requests are directly forwarded from a load balancer to pods. Internal access requests can be forwarded to a pod through a Service.

Figure 11-14 Passthrough networking



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. You are advised not to modify the Service affinity setting after the Service is created. 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 Service Fail to Be Accessed from Within the Cluster](#).
- CCE Turbo clusters support only cluster-level service affinity.

- Dedicated ELB load balancers can be used only in clusters of v1.17 and later.
- Dedicated load balancers must be of the network type (TCP/UDP) supporting private networks (with a private IP). If the Service needs to support HTTP, the specifications of dedicated load balancers must use HTTP/HTTPS (application load balancing) in addition to TCP/UDP (network load balancing).
- 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. In a CCE Turbo cluster, this constraint applies to shared load balancers, but not dedicated ones. Use dedicated load balancers in CCE Turbo clusters.
- 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. You are advised to use different ELB load balancers for the ingress and Service.

Creating a LoadBalancer Service

Step 1 Log in to the CCE console and click the cluster name to access the cluster console.

Step 2 Choose **Networking** in the navigation pane and click **Create Service** in the upper right corner.

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 to which the workload belongs.
- **Service Affinity:** For details, see [externalTrafficPolicy \(Service Affinity\)](#).
 - **Cluster level:** The IP addresses and access ports of all nodes in a cluster can be used to access the workload associated with the Service. Service access will cause performance loss due to route redirection, and the source IP address of the client cannot be obtained.
 - **Node level:** Only the IP address and access port of the node where the workload is located can 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**. A Service selects a pod based on the added label. You can also click **Reference Workload Label** to reference the label of an existing workload. In the dialog box that is displayed, select a workload and click **OK**.
- **Load Balancer**

Select the load balancer to interconnect. Only load balancers in the same VPC as the cluster are supported. If no load balancer is available, click **Create Load Balancer** to create one on the ELB console.

The CCE console supports automatic creation of load balancers. Select **Auto create** from the drop-down list box and set the following parameters:

- **Instance Name:** Enter a load balancer name.
- **Public Access:** If enabled, an EIP with 5 Mbit/s bandwidth will be created.
- **Subnet, AZ, and Specifications** (available only for dedicated load balancers): Configure the subnet AZ, and specifications. Currently, only dedicated load balancers of the network type (TCP/UDP) can be automatically created.

You can click **Edit** in the **Set ELB** area and configure load balancer parameters in the **Set ELB** dialog box.

- **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 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.
- **Type:** This function is disabled by default. You can select **Source IP address**. 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 address algorithm, 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 definition for custom health check, see [Configuring Health Check for Multiple Ports](#).

Table 11-2 Health check parameters

Parameter	Description
Protocol	When the protocol of Port is set to TCP, the TCP and HTTP are supported. When the protocol of Port is set to UDP, the UDP is supported. <ul style="list-style-type: none"> - Check Path (supported only by the HTTP): specifies the health check URL. The check path must start with a slash (/) and contain 1 to 80 characters.
Port	By default, the service port (Node Port and 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 cce-healthz will be added for the Service. <ul style="list-style-type: none"> - Node Port: 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. - Container Port: 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.
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.

- **Port**
 - **Protocol**: protocol used by the Service.
 - **Service Port**: port used by the Service. The port number ranges from 1 to 65535.
 - **Container Port**: port on which the workload listens. For example, Nginx uses port 80 by default.
 - **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 11-2](#).

 **NOTE**

When a LoadBalancer Service is created, a random node port number (NodePort) is automatically generated.

- **Annotation**: The LoadBalancer Service has some advanced CCE functions, which are implemented by annotations. For details, see [Using Annotations to Configure Load Balancing](#).

Step 4 Click **OK**.

----End

Using kubectl to Create a Service (Using an Existing Load Balancer)

You can set the 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 the files named `nginx-deployment.yaml` and `nginx-elb-svc.yaml` and edit them.

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 [Scheduling Policy \(Affinity/Anti-affinity\)](#).

```
apiVersion: v1
kind: Service
metadata:
  name: nginx
annotations:
  kubernetes.io/elb.id: <your_elb_id> # ELB ID. Replace it with the actual value.
  kubernetes.io/elb.class: union # 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 the ELB health check function.
  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 11-3](#).

In addition to the functions in this example, for more annotations and examples related to advanced functions, see [Using Annotations to Configure Load Balancing](#).

Table 11-3 annotations parameters

Parameter	Mandatory	Type	Description
kubernetes.io/elb.id	Yes	String	<p>ID of an enhanced load balancer. Mandatory when an existing load balancer is to be associated.</p> <p>How to obtain:</p> <p>On the management console, click Service List, and choose Networking > Elastic Load Balance. Click the name of the target load balancer. On the Summary tab page, find and copy the ID.</p> <p>NOTE</p> <p>The system preferentially connects to the load balancer based on the kubernetes.io/elb.id field. If this field is not specified, the spec.loadBalancerIP field is used (optional and available only in 1.23 and earlier versions).</p> <p>Do not use the spec.loadBalancerIP field to connect to the load balancer. This field will be discarded by Kubernetes. For details, see Deprecation.</p>
kubernetes.io/elb.class	Yes	String	<p>Select a proper load balancer type.</p> <ul style="list-style-type: none"> • union: shared load balancer • performance: dedicated load balancer, which can be used only in clusters of v1.17 and later. <p>NOTE</p> <p>If a LoadBalancer Service accesses an existing dedicated load balancer, the dedicated load balancer must support TCP/UDP networking.</p>

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 ROUND_ROBIN.</p> <p>Options:</p> <ul style="list-style-type: none"> ● ROUND_ROBIN: weighted round robin algorithm ● LEAST_CONNECTIONS: weighted least connections algorithm ● SOURCE_IP: source IP hash algorithm <p>NOTE If this parameter is set to SOURCE_IP, the weight setting (weight 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 is supported. That is, access requests from the same IP address are forwarded to the same backend server.</p> <ul style="list-style-type: none"> ● Disabling sticky session: Do not configure this parameter. ● Enabling sticky session: Set this parameter to SOURCE_IP, indicating that the sticky session is based on the source IP address. <p>NOTE When kubernetes.io/elb.lb-algorithm is set to SOURCE_IP (source IP address algorithm), sticky session cannot be enabled.</p>
kubernetes.io/elb.session-affinity-option	No	Table 11-4 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"> ● Enabling health check: Leave blank this parameter or set it to on. ● Disabling health check: Set this parameter to off. <p>If this parameter is enabled, the kubernetes.io/elb.health-check-option field must also be specified at the same time.</p>

Parameter	Mandatory	Type	Description
kubernetes.io/elb.health-check-option	No	Table 11-5 object	ELB health check configuration items.

Table 11-4 Data structure of the **elb.session-affinity-option** field

Parameter	Mandatory	Type	Description
persistenc e_timeout	Yes	String	Sticky session timeout, in minutes. This parameter is valid only when elb.session-affinity-mode is set to SOURCE_IP . Value range: 1 to 60. Default value: 60

Table 11-5 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: 5
timeout	No	String	Health check timeout, in seconds. Value range: 1 to 50. Default value: 10
max_retrie s	No	String	Maximum number of health check retries. Value range: 1 to 10. Default value: 3
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 HTTP. Default value: / The value can contain 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-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 access type has been set, 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 11-15 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. Commercial support is available at nginx.com.

Thank you for using nginx.

----End

Using kubectl to Create a Service (Automatically Creating a Load Balancer)

You can set the 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 the files named **nginx-deployment.yaml** and **nginx-elb-svc.yaml** and edit them.

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 [Scheduling Policy \(Affinity/Anti-affinity\)](#).

Example of a Service using a public network shared load balancer:

```

apiVersion: v1
kind: Service
metadata:
  annotations:
    kubernetes.io/elb.class: union
    kubernetes.io/elb.autocreate: '{
      "type": "public",
      "bandwidth_name": "cce-bandwidth-1551163379627",
      "bandwidth_chargemode": "bandwidth",
      "bandwidth_size": 5,
      "bandwidth_sharetype": "PER",
      "eip_type": "5_bgp"
    }'
    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 the ELB health check function.
    kubernetes.io/elb.health-check-option: '{
      "protocol": "TCP",
      "delay": "5",
      "timeout": "10",
      "max_retries": "3"
    }'
  labels:
    app: nginx
    name: nginx
spec:
  ports:
    - name: service0
      port: 80
      protocol: TCP
      targetPort: 80
  selector:
    app: nginx
  type: LoadBalancer

```

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",
      "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 the ELB health check function.
    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 11-6](#).

In addition to the functions in this example, for more annotations and examples related to advanced functions, see [Using Annotations to Configure Load Balancing](#).

Table 11-6 annotations parameters

Parameter	Mandatory	Type	Description
kubernetes.io/elb.class	Yes	String	Select a proper load balancer type. <ul style="list-style-type: none"> • union: shared load balancer • performance: dedicated load balancer, which can be used only in clusters of v1.17 and later.
kubernetes.io/elb.autocreate	Yes	elb.autocreate object	Whether to automatically create a load balancer associated with the Service. <p>Example</p> <ul style="list-style-type: none"> • If a public network load balancer will be automatically created, set this parameter to the following value: <pre>{"type":"public","bandwidth_name":"ce-bandwidth-1551163379627","bandwidth_chargemode":"bandwidth","bandwidth_size":5,"bandwidth_sharetype":"PER","eip_type":"5_bgp","name":"james"}</pre> • 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>
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"> • Mandatory when a cluster of v1.11.7-r0 or earlier is to be automatically created. • Optional for clusters later than v1.11.7-r0.

Parameter	Mandatory	Type	Description
kubernetes.io/elb.enterpriseID	No	String	<p>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.</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 0, resources will be bound to the default enterprise project.</p> <p>How to obtain:</p> <p>Log in to the EPS console. In the navigation pane, choose Project Management. In the enterprise project list, click the name of the enterprise project to be added. On the enterprise project details page, copy the project ID.</p>
kubernetes.io/elb.lb-algorithm	No	String	<p>Specifies the load balancing algorithm of the backend server group. The default value is ROUND_ROBIN.</p> <p>Options:</p> <ul style="list-style-type: none"> • ROUND_ROBIN: weighted round robin algorithm • LEAST_CONNECTIONS: weighted least connections algorithm • SOURCE_IP: source IP hash algorithm <p>NOTE</p> <p>If this parameter is set to SOURCE_IP, the weight setting (weight 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 is supported. That is, access requests from the same IP address are forwarded to the same backend server.</p> <ul style="list-style-type: none"> • Disabling sticky session: Do not configure this parameter. • Enabling sticky session: Set this parameter to SOURCE_IP, indicating that the sticky session is based on the source IP address. <p>NOTE</p> <p>When kubernetes.io/elb.lb-algorithm is set to SOURCE_IP (source IP address algorithm), sticky session cannot be enabled.</p>

Parameter	Mandatory	Type	Description
kubernetes.io/elb.session-affinity-option	No	Table 11-4 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"> Enabling health check: Leave blank this parameter or set it to on. Disabling health check: Set this parameter to off. If this parameter is enabled, the kubernetes.io/elb.health-check-option field must also be specified at the same time.
kubernetes.io/elb.health-check-option	No	Table 11-5 object	ELB health check configuration items.

Table 11-7 Data structure of the **elb.autocreate** field

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: cce-lb+service.UID
type	No	String	Network type of the load balancer. <ul style="list-style-type: none"> public: public network load balancer inner: private network load balancer Default: inner

Parameter	Mandatory	Type	Description
bandwidth_name	Yes for public network load balancers	String	Bandwidth name. The default value is cce-bandwidth-***** . The value can contain 1 to 64 characters. Only letters, digits, underscores (_), hyphens (-), and periods (.) are allowed.
bandwidth_chargemode	No	String	Bandwidth mode. <ul style="list-style-type: none"> bandwidth: billed by bandwidth traffic: billed by traffic Default: bandwidth
bandwidth_size	Yes for public network load balancers	Integer	Bandwidth size. The default value is 1 to 2000 Mbit/s. Configure this parameter based on the bandwidth range allowed in your region. The minimum increment for bandwidth adjustment varies depending on the bandwidth range. <ul style="list-style-type: none"> The minimum increment is 1 Mbit/s if the allowed bandwidth does not exceed 300 Mbit/s. The minimum increment is 50 Mbit/s if the allowed bandwidth ranges from 300 Mbit/s to 1000 Mbit/s. The minimum increment is 500 Mbit/s if the allowed bandwidth exceeds 1000 Mbit/s.
bandwidth_sharetype	Yes for public network load balancers	String	Bandwidth sharing mode. <ul style="list-style-type: none"> PER: dedicated bandwidth
eip_type	Yes for public network load balancers	String	EIP type. <ul style="list-style-type: none"> 5_bgp: dynamic BGP The specific type varies with regions. For details, see the EIP console.
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. The value of this parameter must be the same as that of l4_flavor_name , that is, both are elastic specifications or fixed specifications.
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: <pre>"elb_virsubnet_ids": ["14567f27-8ae4-42b8-ae47-9f847a4690dd"]</pre>

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 access type has been set, 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.XXX.XXX.XXX:80**. **10.XXX.XXX.XXX** indicates the IP address of the load balancer, and **80** indicates the access port displayed on the CCE console.

The Nginx is accessible.

Figure 11-16 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.
Commercial support is available at nginx.com.

Thank you for using nginx.

----End

11.3.4.2 Using Annotations to Configure Load Balancing

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 Protocol](#)
- [Dynamic Adjustment of the Weight of the Backend ECS](#)
- [Pass-through Capability](#)
- [Host Network](#)
- [Timeout](#)

Interconnection with ELB

Table 11-8 Annotations for interconnecting with ELB

Parameter	Type	Description	Supported Cluster Version
kubernetes.io/elb.class	String	Select a proper load balancer type. <ul style="list-style-type: none"> • union: shared load balancer • performance: dedicated load balancer, which can be used only in clusters of v1.17 and later. 	v1.9 or later
kubernetes.io/elb.id	String	Mandatory when an existing load balancer is to be associated . ID of a load balancer. How to obtain: On the management console, click Service List , and choose Networking > Elastic Load Balance . Click the name of the target load balancer. On the Summary tab page, find and copy the ID. NOTE The system preferentially connects to the load balancer based on the kubernetes.io/elb.id field. If this field is not specified, the spec.loadBalancerIP field is used (optional and available only in 1.23 and earlier versions). Do not use the spec.loadBalancerIP field to connect to the load balancer. This field will be discarded by Kubernetes. For details, see Deprecation .	v1.9 or later

Parameter	Type	Description	Supported Cluster Version
kubernetes.io/elb.autocreate	Table 11-16	<p>Mandatory when load balancers are automatically created.</p> <p>Example:</p> <ul style="list-style-type: none"> If a public network load balancer will be automatically created, set this parameter to the following value: <pre>{"type":"public","bandwidth_name":"cce-bandwidth-1551163379627","bandwidth_chargemode":"bandwidth","bandwidth_size":5,"bandwidth_sharetype":"PER","eip_type":"5_bgp","name":"james"}</pre> 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> 	v1.9 or later
kubernetes.io/elb.enterpriseID	String	<p>Optional when load balancers are automatically created.</p> <p>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.</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 0, resources will be bound to the default enterprise project.</p> <p>How to obtain:</p> <p>Log in to the EPS console. In the navigation pane, choose Project Management. In the enterprise project list, click the name of the enterprise project to be added. On the enterprise project details page, copy the project ID.</p>	v1.15 or later

Parameter	Type	Description	Supported Cluster Version
kubernetes.io/elb.subnet-id	String	<p>Optional when load balancers are automatically created.</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"> Mandatory when a cluster of v1.11.7-r0 or earlier is to be automatically created. Optional for clusters later than v1.11.7-r0. 	<p>Mandatory for versions earlier than v1.11.7-r0</p> <p>Discarded in versions 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 ROUND_ROBIN.</p> <p>Options:</p> <ul style="list-style-type: none"> ROUND_ROBIN: weighted round robin algorithm LEAST_CONNECTIONS: weighted least connections algorithm SOURCE_IP: source IP hash algorithm <p>NOTE If this parameter is set to SOURCE_IP, the weight setting (weight 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:

- Associating 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>           # ELB 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 creating a load balancer. For details, see [Using kubectl to Create a Service \(Automatically Creating a Load Balancer\)](#).

Shared load balancer:

```
apiVersion: v1
kind: Service
metadata:
  annotations:
    kubernetes.io/elb.class: union
    kubernetes.io/elb.autocreate: '{
      "type": "public",
      "bandwidth_name": "cce-bandwidth-1551163379627",
      "bandwidth_chargemode": "bandwidth",
      "bandwidth_size": 5,
      "bandwidth_sharetype": "PER",
      "eip_type": "5_bgp"
    }'
    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
  labels:
    app: nginx
    name: nginx
spec:
  ports:
    - name: service0
      port: 80
      protocol: TCP
      targetPort: 80
  selector:
    app: nginx
  type: LoadBalancer
```

Dedicated 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": "bandwidth",
      "bandwidth_size": 5,
      "bandwidth_sharetype": "PER",
      "eip_type": "5_bgp",
      "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
spec:
  selector:
    app: nginx
  ports:
    - name: cce-service-0
      targetPort: 80
```

```
nodePort: 0
port: 80
protocol: TCP
type: LoadBalancer
```

Sticky Session

Table 11-9 Annotations for sticky session

Parameter	Type	Description	Supported Cluster Version
kubernetes.io/elb.session-affinity-mode	String	<p>Source IP address-based sticky session is supported. That is, access requests from the same IP address are forwarded to the same backend server.</p> <ul style="list-style-type: none"> Disabling sticky session: Do not configure this parameter. Enabling sticky session: Set this parameter to SOURCE_IP, indicating that the sticky session is based on the source IP address. <p>NOTE When kubernetes.io/elb.lb-algorithm is set to SOURCE_IP (source IP address algorithm), sticky session cannot be enabled.</p>	v1.9 or later
kubernetes.io/elb.session-affinity-option	Table 11-19	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> # ELB ID. Replace it with the actual value.
    kubernetes.io/elb.class: union # 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 11-10 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"> Enabling health check: Leave blank this parameter or set it to on. Disabling health check: Set this parameter to off. If this parameter is enabled, the kubernetes.io/elb.health-check-option field must also be specified at the same time.	v1.9 or later
kubernetes.io/elb.health-check-option	Table 11-17	ELB health check configuration items.	v1.9 or later
kubernetes.io/elb.health-check-options	Table 11-18	ELB health check configuration item. Each Service port can be configured separately, and you can configure only some ports. NOTE kubernetes.io/elb.health-check-option and kubernetes.io/elb.health-check-options cannot be configured at the same time.	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>           # ELB ID. Replace it with the actual value.
    kubernetes.io/elb.class: union               # Load balancer type
    kubernetes.io/elb.health-check-flag: 'on'    # Enable the ELB health check function.
    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 for Multiple Ports](#).

HTTP Protocol

Table 11-11 Annotations for using HTTP protocols

Parameter	Type	Description	Supported Cluster Version
kubernetes.io/elb.protocol-port	String	Layer-7 forwarding configuration port used by the Service.	v1.19.16 or later
kubernetes.io/elb.cert-id	String	HTTP certificate used by the Service for Layer-7 forwarding.	v1.19.16 or later

For details about the application scenarios, see [Service Using HTTP](#).

Dynamic Adjustment of the Weight of the Backend ECS

Table 11-12 Annotations for dynamically adjusting the weight of the backend ECS

Parameter	Type	Description	Supported Cluster Version
kubernetes.io/elb.adaptive-weight	String	Dynamically adjusts the weight of the load balancer backend ECS based on pods. The requests received by each pod are more balanced. <ul style="list-style-type: none"> • true: enabled • false: disabled This parameter applies only to clusters of v1.21 or later and is invalid in passthrough networking.	v1.21 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> # ELB ID. Replace it with the actual value.
    kubernetes.io/elb.class: union # 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
```

Pass-through Capability

Table 11-13 Annotations for pass-through 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 about the application scenarios, see [Enabling Passthrough Networking for LoadBalancer Services](#).

Host Network

Table 11-14 Annotations for host network

Parameter	Type	Description	Supported Cluster Version
kubernetes.io/hws-hostNetwork	String	If the pod uses hostNetwork , the ELB forwards the request to the host network after this annotation is used. Options: <ul style="list-style-type: none"> true: enabled false (default): disabled 	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> # ELB ID. Replace it with the actual value.
    kubernetes.io/elb.class: union # 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 11-15 Annotation for configuring timeout

Parameter	Type	Description	Supported Cluster Version
kubernetes.io/elb.keepalive_timeout	Integer	<p>Timeout for client connections. If there are no requests reaching the load balancer after the timeout duration elapses, the load balancer will disconnect the connection with the client and establish a new connection when there is a new request.</p> <p>Value:</p> <ul style="list-style-type: none"> For TCP listeners, the value ranges from 10 to 4000 (in seconds). The default value is 300. For HTTP, HTTPS, and TERMINATED_HTTPS listeners, the value ranges from 10 to 4000 (in seconds). The default value is 60. For UDP listeners, this parameter does not take effect. 	v1.19.16-r30, v1.21.10-r10, v1.23.8-r10, v1.25.3-r10, and later

For details about the application scenarios, see [Configuring Timeout for a LoadBalancer Service](#).

Data Structure

Table 11-16 Data structure of the `elb.autocreate` field

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: cce-lb+service.UID
type	No	String	Network type of the load balancer. <ul style="list-style-type: none"> • public: public network load balancer • inner: private network load balancer Default: inner
bandwidth_name	Yes for public network load balancers	String	Bandwidth name. The default value is cce-bandwidth-***** . The value can contain 1 to 64 characters. Only letters, digits, underscores (_), hyphens (-), and periods (.) are allowed.
bandwidth_chargemode	No	String	Bandwidth mode. <ul style="list-style-type: none"> • bandwidth: billed by bandwidth • traffic: billed by traffic Default: bandwidth
bandwidth_size	Yes for public network load balancers	Integer	Bandwidth size. The default value is 1 to 2000 Mbit/s. Configure this parameter based on the bandwidth range allowed in your region. The minimum increment for bandwidth adjustment varies depending on the bandwidth range. <ul style="list-style-type: none"> • The minimum increment is 1 Mbit/s if the allowed bandwidth does not exceed 300 Mbit/s. • The minimum increment is 50 Mbit/s if the allowed bandwidth ranges from 300 Mbit/s to 1000 Mbit/s. • The minimum increment is 500 Mbit/s if the allowed bandwidth exceeds 1000 Mbit/s.

Parameter	Mandatory	Type	Description
bandwidth_sharetype	Yes for public network load balancers	String	Bandwidth sharing mode. <ul style="list-style-type: none"> • PER: dedicated bandwidth
eip_type	Yes for public network load balancers	String	EIP type. <ul style="list-style-type: none"> • 5_bgp: dynamic BGP The specific type varies with regions. For details, see the EIP console.
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.
l7_flavor_name	No	String	Flavor name of the layer-7 load balancer. This parameter is available only for dedicated load balancers. The value of this parameter must be the same as that of l4_flavor_name , that is, both are elastic specifications or fixed specifications.
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. <p>Example:</p> <pre>"elb_virsubnet_ids": ["14567f27-8ae4-42b8-ae47-9f847a4690dd"]</pre>

Table 11-17 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: 5
timeout	No	String	Health check timeout, in seconds. Value range: 1 to 50. Default value: 10
max_retries	No	String	Maximum number of health check retries. Value range: 1 to 10. Default value: 3
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 HTTP. Default value: / The value can contain 1 to 10,000 characters.

Table 11-18 Data structure description of the **elb.health-check-options** field

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. NOTE 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	Initial waiting time (in seconds) for starting the health check. Value range: 1 to 50. Default value: 5
timeout	No	String	Health check timeout, in seconds. Value range: 1 to 50. Default value: 10

Parameter	Mandatory	Type	Description
max_retries	No	String	Maximum number of health check retries. Value range: 1 to 10. Default value: 3
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 HTTP . Default value: / The value can contain 1 to 10,000 characters.

Table 11-19 Data structure of the **elb.session-affinity-option** field

Parameter	Mandatory	Type	Description
persistenc e_timeout	Yes	String	Sticky session timeout, in minutes. This parameter is valid only when elb.session-affinity-mode is set to SOURCE_IP . Value range: 1 to 60. Default value: 60

11.3.4.3 Service Using HTTP

Constraints

- Only clusters of v1.19.16 or later support HTTP.
- Do not connect the ingress and Service that uses HTTP to the same listener of the same load balancer. Otherwise, a port conflict occurs.
- Layer-7 routing of ELB can be enabled for Services. Both shared and dedicated ELB load balancers can be interconnected.

Restrictions on dedicated ELB load balancers are as follows:

- To interconnect with an existing dedicated load balancer, the load balancer flavor **must support both the layer-4 and layer-7 routing**. Otherwise, the load balancer will not work as expected.
- If you use an automatically created load balancer, you cannot use the CCE console to automatically create a layer-7 dedicated load balancer. Instead, you can use YAML to create a layer-7 dedicated load balancer, use both the layer-4 and layer-7 capabilities of the exclusive ELB instance (that is, specify the layer-4 and layer-7 flavors in the annotation of `kubernetes.io/elb.autocreate`).

Service Using HTTP

The following annotations need to be added:

- **kubernetes.io/elb.protocol-port**: "https:443,http:80"
The value of **protocol-port** must be the same as the port in the **spec.ports** field of the Service. The format is *Protocol:Port*. The port matches the one in the **service.spec.ports** field and is released as the corresponding protocol.
- **kubernetes.io/elb.cert-id**: "17e3b4f4bc40471c86741dc3aa211379"
cert-id indicates the certificate ID in ELB certificate management. When **https** is configured for **protocol-port**, the certificate of the ELB listener will be set to the **cert-id** certificate. When multiple HTTPS services are released, the same certificate is used.

The following is a configuration example. The two ports in **spec.ports** correspond to those in **kubernetes.io/elb.protocol-port**. Ports 443 and 80 are enabled for HTTPS and HTTP requests, respectively.

```
apiVersion: v1
kind: Service
metadata:
  annotations:
    # When an ELB load balancer is automatically created, both layer-4 and layer-7 flavors need to be
    # specified.
    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"
    }
    kubernetes.io/elb.class: performance
    kubernetes.io/elb.protocol-port: "https:443,http:80"
    kubernetes.io/elb.cert-id: "17e3b4f4bc40471c86741dc3aa211379"
  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
```

Use the preceding example configurations to create a Service. In the new ELB load balancer, you can see that the listeners on ports 443 and 80 are created.

11.3.4.4 Configuring Health Check for Multiple Ports

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.

Constraints

- This feature takes effect only 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
- **kubernetes.io/elb.health-check-option** and **kubernetes.io/elb.health-check-options** cannot be configured at the same time.
- 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> # ELB 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"
  }
]'
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 11-20 Data structure description of the **elb.health-check-options** field

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. NOTE 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	Initial waiting time (in seconds) for starting the health check. Value range: 1 to 50. Default value: 5
timeout	No	String	Health check timeout, in seconds. Value range: 1 to 50. Default value: 10
max_retries	No	String	Maximum number of health check retries. Value range: 1 to 10. Default value: 3
protocol	No	String	Health check protocol. Default value: protocol of the associated Service Value options: TCP, UDP, or HTTP

Parameter	Mandatory	Type	Description
path	No	String	Health check URL. This parameter needs to be configured when the protocol is HTTP . Default value: / The value can contain 1 to 10,000 characters.

11.3.4.5 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 there are no requests reaching the load balancer after the timeout duration elapses, the load balancer will disconnect the connection with the client and establish a new connection when there is a new request.

Constraints

- This feature takes effect only in the following versions:
 - v1.19: v1.19.16-r30 or later
 - v1.21: v1.21.10-r10 or later
 - v1.23: v1.23.8-r10 or later
 - v1.25: v1.25.3-r10 or later
- The timeout can be configured only for the LoadBalancer Services using dedicated load balancers.
- If you delete the timeout configuration during Service update, the timeout configuration on the existing listeners will be retained.

Procedure

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.
    kubernetes.io/elb.class: performance # Load balancer type
    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 11-21 Key annotation parameters

Parameter	Mandatory	Type	Description
kubernetes.io/elb.keepalive_timeout	No	Integer	<p>Timeout for client connections. If there are no requests reaching the load balancer after the timeout duration elapses, the load balancer will disconnect the connection with the client and establish a new connection when there is a new request.</p> <p>Value:</p> <ul style="list-style-type: none"> For TCP listeners, the value ranges from 10 to 4000. The default value is 300. For HTTP, HTTPS, and TERMINATED_HTTPS listeners, the value ranges from 10 to 4000. The default value is 60. For UDP listeners, the value ranges from 10 to 4000. The default value is 300.

11.3.4.6 Enabling Passthrough Networking for LoadBalancer Services

Challenges

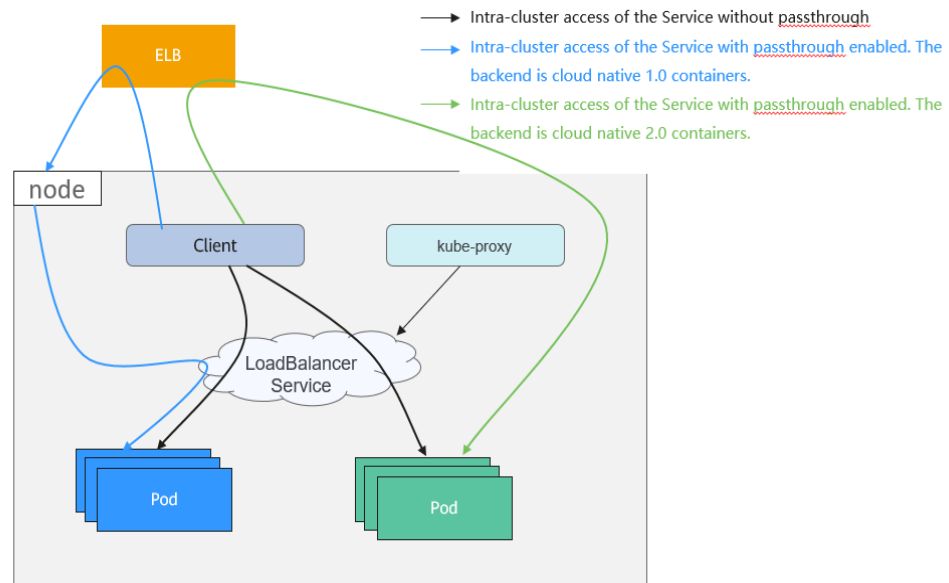
A Kubernetes cluster can publish applications running on a group of pods as Services, which provide unified layer-4 access entries. For a Loadbalancer Service, kube-proxy configures the LoadbalancerIP in **status** of the Service to the local forwarding rule of the node by default. When a pod accesses the load balancer from within the cluster, the traffic is forwarded within the cluster instead of being forwarded by the load balancer.

kube-proxy is responsible for intra-cluster forwarding. kube-proxy has two forwarding modes: iptables and IPVS. iptables is a simple polling forwarding mode. IPVS has multiple forwarding modes but it requires modifying the startup parameters of kube-proxy. Compared with iptables and IPVS, load balancers provide more flexible forwarding policies as well as health check capabilities.

Solution

CCE supports passthrough networking. You can configure the **annotation** of **kubernetes.io/elb.pass-through** for the Loadbalancer Service. Intra-cluster access to the Service load balancer address is then forwarded to backend pods by the load balancer.

Figure 11-17 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

- After passthrough networking is configured for a dedicated load balancer, containers on the node where the workload runs cannot be accessed through the Service.
- Passthrough networking is not supported for clusters of v1.15 or earlier.
- In IPVS network mode, the pass-through settings of Service connected to the same ELB must be the same.

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 Nginx image to create a Deployment.

```
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
```

Step 2 Create a LoadBalancer Service and configure `kubernetes.io/elb.pass-through` to `true`.

```
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_ty
pe": "5_bgp", "name": "james"}'
  labels:
    app: nginx
    name: nginx
spec:
  externalTrafficPolicy: Local
  ports:
  - name: service0
    port: 80
    protocol: TCP
    targetPort: 80
  selector:
    app: nginx
  type: LoadBalancer
```

A shared load balancer named **james** is automatically created. Use `kubernetes.io/elb.subnet-id` to specify the VPC subnet where the load balancer is located. The load balancer and the cluster must be in the same VPC.

----End

Verification

Check the ELB load balancer corresponding to the created Service. The load balancer name is **james**. The number of ELB connections is **0**.

Use `kubectl` to connect to the cluster, go to an Nginx container, and access the ELB address. The access is successful.

```
# kubectl get pod
NAME                READY STATUS RESTARTS AGE
nginx-7c4c5cc6b5-vpncx 1/1   Running 0      9m47s
nginx-7c4c5cc6b5-xj5wl 1/1   Running 0      9m47s
# kubectl exec -it nginx-7c4c5cc6b5-vpncx -- /bin/sh
# curl 120.46.141.192
<!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>
```

Wait for a period of time and view the ELB monitoring data. A new access connection is created for the ELB, indicating that the access passes through the ELB load balancer as expected.

11.3.4.7 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 ECS console, find the ECS corresponding to any node where the workload runs, and click the ECS name. On the displayed ECS details page, record the security group name.
- Step 2** Log in to the VPC console. In the navigation pane on the left, choose **Access Control > Security Groups**. In the security group list on the right, click the security group name obtained in step 1.
- 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**.

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.

----End

11.3.5 Headless Service

The preceding types of 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
```

11.4 Ingresses

11.4.1 Overview

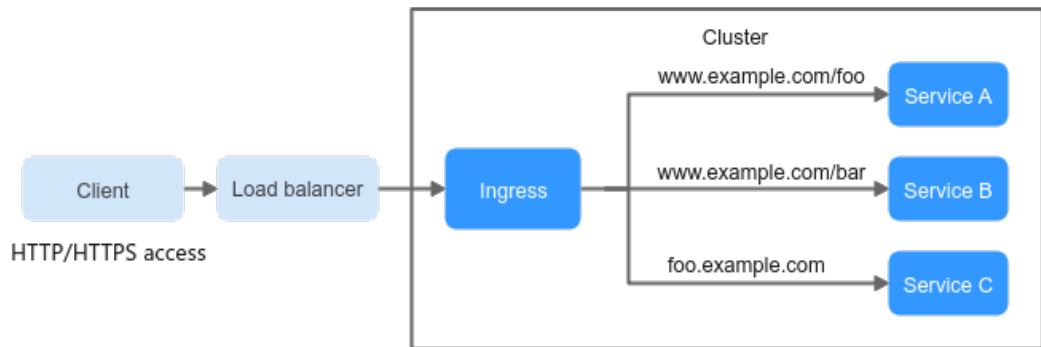
Why We Need Ingresses

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 11-18**, you can customize forwarding rules based on domain names and URLs to implement fine-grained distribution of access traffic.

Figure 11-18 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.

Ingress Controllers provided by different vendors are implemented in different ways. Based on the types of load balancers, Ingress Controllers are classified into ELB Ingress Controller and Nginx Ingress Controller. Both of them are supported in CCE. ELB Ingress Controller forwards traffic through ELB. Nginx Ingress Controller uses the templates and images maintained by the Kubernetes community to forward traffic through the Nginx component.

Ingress Feature Comparison

Table 11-22 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"> Processes must be reloaded for non-backend endpoint changes, which causes loss to persistent connections. Lua supports hot updates of endpoint changes. Processes must be reloaded for a Lua modification.
Component deployment	Deployed on the master node	Deployed on worker nodes, and operations costs required for the Nginx component
Route redirection	Not 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 ELB ingress is essentially different from the open source Nginx ingress. Therefore, their supported Service types are different. For details, see [Services Supported by Ingresses](#).

ELB Ingress Controller is deployed on a master node. All policies and forwarding behaviors are configured on the ELB side. Load balancers outside the cluster can connect to nodes in the cluster only through the IP address of the VPC. Therefore, ELB Ingress supports 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, ELB Ingress uses 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.

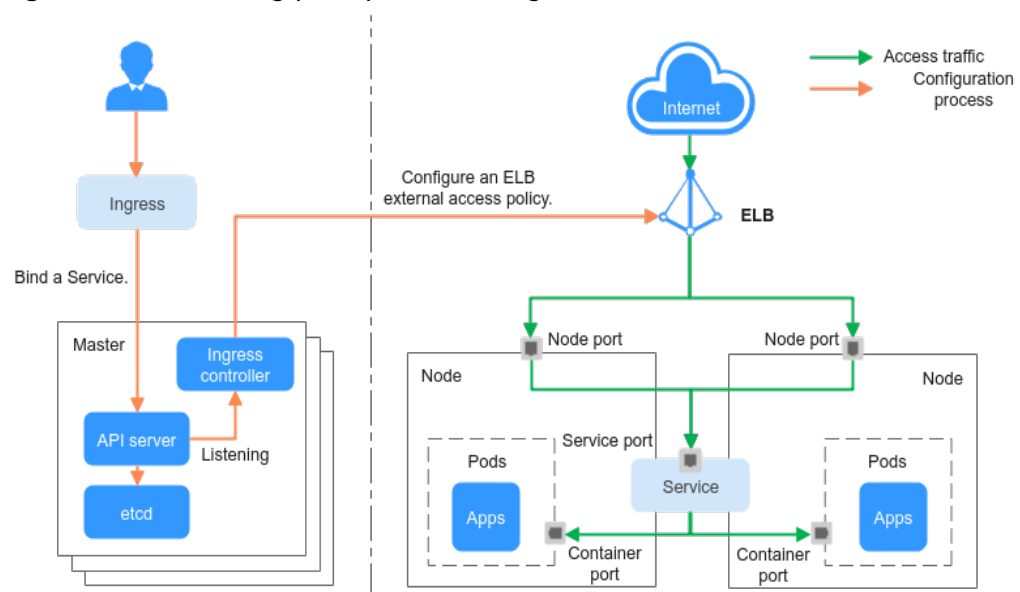
Working Principle of ELB Ingress Controller

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 11-19** 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 11-19 Working principle of ELB Ingress Controller



Working Principle of Nginx Ingress Controller

An Nginx ingress uses ELB as the traffic ingress. The **nginx-ingress** add-on is deployed in a cluster to balance traffic and control access.

NOTE

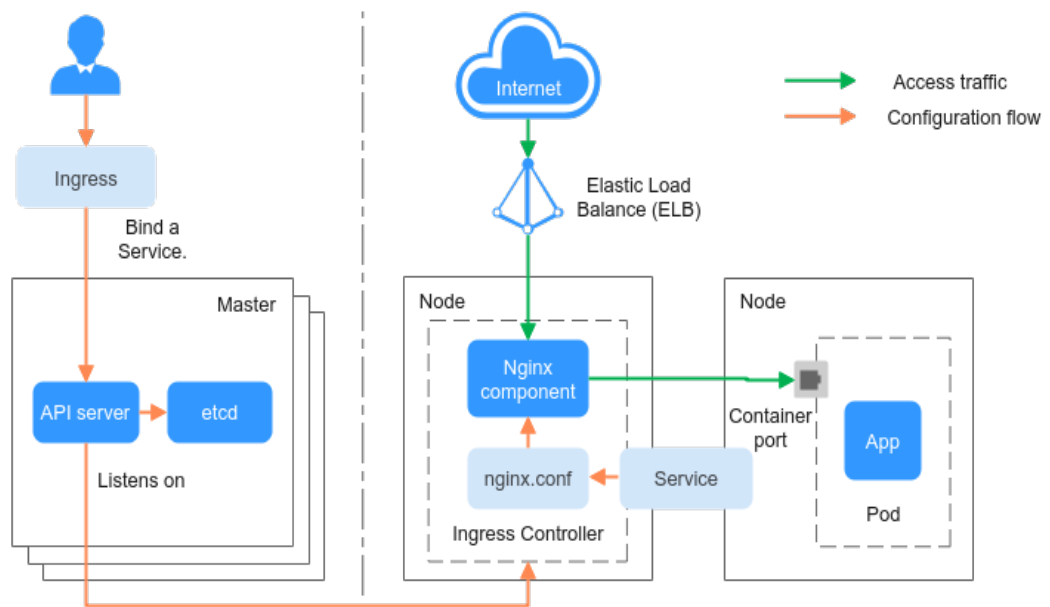
The nginx-ingress add-on in CCE is implemented using the open-source community chart and image. CCE does not maintain the add-on. Therefore, it is not recommended that the nginx-ingress add-on be used commercially.

You can visit the [open source community](#) for more information.

Nginx Ingress Controller is deployed on worker nodes through pods, which will result in O&M costs and Nginx component running overheads. **Figure 11-20** shows the working principles of Nginx Ingress Controller.

1. After you update ingress resources, Nginx Ingress Controller writes a forwarding rule defined in the ingress resources into the **nginx.conf** configuration file of Nginx.
2. The built-in Nginx component reloads the updated configuration file to modify and update the Nginx forwarding rule.
3. When traffic accesses a cluster, the traffic is first forwarded by the created load balancer to the Nginx component in the cluster. Then, the Nginx component forwards the traffic to each workload based on the forwarding rule.

Figure 11-20 Working principle of Nginx Ingress Controller



Services Supported by Ingresses

Table 11-23 lists the services supported by ELB Ingresses.

Table 11-23 Services supported by ELB Ingresses

Cluster Type	ELB Type	ClusterIP	NodePort
CCE 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

Cluster Type	ELB Type	ClusterIP	NodePort
	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.)

Table 11-24 lists the services supported by Nginx Ingresses.

Table 11-24 Services supported by Nginx Ingresses

Cluster Type	ELB Type	ClusterIP	NodePort
CCE cluster	Shared load balancer	Supported	Supported
	Dedicated load balancer	Supported	Supported
CCE Turbo cluster	Shared load balancer	Supported	Supported
	Dedicated load balancer	Supported	Supported

11.4.2 ELB Ingresses

11.4.2.1 Creating an ELB 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](#).
- [Services Supported by Ingresses](#) lists the Service types supported by ELB ingresses.

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 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. You are advised to use different ELB load balancers for the ingress and Service.
- Dedicated load balancers must be the application type (HTTP/HTTPS) supporting private networks (with a private IP).
- If multiple ingresses are used to connect to the same ELB port in the same cluster, the listener configuration items (such as the certificate associated with the listener and the HTTP2 attribute of the listener) are subject to the configuration of the first ingress.

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 and click the cluster name to access the cluster console.

Step 2 Choose **Networking** in the navigation pane, click the **Ingresses** tab, and click **Create Ingress** in the upper right corner.

Step 3 Configure ingress parameters.

- **Name:** specifies a name of an ingress, for example, **ingress-demo**.
- **Interconnect with Nginx:** This option is displayed only after the **NGINX Ingress Controller** add-on is installed. If this option is available, the nginx-ingress add-on has been installed. Enabling this option will create an Nginx ingress. Disable it if you want to create an ELB ingress. For details, see [Creating Nginx Ingresses on the Console](#).
- **Load Balancer**

Select the load balancer to interconnect. Only load balancers in the same VPC as the cluster are supported. If no load balancer is available, click **Create Load Balancer** to create one on the ELB console.

Dedicated load balancers must support HTTP or HTTPS and the network type must support private networks.

The CCE console supports automatic creation of load balancers. Select **Auto create** from the drop-down list box and configure the following parameters:

 - **Instance Name:** Enter a load balancer name.
 - **Public Access:** If enabled, an EIP with 5 Mbit/s bandwidth will be created.
 - **Subnet, AZ, and Specifications** (available only for dedicated load balancers): Configure the subnet AZ, and specifications. Only HTTP- or HTTPS-compliant dedicated load balancers can be automatically created.
- **Listener:** 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 can be specified randomly.

- **Certificate Source:** TLS secret and ELB server certificate are supported.
- **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.
 - **TLS secret:** For details about how to create a secret certificate, see [Creating a Secret](#).
 - **ELB server certificate:** Use the certificate created in the ELB service.

 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:** Server Name Indication (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. 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 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.XXX.XXX.XXX:80/helloworld*), the request is forwarded to the corresponding Service for processing. You can click  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.

- URL 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.
 - **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`.

 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 that your Nginx application contains the same URL, that is, `/usr/share/nginx/html/test`, otherwise, 404 is returned.

- **Destination Service:** Select an existing Service or create a Service. Services that do not meet search criteria are automatically filtered out.
- **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 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. Options are as follows:
 - **Load balancer cookie:** Enter the **Stickiness Duration** , which ranges from 1 to 1,440 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.

- **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 set to TCP, TCP and HTTP are supported. When it is set to UDP, only UDP is supported. <ul style="list-style-type: none"> ○ Check Path (supported only by the HTTP health check protocol): specifies the health check URL. The check path must start with a slash (/) and contain 1 to 80 characters.
Port	By default, the service port (Node Port and 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 cce-healthz will be added for the Service. <ul style="list-style-type: none"> ○ Node Port: 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. ○ Container Port: 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.
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 [Creating an Ingress - Automatically Creating a Load Balancer](#) and [Creating an Ingress - Interconnecting with an Existing Load Balancer](#).

Step 4 After the configuration is complete, click **OK**. 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.

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 11-21](#).

Figure 11-21 Accessing the `/healthz` interface of defaultbackend



----End

11.4.2.2 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

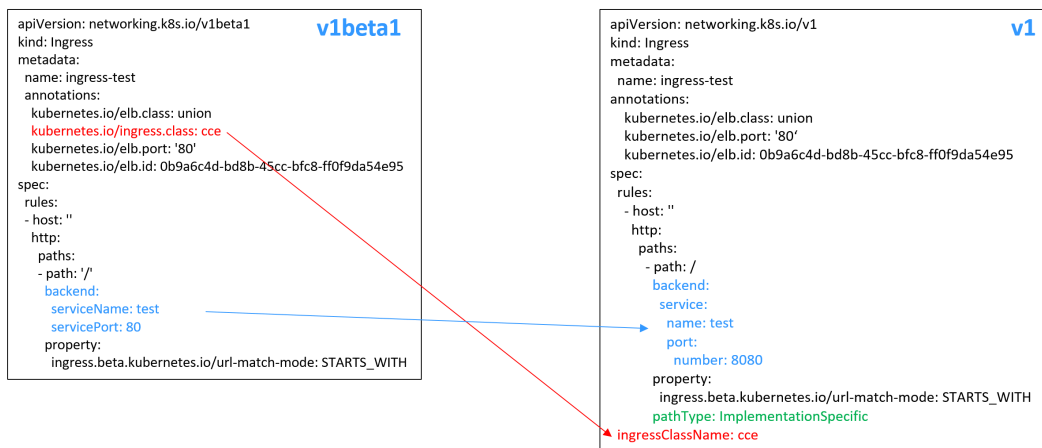
- An ingress provides network access for backend workloads. Ensure that a workload is available in a cluster. If no workload is available, deploy a sample Nginx workload by referring to [Creating a Deployment](#), [Creating a StatefulSet](#), or [Creating a DaemonSet](#).
- [Services Supported by Ingresses](#) lists the Service types supported by ELB ingresses.
- Dedicated load balancers must be the application type (HTTP/HTTPS) supporting private networks (with a private IP).

Ingress Description of networking.k8s.io/v1

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 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, 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 Description of networking.k8s.io/v1](#).

Example of a shared load balancer (public network access) for clusters of v1.23 or later:

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: ingress-test
  annotations:
    kubernetes.io/elb.class: union
    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"
      }'
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 # ELB ingress is used.
```

Example of a shared load balancer (public network access) for clusters of v1.21 or earlier:

```
apiVersion: networking.k8s.io/v1beta1
kind: Ingress
metadata:
  name: ingress-test
  annotations:
    kubernetes.io/elb.class: union
    kubernetes.io/ingress.class: cce # ELB ingress is used.
    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"
    }
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

```

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",
        "available_zone": [
          "eu-west-0a"
        ],
        "elb_virsubnet_ids": ["b4bf8152-6c36-4c3b-9f74-2229f8e640c9"],
        "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 version 1.21 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",
  "available_zone": [
    "eu-west-0a"
  ],
  "elb_virsubnet_ids": ["b4bf8152-6c36-4c3b-9f74-2229f8e640c9"],
  "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 11-25 Key parameters

Parameter	Mandatory	Type	Description
kubernetes.io/elb.class	Yes	String	Select a proper load balancer type. <ul style="list-style-type: none"> • union: shared load balancer • performance: dedicated load balancer, which can be used only in clusters of v1.17 and later.
kubernetes.io/ingress.class	Yes (only for clusters of v1.21 or earlier)	String	cce : 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	cce : The self-developed ELB ingress is used. This parameter is 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 NOTE Some ports are high-risk ports and are blocked by default, for example, port 21.

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"> • Mandatory when a cluster of v1.11.7-r0 or earlier is to be automatically created. • Optional for clusters later than v1.11.7-r0. It is left blank by default.
kubernetes.io/elb.enterpriseID	No	String	<p>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.</p> <p>ID of the enterprise project in which the load balancer will be created. The value contains 1 to 100 characters.</p> <p>How to obtain:</p> <p>Log in to the EPS console. In the navigation pane, choose Project Management. In the enterprise project list, click the name of the enterprise project to be added. On the enterprise project details page, copy the project ID.</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 Table 11-26.</p> <p>Example</p> <ul style="list-style-type: none"> • 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": "bandwidth", "bandwidth_size": 5, "bandwidth_sharetype": "PER", "eip_type": "5_bgp", "name": "james" }</pre> • 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>

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. 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.
path	Yes	String	User-defined route path. All external access requests must match host and path . 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.
ingress.beta.kubernetes.io/url-match-mode	No	String	Route matching policy. Default: STARTS_WITH (prefix match) Options: <ul style="list-style-type: none"> • EQUAL_TO: exact match • STARTS_WITH: prefix match • REGEX: regular expression match

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"> • ImplementationSpecific: The matching method depends on Ingress Controller. The matching method defined by ingress.beta.kubernetes.io/url-match-mode is used in CCE. • Exact: exact matching of the URL, which is case-sensitive. • Prefix: 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. <p>NOTE</p> <ul style="list-style-type: none"> - 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, /foo/bar matches /foo/bar/baz but does not match /foo/barbaz. - 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, /foo/bar matches /foo/bar/. <p>See examples of ingress path matching.</p>

Table 11-26 Data structure of the **elb.autocreate** field

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: cce-lb+service.UID</p>

Parameter	Mandatory	Type	Description
type	No	String	Network type of the load balancer. <ul style="list-style-type: none"> • public: public network load balancer • inner: private network load balancer Default: inner
bandwidth_name	Yes for public network load balancers	String	Bandwidth name. The default value is cce-bandwidth-***** . The value can contain 1 to 64 characters. Only letters, digits, underscores (_), hyphens (-), and periods (.) are allowed.
bandwidth_chargemode	No	String	Bandwidth mode. <ul style="list-style-type: none"> • bandwidth: billed by bandwidth • traffic: billed by traffic Default: bandwidth
bandwidth_size	Yes for public network load balancers	Integer	Bandwidth size. The default value is 1 to 2000 Mbit/s. Configure this parameter based on the bandwidth range allowed in your region. The minimum increment for bandwidth adjustment varies depending on the bandwidth range. <ul style="list-style-type: none"> • The minimum increment is 1 Mbit/s if the allowed bandwidth does not exceed 300 Mbit/s. • The minimum increment is 50 Mbit/s if the allowed bandwidth ranges from 300 Mbit/s to 1000 Mbit/s. • The minimum increment is 500 Mbit/s if the allowed bandwidth exceeds 1000 Mbit/s.
bandwidth_sharetype	Yes for public network load balancers	String	Bandwidth sharing mode. <ul style="list-style-type: none"> • PER: dedicated bandwidth
eip_type	Yes for public network load balancers	String	EIP type. <ul style="list-style-type: none"> • 5_bgp: dynamic BGP The specific type varies with regions. For details, see the EIP console.

Parameter	Mandatory	Type	Description
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.
l7_flavor_name	No	String	Flavor name of the layer-7 load balancer. This parameter is available only for dedicated load balancers. The value of this parameter must be the same as that of l4_flavor_name , that is, both are elastic specifications or fixed specifications.
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 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 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

- Existing dedicated load balancers must be the application type (HTTP/HTTPS) supporting private networks (with a private IP).

If the cluster version is 1.23 or later, the YAML file configuration 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.class: performance # Load balancer type
    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 YAML file configuration 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 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 11-27 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. How to obtain: On the management console, click Service List , and choose Networking > Elastic Load Balance . Click the name of the target load balancer. On the Summary 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.
kubernetes.io/elb.class	Yes	String	Load balancer type. <ul style="list-style-type: none"> • union: shared load balancer • performance: dedicated load balancer, which can be used only in clusters of v1.17 and later. NOTE If an ELB Ingress accesses an existing dedicated load balancer, the dedicated load balancer must be of the application load balancing (HTTP/HTTPS) type.

11.4.2.3 Configuring ELB Ingresses Using Annotations

By adding annotations to a YAML file, you can implement more advanced ingress functions. This section describes the annotations that can be used when you create an ingress of the ELB type.

- [Interconnecting with ELB](#)
- [Using HTTP/2](#)
- [Interconnecting with HTTPS Backend Services](#)
- [Configuring Timeout for an Ingress](#)

Interconnecting with ELB

Table 11-28 Annotations for interconnecting with ELB

Parameter	Type	Description	Supported Cluster Version
kubernetes.io/elb.class	String	Select a proper load balancer type. The value can be: <ul style="list-style-type: none"> • union: shared load balancer • performance: dedicated load balancer, which can be used only in clusters of v1.17 and later. 	v1.9 or later
kubernetes.io/ingress.class	String	<ul style="list-style-type: none"> • cce: The self-developed ELB ingress is used. • nginx: Nginx ingress is used. This parameter is mandatory when an ingress is created by calling the API. For clusters of v1.23 or later, use the parameter ingressClassName . For details, see Using kubectl to Create an ELB Ingress .	Only clusters of v1.21 or earlier
kubernetes.io/elb.port	Integer	This parameter indicates the external port registered with the address of the LoadBalancer Service. Supported range: 1 to 65535 NOTE 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 when an existing load balancer is to be interconnected . ID of a load balancer. How to obtain: On the management console, click Service List , and choose Networking > Elastic Load Balance . Click the name of the target load balancer. On the Summary tab page, find and copy the ID.	v1.9 or later
kubernetes.io/elb.ip	String	Mandatory when an existing load balancer is to be interconnected . 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.	v1.9 or later

Parameter	Type	Description	Supported Cluster Version
kubernetes.io/elb.autocreate	Table 11-32 Object	<p>Mandatory when load balancers are automatically created.</p> <p>Example</p> <ul style="list-style-type: none"> If a public network load balancer will be automatically created, set this parameter to the following value: <pre>{"type":"public","bandwidth_name":"cce-bandwidth-1551163379627","bandwidth_chargemode":"bandwidth","bandwidth_size":5,"bandwidth_sharetype":"PER","eip_type":"5_bgp","name":"james"}</pre> 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> 	v1.9 or later
kubernetes.io/elb.enterpriseID	String	<p>Optional when load balancers are automatically created.</p> <p>Clusters of v1.15 and later support this field. In clusters earlier than v1.15, load balancers are created in the default project by default.</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 0, resources will be bound to the default enterprise project.</p> <p>How to obtain:</p> <p>Log in to the EPS console. In the navigation pane, choose Project Management. In the enterprise project list, click the name of the enterprise project to be added. On the enterprise project details page, copy the project ID.</p>	v1.15 or later

Parameter	Type	Description	Supported Cluster Version
kubernetes.io/elb.subnet-id	String	<p>Optional when load balancers are automatically created.</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"> • Mandatory when a cluster of v1.11.7-r0 or earlier is to be automatically created. • Optional for clusters later than v1.11.7-r0. 	<p>Mandatory for clusters earlier than v1.11.7-r0</p> <p>Discarded in clusters later than v1.11.7-r0</p>

To use the preceding annotations, perform the following steps:

- See [Creating an Ingress - Interconnecting with an Existing Load Balancer](#) to interconnect an existing load balancer.
- See [Creating an Ingress - Automatically Creating a Load Balancer](#) to automatically create a load balancer.

Using HTTP/2

Table 11-29 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. This parameter is supported in clusters of v1.19.16-r0, v1.21.3-r0, and later.</p> <p>Options:</p> <ul style="list-style-type: none"> • true: enabled • false: disabled (default value) <p>Note: HTTP/2 can be enabled or disabled only when the listener uses HTTPS. This parameter is invalid and defaults to false when the listener protocol is HTTP.</p>	v1.19.16-r0, v1.21.3-r0, or later

For details about the application scenarios, see [ELB Ingresses Using HTTP/2](#).

Interconnecting with HTTPS Backend Services

Table 11-30 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 https .	v1.23.8, v1.25.3, or later

For details about the application scenarios, see [Interconnecting ELB Ingresses with HTTPS Backend Services](#).

Configuring Timeout for an Ingress

Table 11-31 Annotations of configuring ingress redirection rules

Parameter	Type	Description	Supported Cluster Version
kubernetes.io/elb.keepalive_timeout	Integer	<p>Timeout for client connections. If there are no requests reaching the load balancer after the timeout duration elapses, the load balancer will disconnect the connection with the client and establish a new connection when there is a new request.</p> <p>Value:</p> <ul style="list-style-type: none"> For TCP listeners, the value ranges from 10 to 4000 (in seconds). The default value is 300. For HTTP or HTTPS listeners, the value ranges from 0 to 4000 (in seconds). The default value is 60. <p>For UDP listeners, this parameter does not take effect.</p>	v1.19.16-r30, v1.21.10-r10, v1.23.8-r10, v1.25.3-r10, and later
kubernetes.io/elb.client_timeout	Integer	<p>Timeout duration for waiting for a response from a client. There are two situations:</p> <ul style="list-style-type: none"> If the client fails to send a request header to the load balancer within 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. <p>The value ranges from 1 to 300 (in seconds). The default value is 60.</p> <p>This parameter is available only for HTTP and HTTPS listeners.</p> <p>Minimum value: 1 Maximum value: 300 Default value: 60</p>	v1.19.16-r30, v1.21.10-r10, v1.23.8-r10, v1.25.3-r10, and later

Parameter	Type	Description	Supported Cluster Version
kubernetes.io/elb.member_timeout	Integer	<p>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 within the duration specified by member_timeout, the load balancer will stop waiting and return HTTP 504 Gateway Timeout.</p> <p>The value ranges from 1 to 300 (in seconds). The default value is 60.</p> <p>This parameter is available only for HTTP and HTTPS listeners.</p> <p>Minimum value: 1</p> <p>Maximum value: 300</p> <p>Default value: 60</p>	v1.19.16-r30, v1.21.10-r10, v1.23.8-r10, v1.25.3-r10, and later

For details about the application scenarios, see [Configuring Timeout for an ELB Ingress](#).

Data Structure

Table 11-32 Data structure of the **elb.autocreate** field

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: cce-lb+service.UID</p>
type	No	String	<p>Network type of the load balancer.</p> <ul style="list-style-type: none"> public: public network load balancer inner: private network load balancer <p>Default: inner</p>

Parameter	Mandatory	Type	Description
bandwidth_name	Yes for public network load balancers	String	Bandwidth name. The default value is cce-bandwidth-***** . The value can contain 1 to 64 characters. Only letters, digits, underscores (_), hyphens (-), and periods (.) are allowed.
bandwidth_chargemode	No	String	Bandwidth mode. <ul style="list-style-type: none"> bandwidth: billed by bandwidth traffic: billed by traffic Default: bandwidth
bandwidth_size	Yes for public network load balancers	Integer	Bandwidth size. The default value is 1 to 2000 Mbit/s. Configure this parameter based on the bandwidth range allowed in your region. The minimum increment for bandwidth adjustment varies depending on the bandwidth range. <ul style="list-style-type: none"> The minimum increment is 1 Mbit/s if the allowed bandwidth does not exceed 300 Mbit/s. The minimum increment is 50 Mbit/s if the allowed bandwidth ranges from 300 Mbit/s to 1000 Mbit/s. The minimum increment is 500 Mbit/s if the allowed bandwidth exceeds 1000 Mbit/s.
bandwidth_sharetype	Yes for public network load balancers	String	Bandwidth sharing mode. <ul style="list-style-type: none"> PER: dedicated bandwidth
eip_type	Yes for public network load balancers	String	EIP type. <ul style="list-style-type: none"> 5_bgp: dynamic BGP The specific type varies with regions. For details, see the EIP console.
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. The value of this parameter must be the same as that of l4_flavor_name , that is, both are elastic specifications or fixed specifications.
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: <pre>"elb_virsubnet_ids": ["14567f27-8ae4-42b8-ae47-9f847a4690dd"]</pre>

11.4.2.4 Configuring HTTPS Certificates for ELB Ingresses

Ingress supports TLS certificate configuration and secures your Services with HTTPS.

Currently, you can use the TLS secret certificate configured in the cluster and the ELB certificate.

 **NOTE**

If HTTPS is enabled for the same port of the same load balancer of multiple ingresses, you must select the same certificate.

Using a TLS Secret Certificate

Step 1 Use `kubectl` to connect to 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 Secret](#).

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 Base64-encoded.

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 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

NOTE

Default security policy (kubernetes.io/elb.tls-ciphers-policy) is supported only in clusters of v1.17.17 or later.

The following uses the automatically created load balancer as an example. The YAML file is configured 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-0a"
    ],
    "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: foo.bar.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

```

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-0a"
        ],
        "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: foo.bar.com
    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

```

Table 11-33 Key parameters

Parameter	Mandatory	Type	Description
kubernetes.io/elb.tls-ciphers-policy	No	String	<p>The default value is tls-1-2, which is the default security policy used by the listener and takes effect only when HTTPS is used.</p> <p>Options:</p> <ul style="list-style-type: none"> • tls-1-0 • tls-1-1 • tls-1-2 • tls-1-2-strict <p>For details of cipher suites for each security policy, see Table 11-34.</p>
tls	No	Array of strings	<p>When HTTPS is used, this parameter must be added to specify the secret certificate.</p> <p>Multiple independent domain names and certificates can be added. For details, see Configuring the Server Name Indication (SNI) for ELB Ingresses.</p>
secretName	No	String	<p>This parameter is mandatory if HTTPS is used. Set this parameter to the name of the created secret.</p>

Table 11-34 `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	
tls-1-2	TLS 1.2	

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

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 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

Using the ELB Certificate

To use the ELB certificate, you can specify the annotations **kubernetes.io/elb.tls-certificate-ids**.

NOTE

1. If you specify both the IngressTLS certificate and the ELB certificate, the latter is used.
2. CCE does not check whether the ELB certificate is valid. It only checks whether the certificate exists.
3. Only clusters of v1.19.16-r2, v1.21.5-r0, v1.23.3-r0, or later support the ELB certificate.

For clusters of v1.21 or earlier:

```
apiVersion: networking.k8s.io/v1 beta1
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: 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
```

11.4.2.5 Configuring the Server Name Indication (SNI) for ELB Ingresses

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

- This function is supported only in clusters of v1.15.11 and later.
- 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.

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.

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-0a"
        ],
        "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.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: 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

```

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-0a"
        ],
        "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.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: example.com
  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
```

11.4.2.6 ELB Ingresses Routing to Multiple Services

Ingresses can route 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 used to access 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
```

11.4.2.7 ELB Ingresses Using HTTP/2

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 HTTP2 requests, you can add the following field to the ingress annotation to enable the use of HTTP/2:

```
kubernetes.io/elb.http2-enable: 'true'
```

The following shows the YAML file for associating with an existing load balancer:

For clusters of v1.21 or earlier:

```
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.port: '443'
    kubernetes.io/ingress.class: cce
    kubernetes.io/elb.http2-enable: 'true' # Enable HTTP/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 # Replace it with the port number of your target Service.
    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.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: 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
```

Table 6 HTTP/2 parameters

Parameter	Mandatory	Type	Description
kubernetes.io/elb.http2-enable	No	Bool	<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. This parameter is supported in clusters of v1.19.16-r0, v1.21.3-r0, and later versions.</p> <p>Options:</p> <ul style="list-style-type: none"> • true: enabled • false: disabled (default value) <p>Note: HTTP/2 can be enabled or disabled only when the listener uses HTTPS. This parameter is invalid when the listener protocol is HTTP, and defaults to false.</p>

11.4.2.8 Interconnecting ELB Ingresses with HTTPS Backend Services

Ingress can interconnect with backend services of 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:

```
kubernetes.io/elb.pool-protocol: https
```

Constraints

- This feature only applies to clusters of v1.23.8, v1.25.3, and later.
- Ingress can interconnect with HTTPS backend services only when dedicated load balancers are used.
- When interconnecting with HTTPS backend services, set **Client Protocol** of ingress to **HTTPS**.

Interconnecting with HTTPS Backend Services

An ingress configuration example:

```
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.
    # Replace its ID with the ID of your dedicated load balancer.
    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
```

11.4.2.9 Configuring Timeout for an ELB Ingress

The ELB 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 setting for waiting for a request from a client: If the client fails to send a request header to the load balancer within the timeout duration or the interval for sending body data exceeds a specified period, the load balancer will close the connection.
- Timeout setting for waiting for a response from a backend server: If the backend server fails to respond after the timeout duration elapses, the load balancer will stop waiting and return HTTP 504 Gateway Timeout to the client.

Constraints

- This feature takes effect only in the following versions:
 - v1.19: v1.19.16-r30 or later
 - v1.21: v1.21.10-r10 or later
 - v1.23: v1.23.8-r10 or later
 - v1.25: v1.25.3-r10 or later
- The timeout can be configured for Ingresses only when dedicated load balancers are used.
- If you delete the timeout configuration during Ingress update, the timeout configuration on the existing listeners will be retained.

Configuring Timeout

An Ingress configuration example 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 duration 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 11-35 Key annotation parameters

Parameter	Mandatory	Type	Description
kubernetes.io/elb.keepalive_timeout	No	Integer	Timeout for client connections. If there are no requests reaching the load balancer after the timeout duration elapses, the load balancer will disconnect the connection with the client and establish a new connection when there is a new request. The value ranges from 0 to 4000 (in seconds). The default value is 60 .
kubernetes.io/elb.client_timeout	No	Integer	Timeout duration for waiting for a response from a client. There are two situations: <ul style="list-style-type: none"> • If the client fails to send a request header to the load balancer within 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. The value ranges from 1 to 300 (in seconds). The default value is 60 .

Parameter	Mandatory	Type	Description
kubernetes.io/elb.member_timeout	No	Integer	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 within the duration specified by member_timeout , the load balancer will stop waiting and return HTTP 504 Gateway Timeout. The value ranges from 1 to 300 (in seconds). The default value is 60 .

11.4.3 Nginx Ingresses

11.4.3.1 Creating 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](#).
- 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 an Nginx ingress, ensure that the nginx-ingress add-on has been installed in the cluster. For details, see [Installing the Add-on](#).

Precautions

- **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 **Networking** in the navigation pane, click the **Ingresses** tab, and click **Create Ingress** in the upper right corner.

Step 3 Configure ingress parameters.

- **Name:** Specify a 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 when the **NGNIX Ingress Controller** 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:

TLS: nginx-ingress supports HTTP and HTTPS. The default listening port reserved during nginx-ingress installation is **80** for HTTP requests and **443** for HTTPS requests. To use HTTPS, configure the server certificate.

- **Server Certificate:** When creating an HTTPS listener, bind a TLS certificate to support encrypted authentication for HTTPS data transmission. For details on how to create a secret, see [Creating a Secret](#).
- **SNI:** Server Name Indication (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. 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.
 - **URL 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.
 - **URL:** access path to be registered, 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 that your Nginx application contains the same URL, that is, `/usr/share/nginx/html/test`, otherwise, 404 is returned.
- **Destination Service:** Select an existing Service or create a Service. Services that do not meet search criteria are automatically filtered out.
- **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 After the configuration is complete, click **OK**.

After the ingress is created, it is displayed in the ingress list.

----End

11.4.3.2 Using kubectl to Create an Nginx Ingress

Scenario

This section uses an [Nginx workload](#) as an example to describe how to create an Nginx ingress using kubectl.

Prerequisites

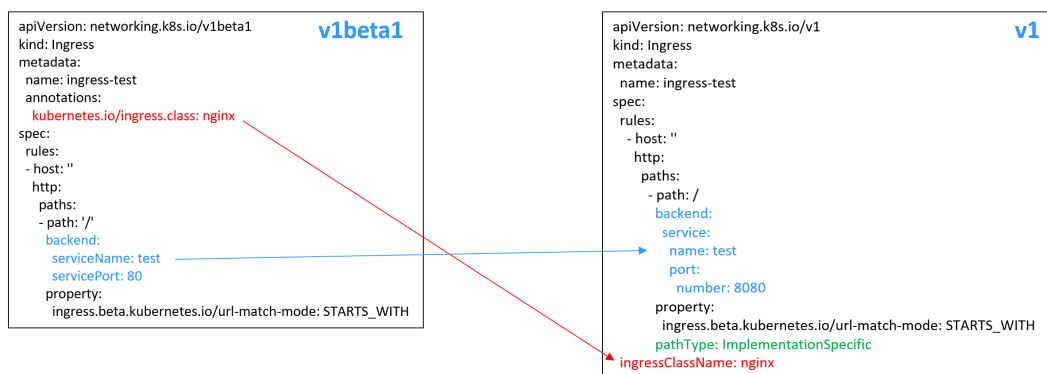
- The nginx-ingress add-on has been installed in a cluster. For details, see [Installing the Add-on](#).
- An ingress provides network access for backend workloads. Ensure that a workload is available in a cluster. If no workload is available, deploy a workload by referring to [Creating a Deployment](#), [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](#).

Ingress Description of `networking.k8s.io/v1`

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 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 Nginx 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, 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 Description of networking.k8s.io/v1](#).

The following uses HTTP as an example to describe how to configure the YAML file:

If the node is in a cluster 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.

```

If the node is in a cluster 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 11-36 Key parameters

Parameter	Mandatory	Type	Description
kubernetes.io/ingress.class	Yes (only for clusters of v1.21 or earlier)	String	nginx : indicates that Nginx ingress is used. This option cannot be used if the nginx-ingress add-on is not 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	nginx : indicates that Nginx ingress is used. This option cannot be used if the nginx-ingress add-on is not 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 archived. Once a domain name rule is configured, 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 host and path.</p> <p>NOTE</p> <ul style="list-style-type: none"> 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 that your Nginx application contains the same URL, that is, /usr/share/nginx/html/test, otherwise, 404 is returned.
ingress.beta.kubernetes.io/url-match-mode	No	String	<p>Route matching policy.</p> <p>Default: STARTS_WITH (prefix match)</p> <p>Options:</p> <ul style="list-style-type: none"> EQUAL_TO: exact match STARTS_WITH: prefix match

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"> • ImplementationSpecific: The matching method depends on Ingress Controller. The matching method defined by ingress.beta.kubernetes.io/url-match-mode is used in CCE. • Exact: exact matching of the URL, which is case-sensitive. • Prefix: 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. <p>NOTE</p> <ul style="list-style-type: none"> - 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, /foo/bar matches /foo/bar/baz but does not match /foo/barbaz. - 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, /foo/bar matches /foo/bar/. <p>See examples 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
```

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 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

11.4.3.3 Configuring HTTPS Certificates for Nginx Ingresses

HTTPS certificates can be configured for ingress to provide security services.

Step 1 Use `kubectl` to connect to the cluster. For details, see [Connecting to a Cluster Using kubectl](#).

Step 2 Ingress supports two TLS key 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 Secret](#).

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 Base64-encoded.

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 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 8080 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
```

View the created ingress.

kubectl get ingress

If information similar to the following is displayed, the ingress has been created 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

11.4.3.4 Configuring URL Rewriting Rules for Nginx Ingresses

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 Rewriting Rules

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: /$2
spec:
  rules:
    - host: 'rewrite.bar.com'
      http:
        paths:
          - path: '/something(/|$)(.*)'
            backend:
              service:
                name: <your_service_name> # Replace it with the name of your target Service.
                port:
                  number: <your_service_port> # Replace 8080 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: /$2
spec:
  rules:
    - host: 'rewrite.bar.com'
      http:
        paths:
          - path: '/something(/|$)(.*)'
            backend:
              serviceName: <your_service_name> # Replace it with the name of your target Service.
              servicePort: <your_service_port> # Replace 8080 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 indicates that all characters matched by the second parenthesis (.) are filled in the **nginx.ingress.kubernetes.io/rewrite-target** annotation.

For example, the preceding ingress definition will result in the following rewrites:

- `rewrite.bar.com/something` rewrites to `rewrite.bar.com/`.
- `rewrite.bar.com/something/` rewrites to `rewrite.bar.com/`.
- `rewrite.bar.com/something/new` rewrites to `rewrite.bar.com/new`.

In the `nginx-ingress-controller` container, you can view all ingress configurations in the **nginx.conf** file in the `/etc/nginx` directory. The rewriting rule in the preceding example generates a Rewrite command and writes it to the **location** field in the **nginx.conf** file.

```
## start server rewrite.bar.com
server {
    server_name rewrite.bar.com ;
    ...
    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 rewrite.bar.com
```

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.

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 configuration to the **server** field in the `nginx.conf` file.
- `nginx.ingress.kubernetes.io/configuration-snippet`: Add custom configuration to the **location** field in the `nginx.conf` file.

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 two rules, the **/something** path is added to the access URL.

- When a user accesses `rewrite.bar.com/stylesheets/new.css`, it rewrites to `rewrite.bar.com/something/stylesheets/new.css`.
- When a user accesses `rewrite.bar.com/images/new.jpg`, it rewrites to `rewrite.bar.com/something/images/new.jpg`.

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 also use the following annotation to forcibly redirect requests to HTTPS.

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 8080 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 8080 with the port number of your target Service.

```

11.4.3.5 Interconnecting Nginx Ingresses with HTTPS Backend Services

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:

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 8080 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 8080 with the port number of your target Service.

```


11.4.3.6 Nginx Ingresses Using Consistent Hashing for Load Balancing

The native Nginx supports multiple load balancing rules, including weighted round robin and IP hash. An 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 the 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

An Nginx ingress can use the `nginx.ingress.kubernetes.io/upstream-hash-by` annotation to configure consistent hashing rules. The following is an example:

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 8080 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 8080 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.

Documentation

[Custom NGINX upstream hashing](#)

11.4.3.7 Configuring Nginx Ingresses Using Annotations

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 URL Rewriting Rule](#)
- [Interconnecting with HTTPS Backend Services](#)
- [Creating a Consistent Hashing Rule for Load Balancing](#)
- [Customized Timeout Interval](#)
- [Customizing Body Size](#)
- [Documentation](#)

Ingress Type

Table 11-37 Ingress type annotations

Parameter	Type	Description	Supported Cluster Version
kubernetes.io/ingress.class	String	<ul style="list-style-type: none"> nginx: Nginx ingress is used. cce: The self-developed ELB ingress is used. <p>This parameter is mandatory when an ingress is created by calling the API. For clusters of v1.23 or later, use the parameter ingressClassName. For details, see Using kubectl to Create an Nginx Ingress.</p>	Only clusters of v1.21 or earlier

For details about how to use the preceding annotations, see [Using kubectl to Create an Nginx Ingress](#).

Configuring a URL Rewriting Rule

Table 11-38 URL rewriting rule annotations

Parameter	Type	Description
nginx.ingress.kubernetes.io/rewrite-target	String	Target URI where the traffic must be redirected.
nginx.ingress.kubernetes.io/ssl-redirect	Bool	Indicates whether the access is available only through SSL. The default value is true when the ingress contains a certificate.
nginx.ingress.kubernetes.io/force-ssl-redirect	Bool	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 about the application scenarios, see [Configuring URL Rewriting Rules for Nginx Ingresses](#).

Interconnecting with HTTPS Backend Services

Table 11-39 Annotations for interconnecting with HTTPS backend services

Parameter	Type	Description
nginx.ingress.kubernetes.io/backend-protocol	String	If this parameter is set to HTTPS , HTTPS is used to forward requests to the backend service container.

For details about the application scenarios, see [Interconnecting Nginx Ingresses with HTTPS Backend Services](#).

Creating a Consistent Hashing Rule for Load Balancing

Table 11-40 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"> 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.

For details about the application scenarios, see [Nginx Ingresses Using Consistent Hashing for Load Balancing](#).

Customized Timeout Interval

Table 11-41 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 Body Size

Table 11-42 Annotations of customizing 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 is returned to the client. You can use this parameter to adjust the upper limit of the body size. Example: nginx.ingress.kubernetes.io/proxy-body-size: 8m

Documentation

For details about annotation parameters supported by Nginx ingresses, see [Annotations](#).

11.5 DNS

11.5.1 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.

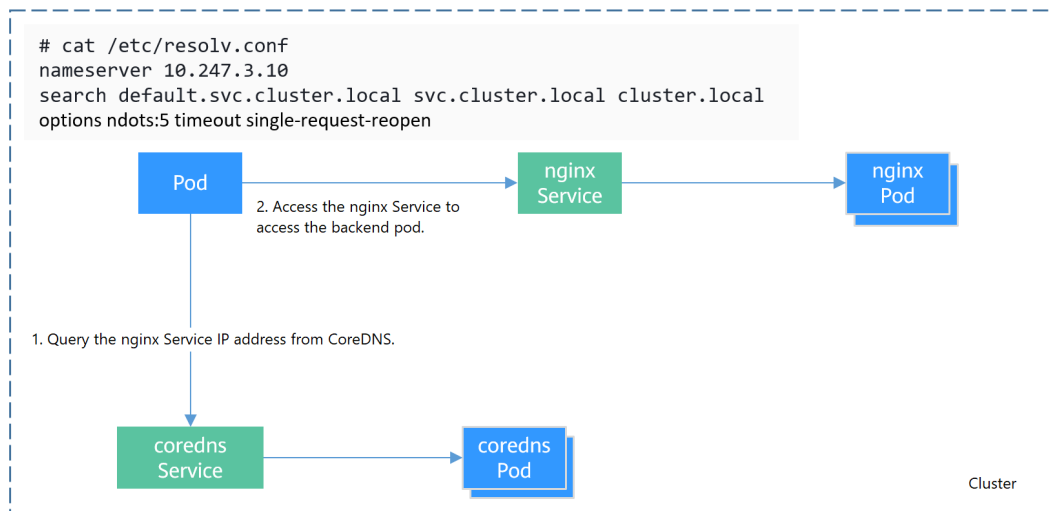
```
$ kubectl get svc -n kube-system
NAME          TYPE        CLUSTER-IP   EXTERNAL-IP   PORT(S)          AGE
coredns       ClusterIP   10.247.3.10  <none>        53/UDP,53/TCP,8080/TCP   13d
```

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 11-22 Example of domain name resolution in a cluster



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.

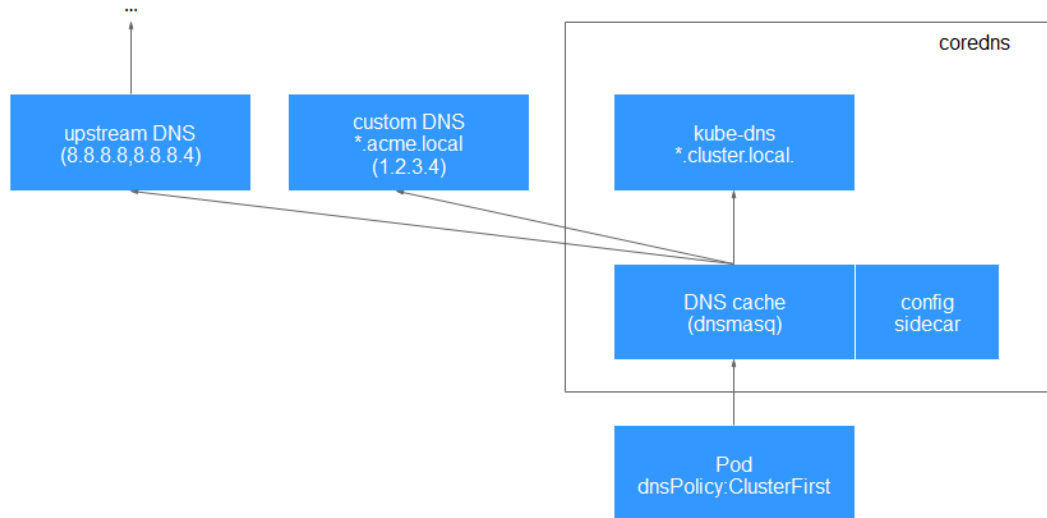
Routing

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 11-23 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](#).

11.5.2 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.***.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 11-43](#).
 - **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: nameservers** in the **dnsConfig**. You can configure the domain name server for the 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 11-43 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 (search option) and ndots: 5 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 --resolv-conf 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 dnsPolicy to ClusterFirstWithHostNet 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 hostNetwork: true dnsPolicy: ClusterFirstWithHostNet

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 --resolv-conf flag points to. In this case, a cloud DNS is used for CCE clusters. Both search and options 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 dnsPolicy is set to None , the dnsConfig field must be specified because all DNS settings are supposed to be provided using the dnsConfig 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 11-44 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 dnsPolicy . Duplicate entries are removed.
nameservers	A list of IP addresses that will be used as DNS servers. If workload's dnsPolicy is set to None , the list must contain at least one IP address, otherwise this property is optional. The servers listed will be combined to the nameservers generated from the specified DNS policy in dnsPolicy with duplicate addresses removed.

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 dnsPolicy . 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
```

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
```

11.5.3 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 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.

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**. On the displayed page, 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.XXX.XXX.XXX`.
- 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.

The corresponding Corefile content is as follows:

```
.:5353 {
  bind {$POD_IP}
  cache 30
  errors
  health {$POD_IP}:8080
  kubernetes cluster.local in-addr.arpa ip6.arpa {
    pods insecure
```



```
    fallthrough in-addr.arpa ip6.arpa
  }
  loadbalance round_robin
  prometheus {$POD_IP}:9153
  forward . /etc/resolv.conf {
    policy random
  }
  reload
  ready {$POD_IP}:8081
}
consul.local:5353 {
  bind {$POD_IP}
  errors
  cache 30
  forward . 10.150.0.1
}
```

----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**. On the displayed page, click **Edit** under **CoreDNS**.
- Step 3** Edit the advanced configuration under **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 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/>.

- 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.

The corresponding Corefile content is as follows:

```
.: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
}
```

----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. 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**. On the displayed page, click **Edit** under **CoreDNS**.
- Step 3** Edit the advanced configuration under **Parameters** and add the following content to the **plugins** field:

```
{
  "name": "rewrite",
  "parameters": "name example.com example.default.svc.cluster.local"
}
```

- 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.

Corresponding Corefile content:

```
.:5353 {
  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  
}
```

----End

Using CoreDNS to Cascade Self-Built DNS

By default, CoreDNS uses the `/etc/resolv.conf` file of the node for resolution. 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**. On the displayed page, click **Edit** under **CoreDNS**.
- Step 3** Edit the advanced configuration under **Parameters** and modify the following content in the **plugins** field:

```
{  
  "configBlock": "policy random",  
  "name": "forward",  
  "parameters": ". 192.168.1.1"  
}
```

- 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.

The corresponding Corefile content is as follows:

```
.: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  
}
```

----End

11.6 Container Network Settings

11.6.1 Host Network

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

If a pod uses the host network, it occupies a host port. The pod IP is the host IP. To use the host network, you must confirm pods do not conflict with each other in terms of the host ports they occupy. Do not use the host network unless you know exactly which host port is used by which pod.

When using the host network, you access the node to access a pod on it. 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()

```

11.6.2 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.

Constraints

The following shows constraints on setting the rate limiting for inter-pod access:

Constraint Type	Tunnel Network Model	VPC Network Model	Cloud Native 2.0 Network Model
Supported versions	All versions	Clusters of v1.19.10 and later	Clusters of v1.19.10 and later

Constraint Type	Tunnel Network Model	VPC Network Model	Cloud Native 2.0 Network Model
Supported runtime types	Only common containers (runC as the container runtime) are supported. Secure containers are not supported.	Only common containers (runC as the container runtime) are supported. Secure containers (Kata as the container runtime) are not supported.	Only common containers (runC as the container runtime) are supported. Secure containers (Kata 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"> Pods access external cloud service CIDR blocks 100.64.0.0/10 and 214.0.0.0/8. Traffic rate limiting of health check
Upper rate limit	Minimum value between the upper bandwidth limit and 34 Gbit/s	Minimum value between the upper bandwidth limit and 4.3 Gbit/s	Minimum value between the upper bandwidth limit and 4.3 Gbit/s
Lower rate limit	Only the rate limit of Kbit/s or higher is supported.	Currently, only the rate limit of Mbit/s or higher is supported.	

Using the CCE Console

When creating a workload on the console, you can set pod ingress and egress bandwidth limits on the **Advanced Settings > Network Configuration** area.

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.

11.6.3 Container Tunnel Network Settings

11.6.3.1 Network Policies

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. (Only egress rules support IP blocks.)

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: Only the following OSs and cluster versions support egress rules.

OS	Cluster Version	Verified Kernel Version
CentOS	v1.23 or later	3.10.0-1062.18.1.el7.x86_64 3.10.0-1127.19.1.el7.x86_64 3.10.0-1160.25.1.el7.x86_64
EulerOS 2.5	v1.23 or later	3.10.0-862.14.1.5.h591.eulerosv2r7.x86_64 3.10.0-862.14.1.5.h687.eulerosv2r7.x86_64
EulerOS 2.9	v1.23 or later	4.18.0-147.5.1.6.h541.eulerosv2r9.x86_64 4.18.0-147.5.1.6.h766.eulerosv2r9.x86_64

- Network isolation is not supported for IPv6 addresses.
- If upgrade to a cluster version that supports egress rules is performed in in-place mode, you cannot use egress rules because the node OS is not upgraded. In this case, reset the node.

Using Ingress Rules

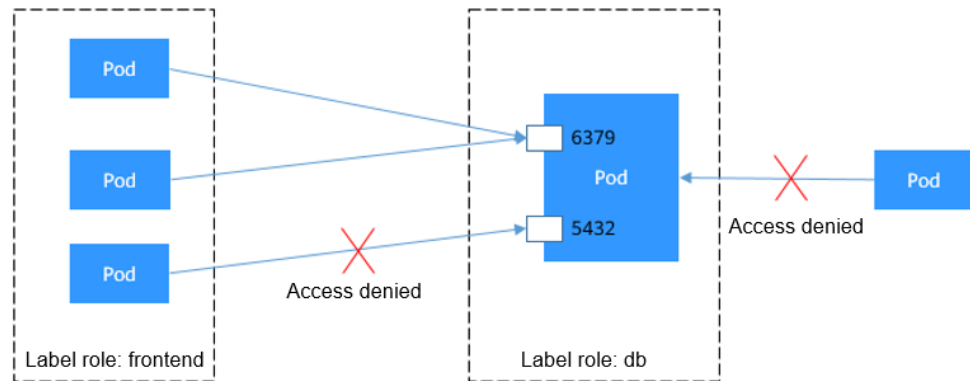
- **Using podSelector to specify the access scope**

```

apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: test-network-policy
  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 traffic from the pods with the "role=frontend" label is allowed.
          matchLabels:
            role: frontend
      ports:             # Only TCP can be used to access port 6379.
        - protocol: TCP
          port: 6379
    
```

The following figure shows how podSelector works.

Figure 11-24 podSelector



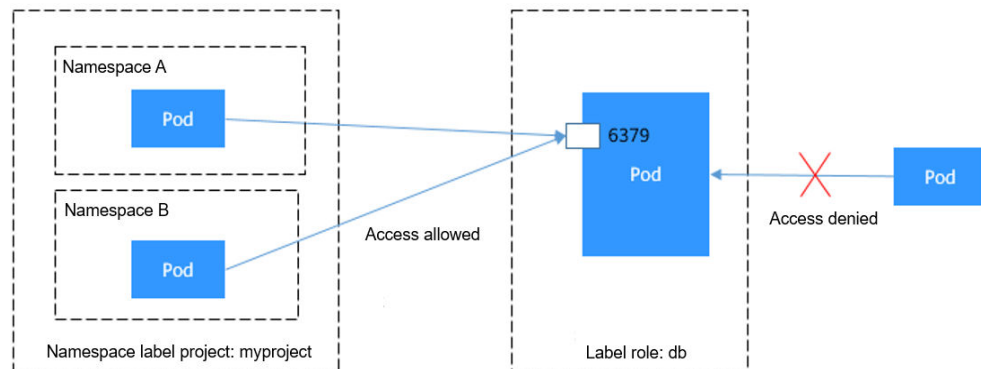
- **Using namespaceSelector to specify the access scope**

```

apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: test-network-policy
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 traffic from the pods in the namespace with the
      "project=myproject" label is allowed.
      matchLabels:
        project: myproject
    ports:               # Only TCP can be used to access port 6379.
    - protocol: TCP
      port: 6379
  
```

The following figure shows how namespaceSelector works.

Figure 11-25 namespaceSelector



Using Egress Rules

Egress supports not only podSelector and namespaceSelector, but also ipBlock.

NOTE

Only clusters of version 1.23 or later support egress rules. Currently, nodes running EulerOS 2.5, EulerOS 2.9, and CentOS 7.X are supported.

```

apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
  
```

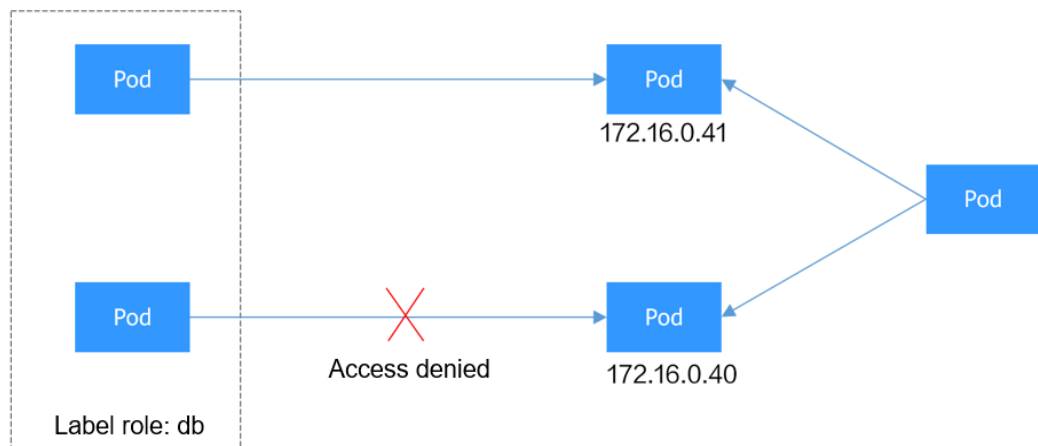
```

metadata:
  name: deny-client-a-via-except-cidr-egress-rule
  namespace: default
spec:
  policyTypes:
    # Must be specified for an egress rule.
    - Egress
  podSelector:
    # The rule takes effect for pods with the role=db label.
    matchLabels:
      role: db
  egress:
    # Egress rule
    - to:
      - ipBlock:
          cidr: 172.16.0.16/16 # Allow access to this CIDR block.
        except:
          - 172.16.0.40/32 # This CIDR block cannot be accessed. This value must fall within the range
            specified by cidr.

```

The following figure shows how ipBlock works.

Figure 11-26 ipBlock



You can define ingress and egress in the same rule.

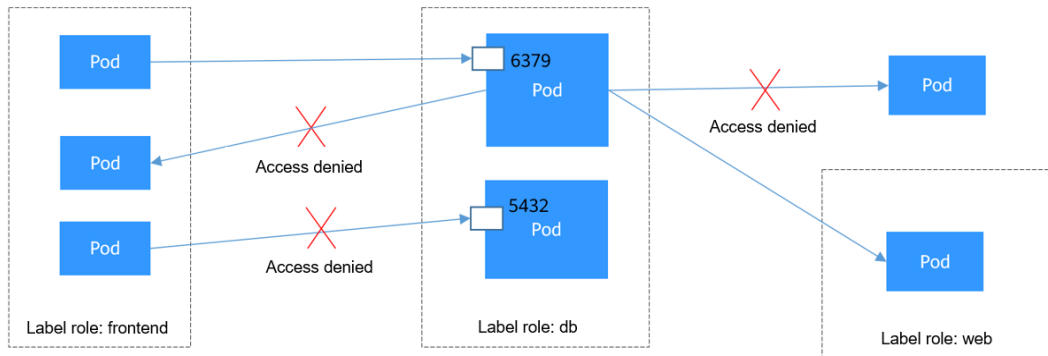
```

apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: test-network-policy
  namespace: default
spec:
  policyTypes:
    - Ingress
    - Egress
  podSelector:
    # The rule takes effect for pods with the role=db label.
    matchLabels:
      role: db
  ingress:
    # This is an ingress rule.
    - from:
      - podSelector:
          # Only traffic from the pods with the "role=frontend" label is allowed.
          matchLabels:
            role: frontend
      ports:
        # Only TCP can be used to access port 6379.
        - protocol: TCP
          port: 6379
  egress:
    # Egress rule
    - to:
      - podSelector:
          # Only pods with the role=web label can be accessed.
          matchLabels:
            role: web

```

The following figure shows how to use ingress and egress together.

Figure 11-27 Using both ingress and egress



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 **Networking** 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 **Confirm**. You can also click **Reference Workload Label** to reference the label of an existing workload.
 - **Inbound Rule:** Click **+** to add an inbound rule. For details about parameter settings, see [Table 11-45](#).

Table 11-45 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

11.6.4 Cloud Native Network 2.0 Settings

11.6.4.1 Security Group Policies

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.


Constraints

- 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 page and click **Create**.
- Step 4** Set the parameters as described in [Table 11-46](#).

Table 11-46 Configuration parameters

Parameter	Description	Example Value
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 Create Security Group . After the security group is created, click the refresh button. NOTICE <ul style="list-style-type: none"> • A maximum of five security groups can be selected. • Hover the cursor on  next to the security group name, and you can view details about the security group. 	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 connect to 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 11-47](#) describes the parameters in the YAML file.

Table 11-47 Description

Field	Description	Mandatory
apiVersion	API version. The value is crd.yangtse.cni/v1 .	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 view 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

11.6.4.2 NetworkAttachmentDefinition

Scenario

In a CCE Turbo cluster, you can set the subnet and security group for a container by namespace using NetworkAttachmentDefinition, a **CRD** resource in the cluster. After NetworkAttachmentDefinition is configured for a namespace, pods in the namespace support the following functions:

- Binding a container with a subnet: The pod IP address is restricted in a specific CIDR block. Different namespaces can be isolated from each other.
- Binding a container with a security group: Security group rules can be set for pods in the same namespace to customize access policies.

Constraints

- NetworkAttachmentDefinition is available only in CCE Turbo clusters of v1.23.8-r0, v1.25.3-r0, and later.
- Only **default-network** supports ENI preheating. User-defined container subnets do not support ENI preheating. If ENI preheating is not enabled, workload instance creation slows down. Therefore, this function is not applicable to high-performance pod creation scenarios.
- To delete a NetworkAttachmentDefinition, delete pods (with the annotation named **cni.yangtse.io/network-status**) created using the configuration in the corresponding namespace first. For details, see [Deleting a Network Configuration](#).

Using the CCE Console

Step 1 Log in to the CCE console.

Step 2 Click the cluster name to access the cluster console. Choose **System Configuration** in the navigation pane and click the **Network Configuration** tab.

NOTE

Each cluster has a **default-network** for namespaces with no container subnets. The default container subnet displayed in the network information on the networking configuration area is the container subnet in **default-network**. The **default-network** cannot be deleted.

- Step 3** Click **Create Network Configurations** in the upper right corner. Configure the basic parameters in the displayed dialog box.
- **Name:** Enter a name that contains a maximum of 253 characters. Do not use **default-network**, **default**, **mgnt0**, and **mgnt1**.
 - **Namespace:** Select a namespace. The namespaces of different 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 a subnet. After the subnet is created, click the refresh button. A maximum of 20 subnets can be selected.
 - **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.
- Step 4** Click **Create**. After the creation is complete, you will be redirected to the network configuration list. You can see that the newly added subnet is in the list.

----End

Using kubectl

This section describes how to create an NAD using kubectl.

- Step 1** Use kubectl to connect to the cluster. For details, see [Connecting to a Cluster Using kubectl](#).
- Step 2** Modify the **networkattachment-test.yaml** file.

vi networkattachment-test.yaml

```
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 11-48 Key parameters

Parameter	Mandatory	Type	Description
apiVersion	Yes	String	API version. The value is fixed at k8s.cni.cncf.io/v1 .
kind	Yes	String	Type of the object to be created. The value is fixed at NetworkAttachmentDefinition .
yangtse.io/ project-id	Yes	String	Project ID.
name	Yes	String	Configuration item name.
namespace	Yes	String	Namespace of the configuration resource. The value is fixed to kube-system .
config	Yes	Table 11-49 object	Configuration content, which is a string in JSON format.

Table 11-49 config parameters

Parameter	Mandatory	Type	Description
type	Yes	String	The value is fixed at eni-neutron .
args	No	Table 11-50 object	Configuration parameters.
selector	No	Table 11-51 object	Namespace on which the configuration takes effect.

Table 11-50 args parameters

Parameter	Mandatory	Type	Description
securityGroups	No	String	<p>Security group ID. If no security group is planned, select the same security group as that in default-network.</p> <p>Obtaining the value:</p> <p>Log in to the VPC console. In the navigation pane, choose Access Control > Security Groups. Click the target security group name and copy the ID on the Summary 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:</p> <pre>[{"subnetID":"27d95***"}, {"subnetID":"827bb***"}, {"subnetID":"bdd6b***"}]</pre> <p>Subnet ID not used by the cluster in the same VPC.</p> <p>Obtaining the value:</p> <p>Log in to the VPC console. In the navigation pane, choose Virtual Private Cloud > Subnets. Click the target subnet name and copy the Subnet ID on the Summary tab page.</p>

Table 11-51 selector parameters

Parameter	Mandatory	Type	Description
namespaceSelector	No	matchLabels Object	<p>A Kubernetes standard selector. Enter the namespace label in the following format:</p> <pre>"matchLabels":{ "kubernetes.io/metadata.name":"default" }</pre> <p>The namespaces of different configurations cannot overlap.</p>

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
```

----End

Deleting a Network Configuration

You can delete the new network configuration or view its YAML file.

NOTE

Before deleting a network configuration, delete the container corresponding to the configuration. Otherwise, the deletion fails.

1. Run the following command to filter the pod that uses the configuration in the cluster (**example** is an example configuration name and you should replace it):

```
kubectl get po -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.

11.7 Cluster Network Settings

11.7.1 Switching a Node Subnet

Scenario

This section describes how to switch subnets for nodes in a cluster.

Constraints

- Only subnets in the same VPC as the cluster can be switched. The security group of the node cannot be switched.

Procedure

Step 1 Log in to the ECS console.

Step 2 Click **More > Manage Network > Change VPC** in the **Operation** column of the target ECS.

Step 3 Set parameters for changing the VPC.

- **VPC:** Select the same VPC as that of the cluster.
- **Subnet:** Select the target subnet to be switched.
- **Private IP Address:** Select **Assign new** or **Use existing** as required.
- **Security Group:** Select the security group of the cluster node. Otherwise, the node is unavailable.

Step 4 Click **OK**.

Step 5 Go to the CCE console and reset the node. You can use the default parameter settings. For details, see [Resetting a Node](#).

----End

11.7.2 Adding a Container CIDR Block for a Cluster

Scenario

If the container CIDR block (container subnet in a CCE Turbo cluster) set during CCE cluster creation is insufficient, you can add a container CIDR block for the cluster.


Constraints

- This function applies to CCE clusters and CCE Turbo clusters of v1.19 or later, but not to clusters using container tunnel networking.
- The container CIDR block or container subnet cannot be deleted after being added. Exercise caution when performing this operation.

Adding a Container CIDR Block for a CCE Cluster

Step 1 Log in to the CCE console and click the cluster name to access the cluster console.

Step 2 On the **Cluster Information** page, click **Add Container CIDR Block** in the **Networking Configuration** area.

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.

Step 4 Click **OK**.

----End

Adding a Container Subnet for a CCE Turbo Cluster

Step 1 Log in to the CCE console and access the CCE Turbo cluster console.

Step 2 On the **Cluster Information** page, locate the **Networking Configuration** area and click **Add Pod Subnet**.

Step 3 Select a container subnet in the same VPC. You can add multiple container subnets at a time. If no other container subnet is available, go to the VPC console to create one.

Step 4 Click **OK**.

----End

11.8 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 2.0 Network**

In the Cloud Native 2.0 network 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.

11.9 Accessing Public Networks from a Container

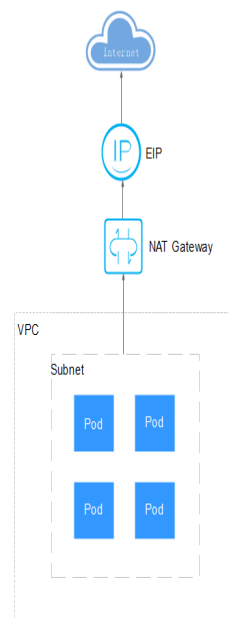
Containers can access public networks in either of the following ways:

- 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 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 11-28 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 11-28 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 **Assign EIP**.

5. Configure 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 **Create Public NAT Gateway** in the upper right corner.
5. Configure 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. Configure 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.
- Cloud Native Network 2.0: Select the subnet where the container is located, that is, the container subnet selected during cluster 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 container subnet (Cloud Native 2.0 Network) or 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

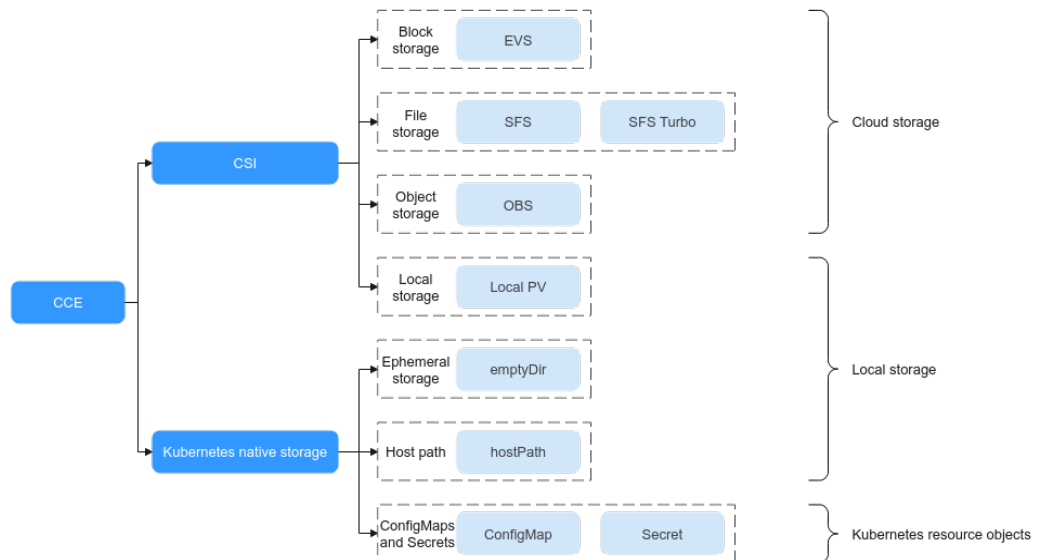
12 Storage

12.1 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.

Figure 12-1 Container storage type



CCE allows you to mount cloud storage volumes to your pods. Their features are described below.

Table 12-1 Cloud storage comparison

Dimension	EVS	SFS	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 petabytes, SFS provides fully hosted shared file storage, highly available and stable to handle data- and bandwidth-intensive applications in HPC, media processing, file sharing, content management, and web services.	Expandable to 320 TB, SFS Turbo provides a 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 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.

Dimension	EVS	SFS	SFS Turbo	OBS
Access mode	Accessible only after being mounted to ECSs or BMSs and initialized.	Mounted to ECSs or BMSs using network protocols. A network address must be specified or mapped to a local directory for access.	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 provisioning	Supported. For details, see Using an Existing EVS Disk Through a Static PV.	Supported. For details, see Using an Existing SFS File System Through a Static PV.	Supported. For details, see Using an Existing SFS Turbo File System Through a Static PV.	Supported. For details, see Using an Existing OBS Bucket Through a Static PV.
Dynamic provisioning	Supported. For details, see Using an EVS Disk Through a Dynamic PV.	Supported. For details, see Using an SFS File System Through a Dynamic PV.	Not supported	Supported. For details, see Using an OBS Bucket Through a Dynamic PV.
Features	Non-shared storage. Each volume can be mounted to only one node.	Shared storage featuring high performance and throughput	Shared storage featuring high performance and bandwidth	Shared, user-mode file system

Dimension	EVS	SFS	SFS Turbo	OBS
Usage	HPC, enterprise core cluster applications, enterprise application systems, and dev/test NOTE HPC apps here require high-speed and high-IOPS storage, such as industrial design and energy exploration.	HPC, media processing, content management, web services, big data, and analysis applications NOTE HPC apps here require high bandwidth and shared file storage, such as gene sequencing and image rendering.	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	SFS 1.0: PB	General-purpose: TB	EB
Latency	1–2 ms	SFS 1.0: 3–20 ms	General-purpose: 1–5 ms	10 ms
IOPS/TPS	33,000 for a single disk	SFS 1.0: 2,000	General-purpose: up to 100,000	Tens of millions
Bandwidth	MB/s	SFS 1.0: GB/s	General-purpose: up to GB/s	TB/s

Enterprise Project Support

NOTE

To use this function, the everest add-on must be upgraded to v1.2.33 or later.

- Automatically creating storage:

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.
- 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 \(EVS\)](#)
- [Scalable File Service \(SFS\)](#)
- [SFS Turbo File Systems](#)
- [Object Storage Service \(OBS\)](#)

12.2 Storage Basics

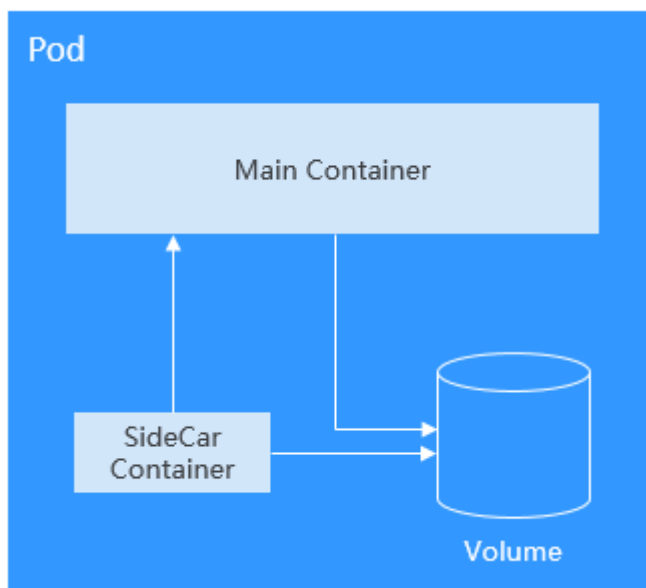
Volumes

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

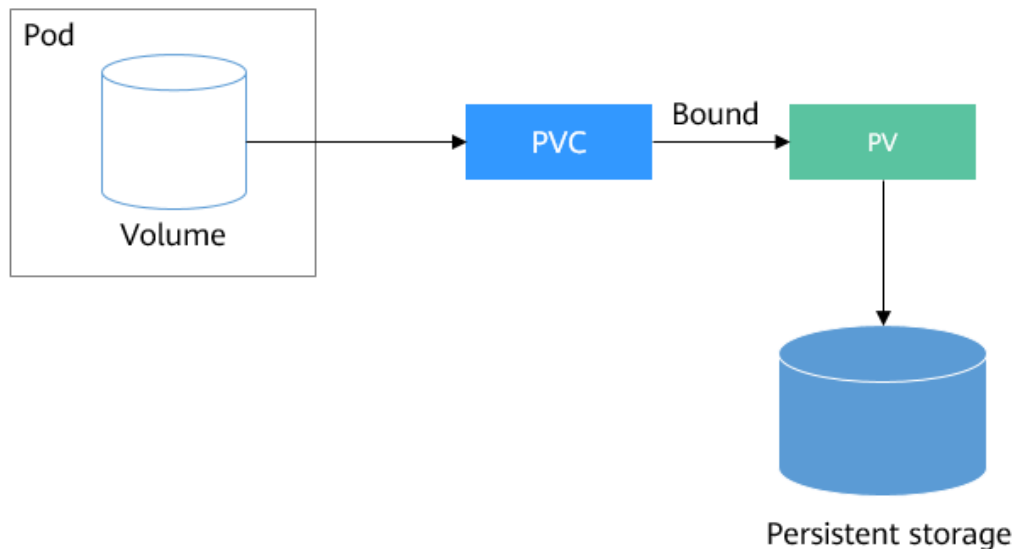
Kubernetes provides PersistentVolumes (PVs) and PersistentVolumeClaims (PVCs) to abstract details of how storage is provided from how it is consumed. You can request specific size of storage when needed, just like pods can request specific levels of resources (CPU and memory).

- 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 12-2 PVC-to-PV binding



CSI

CSI is a standard for container storage interfaces and a storage plug-in 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 12-2 Access modes supported by storage volumes

Storage Type	ReadWriteOnce	ReadWriteMany
EVS	√	×
SFS	×	√

Storage Type	ReadWriteOnce	ReadWriteMany
OBS	×	√
SFS Turbo	×	√

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 12-3 Modes of mounting volumes

Mounting 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 StorageClass 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 volumeClaimTemplates 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, so the associated underlying storage assets from the external infrastructure.
- **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 resources are in the **Released** state and cannot be directly bound to the PVC.

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](#).

12.3 Elastic Volume Service (EVS)

12.3.1 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: number of read/write operations performed by an EVS disk per second
- Throughput: amount of data read from and written into an EVS disk per second
- Read/write I/O latency: minimum interval between two consecutive read/write operations on an EVS disk

Table 12-4 EVS disk performance specifications

Parameter	Ultra-high I/O	High I/O	Common I/O
Max. capacity (GiB)	<ul style="list-style-type: none"> • System disk: 1,024 • Data disk: 32,768 	<ul style="list-style-type: none"> • System disk: 1,024 • Data disk: 32,768 	<ul style="list-style-type: none"> • System disk: 1,024 • Data disk: 32,768
Max. IOPS	50,000	5,000	2,200

Parameter	Ultra-high I/O	High I/O	Common I/O
Max. throughput (MiB/s)	350	150	50
Burst IOPS limit	16,000	5,000	2,200
Disk IOPS	Min. (50,000, 1,800 + 50 x Capacity)	Min. (5,000, 1,800 + 8 x Capacity)	Min. (2,200, 500 + 2 x Capacity)
Disk throughput (MiB/s)	Min. (350, 120 + 0.5 x Capacity)	Min. (150, 100 + 0.15 x Capacity)	50
Single-queue access latency (ms)	1	1-3	5-10
API name	SSD	SAS	SATA

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 to scenarios where the underlying storage is available.
- **Using an EVS Disk Through a Dynamic PV:** dynamic creation mode, where you do not need to create EVS volumes in advance. Instead, specify a StorageClass during PVC creation and an EVS disk and a PV will be automatically created. This mode applies to scenarios where no underlying storage is available.
- **Dynamically Mounting an EVS Disk to a StatefulSet:** Only StatefulSets support 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.

12.3.2 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 to scenarios where the underlying storage is available.

Prerequisites

- You have created a cluster and installed the CSI add-on (**everest**) in the cluster.
- You have created an EVS disk that meets the following requirements:
 - The existing EVS disk cannot be a system disk, DSS disk, or shared disk.
 - The device type of the EVS disk must be **SCSI** (the default device 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.
 - If the EVS disk is encrypted, the key must be available.
 - Only the EVS disks in the enterprise project to which the cluster belongs and the default enterprise project are supported.
 - EVS disks that have partitions or use non-ext4 file systems are not supported.
- If you want to create a cluster using commands, use kubectl to connect to the cluster. For details, see [Connecting to a Cluster Using kubectl](#).

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, create 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 disks attached, 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 disks attached, 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** from the navigation pane, and click the **PersistentVolumeClaims (PVCs)** tab. Click **Create PVC** in the upper right corner. In the dialog box displayed, configure the PVC parameters.

Parameter	Description
PVC Type	In this example, select EVS .
PVC Name	Enter the PVC name, which must be unique in the same namespace.
Creation Method	<ul style="list-style-type: none"> - If underlying storage is available, create a storage volume or use an existing storage volume to statically create a PVC based on whether a PV has been created. - If no underlying storage is available, select Dynamically provision. For details, see Using an EVS Disk Through a Dynamic PV. <p>In this example, select Create new to create a PV and PVC at the same time on the console.</p>

Parameter	Description
PV ^a	Select an existing PV in the cluster. Create a PV in advance. For details, see "Creating a storage volume" in Related Operations . In this example, you do not need to set this parameter.
EVS ^b	Click Select EVS . On the displayed page, select the EVS disk that meets your requirements and click OK .
PV Name ^b	Enter the PV name, which must be unique in the same cluster.
Access Mode ^b	EVS disks support only ReadWriteOnce , indicating that a storage volume can be mounted to one node in read/write mode. For details, see Volume Access Modes .
Reclaim Policy ^b	You can select Delete or Retain to specify the reclaim policy of the underlying storage when the PVC is deleted. For details, see PV Reclaim Policy .

 **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 **PersistentVolumeClaims (PVCs)** and **PersistentVolumes (PVs)** tab pages.

Step 3 Create an application.

1. In the navigation pane on the left, click **Workloads**. 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 12-5](#). For details about other parameters, see [Workloads](#).

Table 12-5 Mounting a storage volume

Parameter	Description
PVC	Select an existing EVS volume. An EVS volume cannot be repeatedly mounted to multiple workloads.

Parameter	Description
Mount Path	<p>Enter a mount path, for example, /tmp.</p> <p>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 / or /var/run; this action may cause container errors. You are advised to 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, causing container startup failures or workload creation failures.</p> <p>NOTICE 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 machine may be damaged.</p>
Subpath	<p>Enter a subpath, for example, tmp, indicating that data in the mount path of the container will be stored in the tmp folder of the volume.</p> <p>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 is used by default.</p>
Permission	<ul style="list-style-type: none"> - Read-only: You can only read the data in the mounted volumes. - Read/Write: You can modify the data volumes mounted to the path. Newly written data is not migrated if the container is migrated, which may cause data loss.

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 cannot be attached to multiple pods in a workload. Otherwise, the pods cannot start properly. Ensure that the number of workload pods is 1 when you attach an EVS disk.

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

(kubectl) Using an Existing EVS Disk

Step 1 Use kubectl to connect to 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 PV is deleted while the
underlying volume is retained.
  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. The value is fixed to ReadWriteOnce for EVS disks.
  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: <your_volume_id> # 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
      everest.io/crypt-key-id: <your_key_id> # (Optional) Encryption key ID. Mandatory for an
encrypted disk.
      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 # Storage class name. The value must be csi-disk for EVS disks.

```

Table 12-6 Key parameters

Parameter	Mandato ry	Description
everest.io/ reclaim-policy: retain-volume- only	No	Optional. Currently, only retain-volume-only 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 retain-volume-only , the associated PV is deleted while the underlying storage volume is retained, when a PVC is deleted.
failure- domain.beta.kub ernetes.io/region	Yes	Region where the cluster is located. For details about the value of region , see Regions and Endpoints .
failure- domain.beta.kub ernetes.io/zone	Yes	AZ where the EVS volume is created. It must be the same as the AZ planned for the workload. For details about the value of zone , see Regions and Endpoints .

Parameter	Mandatory	Description
volumeHandle	Yes	Volume ID of the EVS disk. To obtain the volume ID, log in to the Cloud Server Console . In the navigation pane, choose Elastic Volume Service > Disks . Click the name of the target EVS disk to go to its details page. On the Summary tab page, click the copy button after ID .
everest.io/disk-volume-type	Yes	EVS disk type. All letters are in uppercase. <ul style="list-style-type: none"> - SATA: common I/O - SAS: high I/O - SSD: ultra-high I/O
everest.io/encrypt-key-id	No	Mandatory when the EVS disk is encrypted. Enter the encryption key ID selected during EVS disk creation. To obtain the encryption key ID, log in to the Cloud Server Console . In the navigation pane, choose Elastic Volume Service > Disks . Click the name of the target EVS disk to go to its details page. On the Summary tab page, copy the value of KMS Key ID in the Configuration Information area.
everest.io/enterprise-project-id	No	Optional. Enterprise project ID of the EVS disk. If an enterprise project is specified, specify the same enterprise project when creating a PVC. Otherwise, the PVC cannot be bound to a PV. To obtain the enterprise project ID, log in to the Cloud Server Console . In the navigation pane, choose Elastic Volume Service > Disks . Click the name of the target EVS disk to go to its details page. On the Summary tab page, click the enterprise project in Management Information 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 Delete and Retain reclaim policies are supported. For details, see PV Reclaim Policy. If high data security is required, you are advised to select Retain to prevent data from being deleted by mistake.</p> <p>Delete:</p> <ul style="list-style-type: none"> - If everest.io/reclaim-policy is not specified, both the PV and EVS volume are deleted when a PVC is deleted. - If everest.io/reclaim-policy is set to retain-volume-only, when a PVC is deleted, the PV is deleted but the EVS resources are retained. <p>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 is in the Released status and cannot be bound to the PVC again.</p>
storageClassName	Yes	The storage class name for EVS disks is csi-disk .

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/encrypt-key-id: <your_key_id> # (Optional) Encryption key ID. Mandatory for an
    encrypted disk.
    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 # Storage class type for EVS disks.
volumeName: pv-evs # PV name.
    
```

Table 12-7 Key parameters

Parameter	Mandatory	Description
failure-domain.beta.kubernetes.io/region	Yes	Region where the cluster is located. For details about the value of region , see Regions and Endpoints .
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. For details about the value of zone , see Regions and Endpoints .
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 1 .
storageClassName	Yes	Storage class name, which must be the same as the storage class of the PV in 1 . The storage class name of the EVS volumes is csi-disk .

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 view 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 view 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-evs-0**:

```
kubectl delete pod web-evs-0
```

Expected output:

```
pod "web-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 web-evs-0 -- ls /data
```

Expected output:

```
lost+found
static
```

If the **static** file still exists, the data in the EVS volume can be stored persistently.

----End

Related Operations

You can also perform the operations listed in [Table 12-8](#).

Table 12-8 Related operations

Operation	Description	Procedure
Creating a storage volume (PV)	Create a PV on the CCE console.	<ol style="list-style-type: none"> Choose Storage from the navigation pane, and click the PersistentVolumes (PVs) tab. Click Create Volume in the upper right corner. In the dialog box displayed, configure the parameters. <ul style="list-style-type: none"> Volume Type: Select EVS. EVS: Click Select EVS. On the displayed page, select the EVS disk that meets your requirements and click OK. PV Name: Enter the PV name, which must be unique in the same cluster. Access Mode: EVS disks support only ReadWriteOnce, indicating that a storage volume can be mounted to one node in read/write mode. For details, see Volume Access Modes. Reclaim Policy: Delete or Retain. For details, see PV Reclaim Policy. Click Create.

Operation	Description	Procedure
Expanding the capacity of an EVS disk	Quickly expand the capacity of a mounted EVS disk on the CCE console.	<ol style="list-style-type: none"> 1. Choose Storage from the navigation pane, and click the PersistentVolumeClaims (PVCs) tab. Click More in the Operation column of the target PVC and select Scale-out. 2. Enter the capacity to be added and click OK.
Viewing events	You can 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"> 1. Choose Storage from the navigation pane, and click the PersistentVolumeClaims (PVCs) or PersistentVolumes (PVs) tab. 2. Click View Events in the Operation column of the target PVC or PV to view events generated within one hour (event data is retained for one hour).
Viewing a YAML file	You can view, copy, and download the YAML files of a PVC or PV.	<ol style="list-style-type: none"> 1. Choose Storage from the navigation pane, and click the PersistentVolumeClaims (PVCs) or PersistentVolumes (PVs) tab. 2. Click View YAML in the Operation column of the target PVC or PV to view or download the YAML.

12.3.3 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 CSI add-on ([everest](#)) in the cluster.
- If you want to create a cluster using commands, use `kubectl` to connect to the cluster. For details, see [Connecting to a Cluster Using kubectl](#).

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, create 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 disks attached, 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 disks attached, 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.

- Choose **Storage** from the navigation pane, and click the **PersistentVolumeClaims (PVCs)** tab. Click **Create PVC** in the upper right corner. In the dialog box displayed, configure the PVC parameters.

Parameter	Description
PVC Type	In this example, select EVS .
PVC Name	Enter the PVC name, which must be unique in the same namespace.
Creation Method	<ul style="list-style-type: none"> If no underlying storage is available, select Dynamically provision to create a PVC, PV, and underlying storage on the console in cascading mode. If underlying storage is available, create a storage volume or use an existing storage volume to statically create a PVC based on whether a PV has been created. For details, see Using an Existing EVS Disk Through a Static PV. <p>In this example, select Dynamically provision.</p>
Storage Classes	The storage class for EVS disks is csi-disk .
AZ	Select the AZ of the EVS disk. The AZ must be the same as that of the cluster node. NOTE 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.
Access Mode	EVS disks support only ReadWriteOnce , indicating that a storage volume can be mounted to one node in read/write mode. For details, see Volume Access Modes .
Capacity (GiB)	Capacity of the requested storage volume.
Encryption	You can select Encryption and an encryption key to encrypt underlying storage. Before using the encryption function, check whether the region where the EVS disk is located supports disk encryption.

Parameter	Description
Enterprise Project	Supported enterprise projects: default, the one to which the cluster belongs, or the one specified by the storage class.

2. Click **Create**.

You can choose **Storage** in the navigation pane and view the created PVC and PV on the **PersistentVolumeClaims (PVCs)** and **PersistentVolumes (PVs)** tab pages.

Step 3 Create an application.

1. In the navigation pane on the left, click **Workloads**. 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 12-9](#). For details about other parameters, see [Workloads](#).

Table 12-9 Mounting a storage volume

Parameter	Description
PVC	Select an existing EVS volume. An EVS volume cannot be repeatedly mounted to multiple workloads.
Mount Path	Enter a mount path, for example, /tmp . 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 / or /var/run ; this action may cause container errors. You are advised to 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, causing container startup failures or workload creation failures. NOTICE 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 machine may be damaged.
Subpath	Enter a subpath, for example, tmp , indicating that data in the mount path of the container will be stored in the tmp folder of the volume. 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 is used by default.

Parameter	Description
Permission	<ul style="list-style-type: none"> – Read-only: You can only read the data in the mounted volumes. – Read/Write: You can modify the data volumes mounted to the path. Newly written data is not migrated if the container is migrated, which may cause data loss.

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 cannot be attached to multiple pods in a workload. Otherwise, the pods cannot start properly. Ensure that the number of workload pods is 1 when you attach an EVS disk.

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

(kubectl) Automatically Creating an EVS Disk

Step 1 Use kubectl to connect to 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/crypt-key-id: <your_key_id> # (Optional) Encryption key ID. Mandatory for an
    encrypted disk.
    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 # Storage class type for EVS disks.
```

Table 12-10 Key parameters

Parameter	Mandatory	Description
failure-domain.beta.kubernetes.io/region	Yes	Region where the cluster is located. For details about the value of region , see Regions and Endpoints .
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. For details about the value of zone , see Regions and Endpoints .
everest.io/disk-volume-type	Yes	EVS disk type. All letters are in uppercase. <ul style="list-style-type: none"> - SATA: common I/O - SAS: high I/O - SSD: ultra-high I/O
everest.io/encryption-key-id	No	This parameter is mandatory when an EVS disk is encrypted. Enter the encryption key ID selected during EVS disk creation. You can use a custom key or the default key named evs/default . To obtain a key ID, log in to the DEW console, locate the key to be encrypted, and copy the key ID.
everest.io/enterprise-project-id	No	Optional. Enterprise project ID of the EVS disk. If an enterprise project is specified, specify the same enterprise project when creating a PVC. Otherwise, the PVC cannot be bound to a PV. To obtain the enterprise project ID, log in to the Cloud Server Console . In the navigation pane, choose Elastic Volume Service > Disks . Click the name of the target EVS disk to go to its details page. On the Summary tab page, click the enterprise project in Management Information 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 1 to 32768 .
storageClassName	Yes	The storage class name of the EVS volumes is csi-disk .

2. Run the following command to create a PVC:

```
kubectl apply -f pvc-eva-auto.yaml
```

Step 3 Create an application.

1. Create a file named **web-eva-auto.yaml**. In this example, the EVS volume is mounted to the **/data** path.

```
apiVersion: apps/v1
kind: StatefulSet
metadata:
  name: web-eva-auto
  namespace: default
spec:
  replicas: 1
  selector:
    matchLabels:
      app: web-eva-auto
  serviceName: web-eva-auto # Headless Service name.
  template:
    metadata:
      labels:
        app: web-eva-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-eva-auto # Name of the created PVC.
---
apiVersion: v1
kind: Service
metadata:
  name: web-eva-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-evs-auto
```

Expected output:

```
web-evs-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-evs-auto-0 -- df | grep data
```

Expected output:

```
/dev/sdc          10255636   36888 10202364   0% /data
```

3. Run the following command to view the files in the **/data** path:

```
kubectl exec web-evs-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-evs-auto-0 -- touch /data/static
```

Step 3 Run the following command to view the files in the **/data** path:

```
kubectl exec web-evs-auto-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 web-evs-auto-0
```

Expected output:

```
pod "web-evs-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
```

If the **static** file still exists, the data in the EVS volume can be stored persistently.

----End

Related Operations

You can also perform the operations listed in [Table 12-11](#).

Table 12-11 Related operations

Operation	Description	Procedure
Expanding the capacity of an EVS disk	Quickly expand the capacity of a mounted EVS disk on the CCE console.	<ol style="list-style-type: none"> 1. Choose Storage from the navigation pane, and click the PersistentVolumeClaims (PVCs) tab. Click More in the Operation column of the target PVC and select Scale-out. 2. Enter the capacity to be added and click OK.
Viewing events	You can 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"> 1. Choose Storage from the navigation pane, and click the PersistentVolumeClaims (PVCs) or PersistentVolumes (PVs) tab. 2. Click View Events in the Operation column of the target PVC or PV to view events generated within one hour (event data is retained for one hour).
Viewing a YAML file	You can view, copy, and download the YAML files of a PVC or PV.	<ol style="list-style-type: none"> 1. Choose Storage from the navigation pane, and click the PersistentVolumeClaims (PVCs) or PersistentVolumes (PVs) tab. 2. Click View YAML in the Operation column of the target PVC or PV to view or download the YAML.

12.3.4 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 the storage class to dynamically provision PVs. 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.

Prerequisites

- You have created a cluster and installed the CSI add-on (**everest**) in the cluster.
- If you want to create a cluster using commands, use kubectl to connect to the cluster. For details, see **Connecting to a Cluster Using kubectl**.

(Console) Dynamically Mounting an EVS Disk

- Step 1** Log in to the CCE console and click the cluster name to access the cluster console.
- Step 2** In the navigation pane on the left, click **Workloads**. 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 (VTC)**.
- Step 4** Click **Create PVC**. In the dialog box displayed, configure the PVC parameters.

Click **Create**.

Parameter	Description
PVC Type	In this example, select EVS .
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><Custom PVC name>-<Serial number></i> , for example, <i>example-0</i> .
Creation Method	You can select Dynamically provision to create a PVC, PV, and underlying storage on the console in cascading mode.
Storage Classes	The storage class for EVS disks is csi-disk .
AZ	Select the AZ of the EVS disk. The AZ must be the same as that of the cluster node. NOTE 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.
Access Mode	EVS disks support only ReadWriteOnce , indicating that a storage volume can be mounted to one node in read/write mode. For details, see Volume Access Modes .
Capacity (GiB)	Capacity of the requested storage volume.
Encryption	You can select Encryption and an encryption key to encrypt underlying storage. Only EVS disks and SFS file systems support encryption.
Enterprise Project	Supported enterprise projects: default, the one to which the cluster belongs, or the one specified by the storage class.

- Step 5** Enter the path to which the volume is mounted.

Table 12-12 Mounting a storage volume

Parameter	Description
Mount Path	<p>Enter a mount path, for example, /tmp.</p> <p>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 / or /var/run. Otherwise, errors will occur in containers. 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, causing container startup failures or workload creation failures.</p> <p>NOTICE 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 machine may be damaged.</p>
Subpath	<p>Enter a subpath, for example, tmp, indicating that data in the mount path of the container will be stored in the tmp folder of the volume.</p> <p>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 is used by default.</p>
Permission	<ul style="list-style-type: none"> • Read-only: You can only read the data in the mounted volumes. • Read/Write: You can modify the data volumes mounted to the path. Newly written data is not migrated if the container is migrated, which may cause data loss.

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

(kubectl) Using an Existing EVS Disk

Step 1 Use kubectl to connect to 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-eva
  template:
    metadata:
      labels:
        app: statefulset-eva
    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-eva # 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/crypt-key-id: <your_key_id> # (Optional) Encryption key ID. Mandatory for an encrypted
disk.
              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 # Storage class type for EVS disks.
---
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 12-13 Key parameters

Parameter	Mandatory	Description
failure-domain.beta.kubernetes.io/region	Yes	Region where the cluster is located. For details about the value of region , see Regions and Endpoints .
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. For details about the value of zone , see Regions and Endpoints .
everest.io/disk-volume-type	Yes	EVS disk type. All letters are in uppercase. <ul style="list-style-type: none"> • SATA: common I/O • SAS: high I/O • SSD: ultra-high I/O
everest.io/encryption-key-id	No	Mandatory when the EVS disk is encrypted. Enter the encryption key ID selected during EVS disk creation. To obtain the encryption key ID, log in to the Cloud Server Console . In the navigation pane, choose Elastic Volume Service > Disks . Click the name of the target EVS disk to go to its details page. On the Summary tab page, copy the value of KMS Key ID in the Configuration Information area.
everest.io/enterprise-project-id	No	Optional. Enterprise project ID of the EVS disk. If an enterprise project is specified, specify the same enterprise project when creating a PVC. Otherwise, the PVC cannot be bound to a PV. To obtain the enterprise project ID, log in to the Cloud Server Console . In the navigation pane, choose Elastic Volume Service > Disks . Click the name of the target EVS disk to go to its details page. On the Summary tab page, click the enterprise project in Management Information 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 1 to 32768 .
storageClassName	Yes	The storage class name for EVS disks is csi-disk .

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 view 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 view 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
```

If the **static** file still exists, the data in the EVS volume can be stored persistently.

----End

Related Operations

You can also perform the operations listed in [Table 12-14](#).

Table 12-14 Related operations

Operation	Description	Procedure
Expanding the capacity of an EVS disk	Quickly expand the capacity of a mounted EVS disk on the CCE console.	<ol style="list-style-type: none"> 1. Choose Storage from the navigation pane, and click the PersistentVolumeClaims (PVCs) tab. Click More in the Operation column of the target PVC and select Scale-out. 2. Enter the capacity to be added and click OK.
Viewing events	You can 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"> 1. Choose Storage from the navigation pane, and click the PersistentVolumeClaims (PVCs) or PersistentVolumes (PVs) tab. 2. Click View Events in the Operation column of the target PVC or PV to view events generated within one hour (event data is retained for one hour).
Viewing a YAML file	You can view, copy, and download the YAML files of a PVC or PV.	<ol style="list-style-type: none"> 1. Choose Storage from the navigation pane, and click the PersistentVolumeClaims (PVCs) or PersistentVolumes (PVs) tab. 2. Click View YAML in the Operation column of the target PVC or PV to view or download the YAML.

12.3.5 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 (SCSI or VBD), data encryption, 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 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`.
- Snapshot data of encrypted disks is stored encrypted, and that of non-encrypted disks is stored non-encrypted.

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 and go to the cluster console. 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 an EVS PVC.

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 # Snapshot name
  namespace: default
spec:
  source:
    persistentVolumeClaimName: pvc-eva-test # PVC name. Only an EVS PVC can be selected.
    volumeSnapshotClassName: csi-disk-snapclass
```

Using a Snapshot to Create a PVC

The disk type, encryption setting, 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 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
```

12.4 Scalable File Service (SFS)

12.4.1 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 read/write 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

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 to scenarios where the underlying storage is available.
- **Using an SFS File System Through a Dynamic PV:** dynamic creation mode, where you do not need to create SFS volumes in advance. Instead, specify a StorageClass during PVC creation and an SFS volume and a PV will be automatically created. This mode applies to scenarios where no underlying storage is available.

12.4.2 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 and implement data persistence and sharing in workloads.

Prerequisites

- You have created a cluster and installed the CSI add-on ([everest](#)) in the cluster.
- If you want to create a cluster using commands, use `kubectl` to connect to the cluster. 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.

Constraints

- Multiple PVs can use the same SFS or SFS Turbo file system with the following restrictions:
 - If multiple PVCs/PVs use the same underlying SFS or SFS Turbo file system, when you attempt to mount these PVCs/PVs to the same pod, all PVCs cannot be mounted to the pod and the pod startup fails. This is because the `volumeHandle` values of these PVs are the same.
 - 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.

(kubectl) Using an Existing SFS File System

Step 1 Use `kubectl` to connect to the cluster.

Step 2 Create a PV.

1. Create the `pv-sfs.yaml` file.

SFS Capacity-Oriented:

```
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: pv-sfs # PV name.
spec:
  accessModes:
    - ReadWriteMany # Access mode. The value must be ReadWriteMany for SFS.
  capacity:
    storage: 1Gi # SFS volume capacity.
  csi:
    driver: disk.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 # Storage class name. csi-nas indicates that SFS Capacity-
  Oriented is used.
  mountOptions: [] # Mount options.
```

Table 12-15 Key parameters

Parameter	Mandatory	Description
everest.io/reclaim-policy:retain-volume-only	No	Optional. Currently, only retain-volume-only 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 retain-volume-only , the associated PV is deleted while the underlying storage volume is retained, when a PVC is deleted.
volumeHandle	Yes	<ul style="list-style-type: none"> If an SFS Capacity-Oriented volume is used, enter the volume ID. Log in to the console, choose Service List > Storage > Scalable File Service, and select SFS Turbo. In the list, click the name of the target SFS file system. On the details page, copy the content following ID.
everest.io/share-export-location	Yes	Shared path of the file system. <ul style="list-style-type: none"> For an SFS Capacity-Oriented file system, log in to the console, choose Service List > Storage > Scalable File Service, and obtain the shared path from the Mount Address column.
mountOptions	Yes	Mount options. If not specified, the following configurations are used by default. For details, see Configuring SFS Volume Mount Options . mountOptions: <ul style="list-style-type: none"> - vers=3 - timeo=600 - nolock - hard

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 Delete and Retain reclaim policies are supported. For details, see PV Reclaim Policy. If multiple PVs use the same SFS volume, use Retain to avoid cascading deletion of underlying volumes.</p> <p>Delete:</p> <ul style="list-style-type: none"> - If everest.io/reclaim-policy is not specified, both the PV and SFS volume are deleted when a PVC is deleted. - If everest.io/reclaim-policy is set to retain-volume-only set, when a PVC is deleted, the PV is deleted but the SFS volume resources are retained. <p>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 is in the Released status and cannot be bound to the PVC again.</p>
storage	Yes	<p>Requested capacity in the PVC, in Gi.</p> <p>For SFS, this field is used only for verification (cannot be empty or 0). Its value is fixed at 1, and any value you set does not take effect for SFS file systems.</p>

2. Run the following command to create a PV:

```
kubectl apply -f pv-sfs.yaml
```

Step 3 Create a PVC.

1. 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 # Storage class name, which must be the same as the PV's storage class.
  volumeName: pv-sfs # PV name.
```

Table 12-16 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.
volumeName	Yes	PV name, which must be the same as the PV name in 1 .

2. Run the following command to create a PVC:

```
kubectl apply -f pvc-sfs.yaml
```

Step 4 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 # Name of the created PVC.
```

2. Run the following command to create an application 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 applications 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 view 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
```

If the **static** file still exists, the data 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 another pods, the two pods share the same volume.

----End

Related Operations

You can also perform the operations listed in [Table 12-17](#).

Table 12-17 Related operations

Operation	Description	Procedure
Viewing events	You can 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"> 1. Choose Storage from the navigation pane, and click the PersistentVolumeClaims (PVCs) or PersistentVolumes (PVs) tab. 2. Click View Events in the Operation column of the target PVC or PV to view events generated within one hour (event data is retained for one hour).
Viewing a YAML file	You can view, copy, and download the YAML files of a PVC or PV.	<ol style="list-style-type: none"> 1. Choose Storage from the navigation pane, and click the PersistentVolumeClaims (PVCs) or PersistentVolumes (PVs) tab. 2. Click View YAML in the Operation column of the target PVC or PV to view or download the YAML.

12.4.3 Using an SFS File System Through a Dynamic PV

This section describes how to use storage classes to dynamically create PVs and PVCs and implement data persistence and sharing in workloads.

(kubectl) Automatically Creating an SFS File System

Step 1 Use **kubectl** to connect to 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
  annotations:
    everest.io/crypt-key-id: <your_key_id> # (Optional) ID of the key for encrypting file systems
    everest.io/crypt-alias: sfs/default # (Optional) Key name. Mandatory for encrypting volumes
    everest.io/crypt-domain-id: <your_domain_id> # (Optional) ID of the tenant to which an
encrypted volume belongs. Mandatory for encrypting volumes
spec:
  accessModes:
    - ReadWriteMany # The value must be ReadWriteMany for SFS.
  resources:
    requests:
      storage: 1Gi # SFS volume capacity.
      storageClassName: csi-nas # The storage class type is SFS.

```

Table 12-18 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 0). Its value is fixed at 1 , and any value you set does not take effect for SFS file systems.
everest.io/crypt-key-id	No	This parameter is mandatory when an SFS system is encrypted. Enter the encryption key ID selected during SFS system creation. You can use a custom key or the default key named sfs/default . To obtain a key ID, log in to the DEW console, locate the key to be encrypted, and copy the key ID.
everest.io/crypt-alias	No	Key name, which is mandatory when you create an encrypted volume. To obtain a key name, log in to the DEW console, locate the key to be encrypted, and copy the key name.
everest.io/crypt-domain-id	No	ID of the tenant to which the encrypted volume belongs. This parameter is mandatory for creating an encrypted volume. To obtain a tenant ID, hover the cursor over the username in the upper right corner of the ECS console, choose My Credentials , and copy the account ID.

- 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 an application 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 applications 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 view 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
```

If the **static** file still exists, the data 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 another pods, the two pods share the same volume.

----End

Related Operations

You can also perform the operations listed in [Table 12-19](#).

Table 12-19 Related operations

Operation	Description	Procedure
Viewing events	You can 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"> 1. Choose Storage from the navigation pane, and click the PersistentVolumeClaims (PVCs) or PersistentVolumes (PVs) tab. 2. Click View Events in the Operation column of the target PVC or PV to view events generated within one hour (event data is retained for one hour).
Viewing a YAML file	You can view, copy, and download the YAML files of a PVC or PV.	<ol style="list-style-type: none"> 1. Choose Storage from the navigation pane, and click the PersistentVolumeClaims (PVCs) or PersistentVolumes (PVs) tab. 2. Click View YAML in the Operation column of the target PVC or PV to view or download the YAML.

12.4.4 Configuring SFS Volume Mount Options

This section describes how to configure SFS 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 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.

SFS Volume Mount Options

The everest add-on in CCE presets the options described in [Table 12-20](#) for mounting SFS volumes.

Table 12-20 SFS volume mount options

Parameter	Value	Description
keep-original-ownership	Leave it 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"> By default, this option is not added, and the mount point ownership is root:root when SFS is mounted. If this option is added, the original ownership of the file system is retained when SFS is mounted.
vers	3	File system version. Currently, only NFSv3 is supported. Value: 3
nolock	Leave it blank.	Whether to lock files on the server using the NLM protocol. If nolock 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: 600
hard/soft	Leave it blank.	Mount mode. <ul style="list-style-type: none"> hard: If the NFS request times out, the client keeps resending the request until the request is successful. soft: If the NFS request times out, the client returns an error to the invoking program. The default value is hard .

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 connect to 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 PV is deleted while the underlying
    volume is retained.
  name: pv-sfs
spec:
  accessModes:
```

```
- ReadOnlyMany # Access mode. The value must be ReadWriteMany for SFS.
capacity:
storage: 1Gi # SFS volume capacity.
csi:
driver: disk.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 # Storage class 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 are set successfully.

```
<Your shared path> on /data type nfs
```

```
(rw,relatime,vers=3,rsize=1048576,wsize=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

Setting 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 connect to the cluster. For details, see [Connecting to a Cluster Using kubectl](#).

Step 2 Create a customized 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 are set successfully.

```
<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

12.5 SFS Turbo File Systems

12.5.1 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 a 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

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 and Mounting Subdirectories of an SFS Turbo File System:** 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.

12.5.2 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 and implement data persistence and sharing in workloads.

Prerequisites

- You have created a cluster and installed the CSI add-on ([everest](#)) in the cluster.
- If you want to create a cluster using commands, use `kubectl` to connect to the cluster. 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.

Constraints

- Multiple PVs can use the same SFS or SFS Turbo file system with the following restrictions:
 - If multiple PVCs/PVs use the same underlying SFS or SFS Turbo file system, when you attempt to mount these PVCs/PVs to the same pod, all PVCs cannot be mounted to the pod and the pod startup fails. This is because the `volumeHandle` values of these PVs are the same.
 - 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 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** from the navigation pane, and click the **PersistentVolumeClaims (PVCs)** tab. Click **Create PVC** in the upper right corner. In the dialog box displayed, configure the PVC parameters.

Parameter	Description
PVC Type	In this section, select SFS Turbo .
PVC Name	Enter the PVC name, which must be unique in the same namespace.
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 Create new to create a PV and PVC at the same time on the console.
PV ^a	Select an existing PV volume in the cluster. Create a PV in advance. For details, see "Creating a storage volume" in Related Operations . You do not need to specify this parameter in this example.
SFS Turbo ^b	Click Select SFS Turbo . On the displayed page, select the SFS Turbo file system that meets your requirements and click OK .
PV Name ^b	Enter the PV name, which must be unique in the same cluster.
Access Mode ^b	SFS Turbo volumes support only ReadWriteMany , indicating that a storage volume can be mounted to multiple nodes in read/write mode. For details, see Volume Access Modes .
Reclaim Policy ^b	Only Retain is supported, indicating that the PV is not deleted when the PVC is deleted. For details, see PV Reclaim Policy .
Mount Options ^b	Enter the mounting parameter key-value pairs. For details, see Configuring SFS Turbo Mount Options .

 **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 **PersistentVolumeClaims (PVCs)** and **PersistentVolumes (PVs)** tab pages.

Step 3 Create an application.

1. In the navigation pane on the left, click **Workloads**. 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 12-21](#). For details about other parameters, see [Workloads](#).

Table 12-21 Mounting a storage volume

Parameter	Description
PVC	Select an existing SFS Turbo volume.
Mount Path	<p>Enter a mount path, for example, /tmp.</p> <p>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 / or /var/run. 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, causing container startup failures or workload creation failures.</p> <p>NOTICE 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 machine may be damaged.</p>
Subpath	<p>Enter a subpath, for example, tmp, indicating that data in the mount path of the container will be stored in the tmp folder of the volume.</p> <p>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 is used by default.</p>
Permission	<ul style="list-style-type: none"> – Read-only: You can only read the data in the mounted volumes. – Read/Write: You can modify the data volumes mounted to the path. Newly written data is not migrated if the container is migrated, which may cause data loss.

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

(kubectl) Using an Existing SFS File System

Step 1 Use kubectl to connect to the cluster.

Step 2 Create a PV.

1. Create the `pv-sfsturbo.yaml` file.

```
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 # Storage class name of the SFS Turbo file system.
  mountOptions: [] # Mount options.
```

Table 12-22 Key parameters

Parameter	Mandatory	Description
volumeHandle	Yes	SFS Turbo volume ID. How to obtain: Log in to the console, choose Service List > Storage > Scalable File Service , and select SFS Turbo . In the list, click the name of the target SFS Turbo volume. On the details page, copy the content following ID .
everest.io/share-export-location	Yes	Shared path of the SFS Turbo volume. Log in to the console, choose Service List > Storage > Scalable File Service , and select SFS Turbo . You can obtain the shared path of the file system from the Mount Address column.

Parameter	Mandatory	Description
everest.io/enterprise-project-id	No	Project ID of the SFS Turbo volume. How to obtain: On the SFS console, click SFS Turbo in the left navigation pane. Click the name of the SFS Turbo file system to interconnect. On the Basic Info 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 Configuring SFS Turbo Mount Options . 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. Only the Retain reclaim policy is supported. For details, see Verifying Data Persistence and Sharing . 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 is in the Released status and cannot be bound to the PVC again.
storage	Yes	Requested capacity in the PVC, in Gi.
storageClassName	Yes	The storage class name of SFS Turbo volumes is csi-sfsturbo .

- 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 # Storage class of the SFS Turbo volume, which must be the
same as that of the PV.
volumeName: pv-sfsturbo # PV name.
    
```

Table 12-23 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 SFS Turbo in the left navigation pane. Click the name of the SFS Turbo file system to interconnect. On the Basic Info 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	Storage class name, which must be the same as the storage class of the PV in 1 . The storage class name of SFS Turbo volumes is csi-sfsturbo .
volumeName	Yes	PV name, which must be the same as the PV name in 1 .

- 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 an application 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 applications 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 view 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
```


If the **static** file still exists, the data 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 another pods, the two pods share the same volume.

----End

Related Operations

You can also perform the operations listed in [Table 12-24](#).

Table 12-24 Related operations

Operation	Description	Procedure
Creating a storage volume (PV)	Create a PV on the CCE console.	<ol style="list-style-type: none"> Choose Storage from the navigation pane, and click the PersistentVolumes (PVs) tab. Click Create Volume in the upper right corner. In the dialog box displayed, configure the parameters. <ul style="list-style-type: none"> Volume Type: Select SFS Turbo. SFS Turbo: Click Select SFS Turbo. On the page displayed, select the SFS Turbo volume that meets the requirements and click OK. PV Name: Enter the PV name, which must be unique in the same cluster. Access Mode: SFS volumes support only ReadWriteMany, indicating that a storage volume can be mounted to multiple nodes in read/write mode. For details, see Volume Access Modes. Reclaim Policy: Only Retain is supported. For details, see PV Reclaim Policy. Mount Options: Enter the mounting parameter key-value pairs. For details, see Configuring SFS Turbo Mount Options. Click Create.
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"> Choose Storage from the navigation pane, and click the PersistentVolumeClaims (PVCs) tab. Click More in the Operation column of the target PVC and select Scale-out. Enter the capacity to be added and click OK.
Viewing events	You can 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"> Choose Storage from the navigation pane, and click the PersistentVolumeClaims (PVCs) or PersistentVolumes (PVs) tab. Click View Events in the Operation column of the target PVC or PV to view events generated within one hour (event data is retained for one hour).

Operation	Description	Procedure
Viewing a YAML file	You can view, copy, and download the YAML files of a PVC or PV.	<ol style="list-style-type: none"> 1. Choose Storage from the navigation pane, and click the PersistentVolumeClaims (PVCs) or PersistentVolumes (PVs) tab. 2. Click View YAML in the Operation column of the target PVC or PV to view or download the YAML.

12.5.3 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 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.

SFS Turbo Mount Options

The everest add-on in CCE presets the options described in [Table 12-25](#) for mounting SFS Turbo volumes.

Table 12-25 SFS Turbo mount options

Parameter	Value	Description
vers	3	File system version. Currently, only NFSv3 is supported. Value: 3
nolock	Leave it blank.	Whether to lock files on the server using the NLM protocol. If nolock 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: 600
hard/soft	Leave it blank.	Mount mode. <ul style="list-style-type: none"> • hard: If the NFS request times out, the client keeps resending the request until the request is successful. • soft: If the NFS request times out, the client returns an error to the invoking program. The default value is hard .

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 connect to 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 # SFS Turbo storage class 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 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
```

Command output:

```
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,noresvport,proto=tcp,
```

```
timeo=600,retrans=2,sec=sys,mountaddr=*. *. *. *. *,mountvers=3,mountport=20048,mountproto=tc  
p,local_lock=all,addr=*. *. *. *. *)
```

----End

12.5.4 Dynamically Creating and Mounting Subdirectories of an SFS Turbo File System

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.
- 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.
- Nodes running Ubuntu OS do not support this function if they use the Docker container engine.

Creating an SFS Turbo Volume of the subpath Type

⚠ CAUTION

Do not expand, disassociate, or delete a **subpath** volume.

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.
 - More options are available in everest 1.2.8 or a later version. **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 the subpath mode, it is useless to specify this parameter. The storage capacity is limited by the total capacity of the SFS Turbo file system. If the total capacity of the SFS Turbo file system is insufficient, expand the capacity on the SFS Turbo page in a timely manner.

**Step 5** Run `kubectl create -f sfs-turbo-test.yaml`.

----End

 **NOTE**

It is meaningless to conduct capacity expansion on an SFS Turbo volume created in the subpath mode. This operation does not expand the capacity of the SFS Turbo file system. Ensure that the total capacity of the SFS Turbo file system is not used up.

## Creating a Deployment and Mounting an Existing Volume to the Deployment

**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 Deployment.
- **image**: specifies the image used by the Deployment.
- **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 StatefulSet.
- **image**: specifies the image used by the StatefulSet.
- **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 are mapped to each other.
- **storageClassName**: indicates the name of the StorageClass.

**Step 2** Create the StatefulSet.

```
kubectl create -f statefulset-test.yaml
```

```
----End
```

## 12.6 Object Storage Service (OBS)

### 12.6.1 Overview

#### Introduction

Object Storage Service (OBS) provides massive, secure, and cost-effective data storage capabilities 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 (read/write 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.

- **Parallel File System (PFS, recommended):** It is an optimized high-performance file system provided by OBS. It provides millisecond-level access latency, TB/s-level bandwidth, and million-level IOPS, and can quickly process HPC workloads. PFS outperforms OBS buckets.
- **Object bucket (not recommended):**
  - **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.

## 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 to scenarios where the underlying storage is available.
- **Using an OBS Bucket Through a Dynamic PV:** dynamic creation mode, where you do not need to create OBS volumes in advance. Instead, specify a StorageClass during PVC creation and an OBS volume and a PV will be automatically created. This mode applies to scenarios where no underlying storage is available.

### 12.6.2 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 and implement data persistence and sharing in workloads.

## Prerequisites

- You have created a cluster and installed the CSI add-on ([everest](#)) in the cluster.
- If you want to create a cluster using commands, use `kubectl` to connect to the cluster. For details, see [Connecting to a Cluster Using kubectl](#).
- You have created an OBS bucket. An OBS bucket of the parallel file system type can be selected only when it is in the same region as the cluster.

## Constraints

- When parallel file systems and object buckets are used, the group and permission of the mount point cannot be modified.
- CCE allows you to use OBS parallel file systems by calling the OBS SDK or mounting a PVC through the `obsfs` tool provided by OBS. Each time an OBS parallel file system is mounted, an `obsfs` resident process is generated, as shown in the following figure.

Figure 12-3 `obsfs` resident process

```

k8s@k8s-master-1:~$ ps -ef | grep obsfs
root 2928 0.0 0.1 32928 4096 ? Ss 11:00 0:00 /usr/bin/obsfs pvc-e178f8ab-3307-4814-9a23-fbaa5593cf1 /mnt/paas/kubernetes/kubelst/pods/6805502-30ac-4104-b004-bb3d3168-d01/volumes/kubernetes.io~csi/pvc-e178f8ab-3307-4814-9a23-fbaa5593cf1/mount --url=https://obs.cn-north-7.obs.amazonaws.com.cn:443 --endpoint=cn-north-7 --passed_file=/opt/everest-host-connector/161793776390200339_obsfsprcd/pvc-e178f8ab-3307-4814-9a23-fbaa5593cf1 --a
k8s@k8s-master-1:~$ ps -ef | grep obsfs
k8s@k8s-master-1:~$ ps -ef | grep obsfs

```

Reserve 1 GiB of memory for each `obsfs` process. For example, for a node with 4 vCPUs and 8 GiB of memory, the `obsfs` parallel file system should be mounted to **no more than** eight pods.

### NOTE

An `obsfs` resident process runs on a node. If the consumed memory exceeds the upper limit of the node, the node malfunctions. On a node with 4 vCPUs and 8 GiB of memory, if more than 100 pods are mounted to parallel file systems, the node will be unavailable. Control the number of pods mounted to parallel file systems on a single node.

- Multiple PVs can use the same OBS storage volume with the following restrictions:
  - If multiple PVCs/PVs use the same underlying object storage volume, when you attempt to mount the volume to the same pod, the operation will fail because the `volumeHandle` values of these PVs are the same.
  - 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 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** from the navigation pane, and click the **PersistentVolumeClaims (PVCs)** tab. Click **Create PVC** in the upper right corner. In the dialog box displayed, configure the PVC parameters.

| Parameter                   | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
|-----------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| PVC Type                    | In this section, select <b>OBS</b> .                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| PVC Name                    | Enter the PVC name, which must be unique in the same namespace.                                                                                                                                                                                                                                                                                                                                                                                                                                 |
| Creation Method             | <ul style="list-style-type: none"> <li>- If underlying storage is available, create a storage volume or use an existing storage volume to statically create a PVC based on whether a PV has been created.</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 a PV and PVC at the same time on the console.</p> |
| PV <sup>a</sup>             | <p>Select an existing PV volume in the cluster. Create a PV in advance. For details, 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>            | <p>Click <b>Select OBS</b>. On the displayed page, select the OBS bucket that meets your requirements and click <b>OK</b>.</p> <p><b>NOTE</b><br/>Currently, only parallel file systems are supported.</p>                                                                                                                                                                                                                                                                                      |
| 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><br/>If multiple PVs use the same OBS volume, use <b>Retain</b> to avoid cascading deletion of underlying volumes.</p>                                                                                                                                                                  |

| Parameter                  | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
|----------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Secret <sup>b</sup>        | <p><b>Custom:</b> 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> |
| 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 **PersistentVolumeClaims (PVCs)** and **PersistentVolumes (PVs)** tab pages.

**Step 3** Create an application.

1. In the navigation pane on the left, click **Workloads**. 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 12-26](#). For details about other parameters, see [Workloads](#).

**Table 12-26** Mounting a storage volume

| Parameter | Description                               |
|-----------|-------------------------------------------|
| PVC       | Select an existing object storage volume. |

| Parameter  | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
|------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Mount Path | <p>Enter a mount path, for example, <b>/tmp</b>.</p> <p>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>. 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, causing container startup failures or workload creation failures.</p> <p><b>NOTICE</b><br/>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 machine may be damaged.</p> |
| Subpath    | <p>Enter a subpath, for example, <b>tmp</b>, indicating that data in the mount path of the container will be stored in the <b>tmp</b> folder of the volume.</p> <p>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 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 is not 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 [PV Reclaim Policy](#).

----End

## (kubectl) Using an Existing OBS Bucket

**Step 1** Use kubectl to connect to 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 PV is deleted while the
underlying volume is retained.
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.
 driver: obs.csi.everest.io # 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 # Storage class name.
 mountOptions: [] # Mount options.
```

**Table 12-27** Key parameters

| Parameter                                     | Mandatory | Description                                                                                                                                                                                                                                                                                                                                                                            |
|-----------------------------------------------|-----------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| everest.io/reclaim-policy: retain-volume-only | No        | Optional.<br>Currently, only <b>retain-volume-only</b> is supported.<br>This field 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> .<br>– <b>obsfs</b> : Parallel file system, which is mounted using obsfs (recommended).<br>– <b>s3fs</b> : Object bucket, which is mounted using s3fs.                                                                                                                                                                     |
| volumeHandle                                  | Yes       | OBS volume name.                                                                                                                                                                                                                                                                                                                                                                       |
| everest.io/obs-volume-type                    | Yes       | OBS storage class.<br>– If <b>fsType</b> is set to <b>s3fs</b> , <b>STANDARD</b> (standard bucket) and <b>WARM</b> (infrequent access bucket) are supported.<br>– This parameter is invalid when <b>fsType</b> is set to <b>obsfs</b> .                                                                                                                                                |

| Parameter                        | Mandatory | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
|----------------------------------|-----------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| everest.io/region                | Yes       | Region where the OBS bucket is deployed. For details about the value of <b>region</b> , see <a href="#">Regions and Endpoints</a> .                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
| everest.io/enterprise-project-id | No        | Optional.<br>Enterprise project ID of OBS. If an enterprise project is specified, specify the same enterprise project when creating a PVC. Otherwise, the PVC cannot be bound to a PV.<br><br><b>How to obtain:</b> On the OBS console, choose <b>Buckets</b> or <b>Parallel File Systems</b> in the navigation pane on the left. 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. |
| nodePublishSecretRef             | No        | 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> .<br><br>An example is as follows:<br>nodePublishSecretRef:<br>name: secret-demo<br>namespace: default                                                                                                                                                                                                                                                                                                   |
| mountOptions                     | No        | Mount options. For details, see <a href="#">Configuring OBS Mount Options</a> .                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |



| 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 multiple PVs use the same OBS volume, use <b>Retain</b> to avoid cascading deletion of underlying volumes.</p> <p><b>Delete:</b></p> <ul style="list-style-type: none"> <li>- If <code>everest.io/reclaim-policy</code> is not specified, both the PV and storage resources are deleted when a PVC is deleted.</li> <li>- If <code>everest.io/reclaim-policy</code> is set to <b>retain-volume-only</b>, when a PVC is deleted, the PV is deleted but the 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 is in the <b>Released</b> status and cannot be bound to the PVC again.</p> |
| storage                       | Yes       | <p>Storage capacity, in Gi.</p> <p>For OBS buckets, 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 buckets.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| storageClassName              | Yes       | <p>The storage class name of OBS volumes is <b>csi-obs</b>.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |

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 # Storage class name, which must be the same as that of the PV.
 volumeName: pv-obs # PV name.

```

**Table 12-28** 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.<br><b>How to obtain:</b> On the OBS console, choose <b>Buckets</b> or <b>Parallel File Systems</b> in the navigation pane on the left. 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.<br>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       | Storage class name, which must be the same as the storage class of the PV in <b>1</b> .<br>The storage class name of OBS volumes is <b>csi-obs</b> .                                                                                                                                                                                                                                                                                                        |
| 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-obs.yaml
```

**Step 4** Create an application.

1. 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.
```

2. Run the following command to create an application 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 applications 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 view 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
```

If the **static** file still exists, the data 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 another pods, the two pods share the same volume.

----End

## Related Operations

You can also perform the operations listed in [Table 12-29](#).

**Table 12-29** 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> from the navigation pane, and click the <b>PersistentVolumes (PVs)</b> tab. Click <b>Create Volume</b> in the upper right corner. In the dialog box displayed, configure the parameters. <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 storage that meets your requirements and click <b>OK</b>.</li> <li><b>PV Name:</b> Enter the PV name, which must be unique in the same 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>. For details, see <a href="#">PV Reclaim Policy</a>.</li> </ul> <p><b>NOTE</b><br/>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>Custom:</b> 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> </li> <li>Click <b>Create</b>.</li> </ol> |

| 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> from the navigation pane, and click the <b>PersistentVolumeClaims (PVCs)</b> tab. Click <b>More &gt; Update Access Key</b> in the <b>Operation</b> column of the PVC.</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><br/>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         | You can 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> from the navigation pane, and click the <b>PersistentVolumeClaims (PVCs)</b> or <b>PersistentVolumes (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 (event data is retained for one hour).</li> </ol>                                                                                                                                                                            |
| Viewing a YAML file    | You can view, copy, and download the YAML files of a PVC or PV.                                                                                    | <ol style="list-style-type: none"> <li>1. Choose <b>Storage</b> from the navigation pane, and click the <b>PersistentVolumeClaims (PVCs)</b> or <b>PersistentVolumes (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>                                                                                                                                                                                                                                |

### 12.6.3 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.

#### Constraints

- When parallel file systems and object buckets are used, the group and permission of the mount point cannot be modified.
- CCE allows you to use OBS parallel file systems by calling the OBS SDK or mounting a PVC through the **obsfs** tool provided by OBS. Each time an OBS parallel file system is mounted, an obsfs resident process is generated, as shown in the following figure.

Figure 12-4 obsfs resident process

```

root@k8s-master:~# ps -ef | grep obsfs
root 953 0.0 0.3 53598 4048 ? Ssl 11:09 0:00 /usr/bin/obsfs pvc-e17bf868-3367-4b14-9a23-fbaa55693c11 /mnt/paas/kubernetes/kubelet/pods/6895502_28ac-4300-b684-b32e168b001/volumes/kubernetes.io~csi/pvc-e17bf868-3367-4b14-9a23-fbaa55693c11/mount --url=https://obs.cn-north-1.amazonaws.com.cn/obsfs --uid=443 --endpoint=cn-north-1 --passwd_1=your@everest-host-connector/15179377399250039_obsfs/red/pvc-e17bf868-3367-4b14-9a23-fbaa55693c11 -->
11ow-ether --> zsmemty --> 33x-wr10w --> 33x-wr10w1202 --> max-backround=100 --> oss-10 --> m-obsfs-certificat --> obsfs

```

Reserve 1 GiB of memory for each obsfs process. For example, for a node with 4 vCPUs and 8 GiB of memory, the obsfs parallel file system should be mounted to **no more than** eight pods.

 **NOTE**

An obsfs resident process runs on a node. If the consumed memory exceeds the upper limit of the node, the node malfunctions. On a node with 4 vCPUs and 8 GiB of memory, if more than 100 pods are mounted to parallel file systems, the node will be unavailable. Control the number of pods mounted to parallel file systems on a single node.

- 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. As a result, no more OBS buckets can be created. In this scenario, use OBS through the OBS API or SDK and do not mount OBS buckets to the workload on the console.

## 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** from the navigation pane, and click the **PersistentVolumeClaims (PVCs)** tab. Click **Create PVC** in the upper right corner. In the dialog box displayed, configure the PVC parameters.

| Parameter       | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
|-----------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| PVC Type        | In this section, select <b>OBS</b> .                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| PVC Name        | Enter the PVC name, which must be unique in the same 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 storage volume or use an existing storage volume to statically create a PVC based on whether a PV has been created. For details, see <a href="#">Using an Existing OBS Bucket Through a Static PV</a>.</li> </ul> <p>In this example, select <b>Dynamically provision</b>.</p> |
| Storage Classes | The storage class of 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. <b>Parallel file systems are recommended.</b></li> <li>– <b>Object bucket:</b> a container that stores objects in OBS. All objects in a bucket are at the same logical level.</li> </ul>                                                                                                                               |

| Parameter          | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
|--------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| OBS Class          | <p>You can select the following object bucket types:</p> <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        | <p>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>.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| Secret             | <p><b>Custom:</b> 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> |
| Enterprise Project | <p>Supported enterprise projects: default, the one to which the cluster belongs, or the one specified by the storage class.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |

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 **PersistentVolumeClaims (PVCs)** and **PersistentVolumes (PVs)** tab pages.

**Step 3** Create an application.

1. In the navigation pane on the left, click **Workloads**. 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 12-30](#). For details about other parameters, see [Workloads](#).



**Table 12-30** Mounting a storage volume

| Parameter  | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
|------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| PVC        | Select an existing object storage volume.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| Mount Path | <p>Enter a mount path, for example, <b>/tmp</b>.</p> <p>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>. 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, causing container startup failures or workload creation failures.</p> <p><b>NOTICE</b><br/>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 machine may be damaged.</p> |
| Subpath    | <p>Enter a subpath, for example, <b>tmp</b>, indicating that data in the mount path of the container will be stored in the <b>tmp</b> folder of the volume.</p> <p>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 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 is not 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 [PV Reclaim Policy](#).

----End

## (kubectl) Automatically Creating an OBS Volume

**Step 1** Use `kubectl` to connect to 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 # Object storage type.
 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 # For object storage, the value must be ReadWriteMany.
 resources:
 requests:
 storage: 1Gi # OBS volume capacity.
 storageClassName: csi-obs # The storage class type is OBS.

```

**Table 12-31** Key parameters

| Parameter                                        | Mandatory | Description                                                                                                                                                                                                                                                                               |
|--------------------------------------------------|-----------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| everest.io/obs-volume-type                       | Yes       | OBS storage class.<br><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> .<br><ul style="list-style-type: none"> <li><b>obsfs</b>: Parallel file system, which is mounted using obsfs (recommended).</li> <li><b>s3fs</b>: Object bucket, which is mounted using s3fs.</li> </ul>                      |
| csi.storage.k8s.io/node-publish-secret-name      | No        | Custom secret name.<br>(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.<br><b>How to Obtain:</b><br><b>How to obtain:</b> On the OBS console, choose <b>Buckets</b> or <b>Parallel File Systems</b> in the navigation pane on the left. 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.<br>For OBS buckets, 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 buckets.                                                                                                                                                                                                                                                              |
| storageClassName                 | Yes       | Storage class name. The storage class name of OBS volumes is <b>csi-obs</b> .                                                                                                                                                                                                                                                                                                                                                                                                        |

- 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.

```

2. Run the following command to create an application 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 applications 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 view 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
```

If the **static** file still exists, the data can be stored persistently.

**Step 5** Verify data sharing.

- 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
```
- 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
```
- 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 another pods, the two pods share the same volume.

----End

## Related Operations

You can also perform the operations listed in [Table 12-32](#).

**Table 12-32** 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>Choose <b>Storage</b> from the navigation pane, and click the <b>PersistentVolumeClaims (PVCs)</b> tab. Click <b>More &gt; Update Access Key</b> in the <b>Operation</b> column of the PVC.</li> <li>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><br/>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> |

| Operation           | Description                                                                                                                                        | Procedure                                                                                                                                                                                                                                                                                                                                                                  |
|---------------------|----------------------------------------------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Events              | You can 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> from the navigation pane, and click the <b>PersistentVolumeClaims (PVCs)</b> or <b>PersistentVolumes (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 (event data is retained for one hour).</li> </ol> |
| Viewing a YAML file | You can view, copy, and download the YAML files of a PVC or PV.                                                                                    | <ol style="list-style-type: none"> <li>1. Choose <b>Storage</b> from the navigation pane, and click the <b>PersistentVolumeClaims (PVCs)</b> or <b>PersistentVolumes (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>                                                     |

## 12.6.4 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 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.

### OBS Mount Options

When mounting an OBS volume, the everest add-on presets the options described in [Table 12-33](#) and [Table 12-34](#) by default. The options in [Table 12-33](#) are mandatory.

**Table 12-33** Mandatory mount options configured by default

| Parameter  | Value           | Description                                                                                 |
|------------|-----------------|---------------------------------------------------------------------------------------------|
| use_ino    | Leave it blank. | If enabled, obsfs allocates the <b>inode</b> number. Enabled by default in read/write mode. |
| big_writes | Leave it blank. | If configured, the maximum size of the cache can be modified.                               |

| Parameter            | Value           | Description                                                                                                                                                                                                             |
|----------------------|-----------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| nonempty             | Leave it blank. | Allows non-empty mount paths.                                                                                                                                                                                           |
| allow_other          | Leave it blank. | Allows other users to access the parallel file system.                                                                                                                                                                  |
| no_check_certificate | Leave it blank. | Disables server certificate verification.                                                                                                                                                                               |
| enable_noobj_cache   | Leave it blank. | Enables cache entries for objects that do not exist, which can improve performance. Enabled by default in object bucket read/write mode.<br><b>This option is no longer configured by default since everest 1.2.40.</b> |
| sigv2                | Leave it blank. | Specifies the signature version. Used by default in object buckets.                                                                                                                                                     |

**Table 12-34** 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. |
| public_bucket       | 1               | If set to <b>1</b> , public buckets are mounted anonymously. Enabled by default in object bucket read/write mode.  |
| umask               | Leave it blank. | Mask of the configuration file permission.                                                                         |

## 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 connect to 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 PV is deleted while the underlying
volume is retained.
 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 # Storage class name.
 mountOptions:
 - umask=0027

```

**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=0027 -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=0027 -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 connect to the cluster. For details, see [Connecting to a Cluster Using kubectl](#).

**Step 2** Create a customized 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=0027
```

**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=0027 -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=0027 -o max_write=131072
```

----End

## 12.6.5 Using a Custom Access Key (AK/SK) to Mount an OBS Volume

### Scenario

You can solve this issue by using everest 1.2.8 or later 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.

## 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

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, 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-k --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=supporthis=sfsfs --project-id=b6315d340ff4bc5b31a9c3253794989
--cluster-id=827dced9-c2ad-11ea-bf6e-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>$!
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.                                                                                                                         |
| 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> .<br>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-0
 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 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
```

## 12.7 Local Persistent Volumes (Local PVs)

### 12.7.1 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.

#### Mounting Modes

Local PVs can be mounted only in the following modes:

- **Using a Local PV Through a Dynamic PV:** dynamic creation mode, where you specify a StorageClass during PVC creation and an OBS volume and a PV will be automatically created.
- **Dynamically Mounting a Local PV to a StatefulSet:** Only StatefulSets support 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.

## 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, removing, resetting, or scaling in a node will cause the PVC/PV data of the local PV associated with the node to be lost, which cannot be restored or used again. For details, see [Removing a Node](#), [Deleting a Node](#), [Resetting a Node](#), and [Scaling a Node](#). 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.
- A local PV cannot be mounted to multiple workloads or jobs at the same time.

## 12.7.2 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.

### 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 storage volume is insufficient, you can manually import a storage pool.

- Step 1** Go to the ECS console and add a SCSI disk to the node.
- Step 2** Log in to the CCE console and click the cluster name to access the cluster console.
- Step 3** In the navigation pane, choose **Storage** and switch to 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, allowing data to be concurrently read and written. Select this option only when there are multiple volumes.

----End

## 12.7.3 Using a Local PV Through a Dynamic PV

### Prerequisites

- You have created a cluster and installed the CSI add-on ([everest](#)) in the cluster.
- If you want to create a cluster using commands, use `kubectl` to connect to the cluster. 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](#).

### 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, removing, resetting, or scaling in a node will cause the PVC/PV data of the local PV associated with the node to be lost, which cannot be restored or used again. For details, see [Removing a Node](#), [Deleting a Node](#), [Resetting a Node](#), and [Scaling a Node](#). 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.
- A local PV cannot be mounted to multiple workloads or jobs at the same time.

## 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.

1. Choose **Storage** from the navigation pane, and click the **PersistentVolumeClaims (PVCs)** tab. Click **Create PVC** in the upper right corner. In the dialog box displayed, configure the 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 the same 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 storage class of local PVs is <b>csi-local-topology</b> .                                                                                                                        |
| Access Mode     | Local PVs 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> . |
| 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> .                        |
| Capacity (GiB)  | Capacity of the requested storage volume.                                                                                                                                            |

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 **PersistentVolumeClaims (PVCs)** and **PersistentVolumes (PVs)** tab pages.

### 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. In the navigation pane on the left, click **Workloads**. 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 12-35](#). For details about other parameters, see [Workloads](#).

**Table 12-35** Mounting a storage volume

| Parameter  | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
|------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| PVC        | Select an existing local PV.<br>A local PV cannot be repeatedly mounted to multiple workloads.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
| Mount Path | Enter a mount path, for example, <b>/tmp</b> .<br>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> . 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, causing container startup failures or workload creation failures.<br><b>NOTICE</b><br>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 machine may be damaged. |
| Subpath    | Enter a subpath, for example, <b>tmp</b> , indicating that data in the mount path of the container will be stored in the <b>tmp</b> folder of the volume.<br>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 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 is not 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

## (kubectl) Automatically Creating a Local PV

**Step 1** Use kubectl to connect to 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 local PV must adopt ReadWriteOnce.
 resources:
 requests:
 storage: 10Gi # Size of the local PV.
 storageClassName: csi-local-topology # StorageClass is local PV.
```

**Table 12-36** Key parameters

| Parameter        | Mandatory | Description                                                                           |
|------------------|-----------|---------------------------------------------------------------------------------------|
| storage          | Yes       | Requested capacity in the PVC, in Gi.                                                 |
| storageClassName | Yes       | Storage class name. The storage class name of local PV is <b>csi-local-topology</b> . |

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-demo.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 an application 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 view 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 view 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
```

If the **static** file still exists, the data in the local PV can be stored persistently.

----End

## Related Operations

You can also perform the operations listed in [Table 12-37](#).

**Table 12-37** Related operations

| Operation           | Description                                                                                                                                        | Procedure                                                                                                                                                                                                                                                                                                                                                                  |
|---------------------|----------------------------------------------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Viewing events      | You can 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> from the navigation pane, and click the <b>PersistentVolumeClaims (PVCs)</b> or <b>PersistentVolumes (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 (event data is retained for one hour).</li> </ol> |
| Viewing a YAML file | You can view, copy, and download the YAML files of a PVC or PV.                                                                                    | <ol style="list-style-type: none"> <li>1. Choose <b>Storage</b> from the navigation pane, and click the <b>PersistentVolumeClaims (PVCs)</b> or <b>PersistentVolumes (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>                                                     |

## 12.7.4 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 the storage class to dynamically provision PVs. 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.

### Prerequisites

- You have created a cluster and installed the CSI add-on ([everest](#)) in the cluster.
- If you want to create a cluster using commands, use `kubectl` to connect to the cluster. 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** In the navigation pane on the left, click **Workloads**. 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 (VTC)**.
- 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, <i>example-0</i> . |
| 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 storage class of local PVs is <b>csi-local-topology</b> .                                                                                                                                                        |

| Parameter      | Description                                                                                                                                                                          |
|----------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Access Mode    | Local PVs 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> . |
| 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> .                        |
| Capacity (GiB) | Capacity of the requested storage volume.                                                                                                                                            |

**Step 5** Enter the path to which the volume is mounted.

**Table 12-38** Mounting a storage volume

| Parameter  | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
|------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Mount Path | <p>Enter a mount path, for example, <b>/tmp</b>.</p> <p>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>. Otherwise, errors will occur in containers. 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, causing container startup failures or workload creation failures.</p> <p><b>NOTICE</b><br/>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 machine may be damaged.</p> |
| Subpath    | <p>Enter a subpath, for example, <b>tmp</b>, indicating that data in the mount path of the container will be stored in the <b>tmp</b> folder of the volume.</p> <p>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 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 is not 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

## (kubectl) Using an Existing Local PV

**Step 1** Use kubectl to connect to 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 local PV must adopt ReadWriteOnce.
 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 12-39** Key parameters

| Parameter        | Mandatory | Description                                                   |
|------------------|-----------|---------------------------------------------------------------|
| storage          | Yes       | Requested capacity in the PVC, in Gi.                         |
| storageClassName | Yes       | The storage class of local PVs is <b>csi-local-topology</b> . |

**Step 3** Run the following command to create an application 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 view 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 view 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
```

If the **static** file still exists, the data in the local PV can be stored persistently.

----End

## Related Operations

You can also perform the operations listed in [Table 12-40](#).

**Table 12-40** Related operations

| Operation           | Description                                                                                                                                        | Procedure                                                                                                                                                                                                                                                                                                                                                                  |
|---------------------|----------------------------------------------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Viewing events      | You can 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> from the navigation pane, and click the <b>PersistentVolumeClaims (PVCs)</b> or <b>PersistentVolumes (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 (event data is retained for one hour).</li> </ol> |
| Viewing a YAML file | You can view, copy, and download the YAML files of a PVC or PV.                                                                                    | <ol style="list-style-type: none"> <li>1. Choose <b>Storage</b> from the navigation pane, and click the <b>PersistentVolumeClaims (PVCs)</b> or <b>PersistentVolumes (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>                                                     |

## 12.8 Ephemeral Volumes (emptyDir)

### 12.8.1 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 temporary 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 containers. LVs deliver better performance than the default storage medium of emptyDir.

## 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.

## 12.8.2 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.

## 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](#).

### Imported manually

If no EV is imported during node creation, or the capacity of the current storage volume is insufficient, you can manually import a storage pool.

- Step 1** Go to the ECS console and add a SCSI disk to the node.
- Step 2** Log in to the CCE console and click the cluster name to access the cluster console.
- Step 3** In the navigation pane, choose **Storage** and switch to 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, allowing data to be concurrently read and written. Select this option only when there are multiple volumes.

----End

## 12.8.3 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.
- If you want to create a cluster using commands, use kubectl to connect to the cluster. 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](#).

### 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.

- The `/var/lib/kubelet/pods/` directory cannot be mounted to pods running on the node. Otherwise, the pods 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** In the navigation pane on the left, click **Workloads**. In the right pane, click the **Deployments** tab.
- Step 3** Click **Create Workload** in the upper right corner of the page. In the **Container Settings** area, click the **Data Storage** tab and click **Add Volume > Local Ephemeral Volume (emptyDir)**.
- Step 4** Mount and use storage volumes, as shown in [Table 12-41](#). For details about other parameters, see [Workloads](#).

**Table 12-41** Mounting a local EV

| Parameter  | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
|------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Capacity   | Capacity of the requested storage volume.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| Mount Path | <p>Enter a mount path, for example, <code>/tmp</code>.</p> <p>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 <code>/</code> or <code>/var/run</code>. 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, causing container startup failures or workload creation failures.</p> <p><b>NOTICE</b><br/>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 machine may be damaged.</p> |
| Subpath    | <p>Enter a subpath, for example, <code>tmp</code>, indicating that data in the mount path of the container will be stored in the <code>tmp</code> folder of the volume.</p> <p>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 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 is not migrated if the container is migrated, which may cause data loss.</li> </ul>                                                                                                                                                                                                                                                                                                                                                                                                                                                  |

- Step 5** After the configuration, click **Create Workload**.

----End

## Using kubectl to Mount a Local EV

**Step 1** Use kubectl to connect to 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 # Path to which an EV 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
```

### 12.8.4 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 the Console to Use a Temporary Path

**Step 1** Log in to the CCE console and click the cluster name to access the cluster console.

**Step 2** In the navigation pane on the left, click **Workloads**. In the right pane, click the **Deployments** tab.

**Step 3** Click **Create Workload** in the upper right corner of the page. In the **Container Settings** area, click the **Data Storage** tab and click **Add Volume > emptyDir**.

**Step 4** Mount and use storage volumes, as shown in [Table 12-42](#). For details about other parameters, see [Workloads](#).

**Table 12-42** 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 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>. 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, causing 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 machine may be damaged.</p>                                                                                                           |
| Subpath        | <p>Enter a subpath, for example, <b>tmp</b>, indicating that data in the mount path of the container will be stored in the <b>tmp</b> folder of the volume.</p> <p>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 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 is not migrated if the container is migrated, which may cause data loss.</li> </ul>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |

**Step 5** After the configuration, click **Create Workload**.

----End

## Using kubectl to Use a Temporary Path

**Step 1** Use kubectl to connect to 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 # Path to which an EV 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

## 12.9 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 paths 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 the **Container Settings** area. Click **Add Volume** and choose **hostPath** from the drop-down list.
- Step 3** Set parameters for adding a local volume, as listed in [Table 12-43](#).

**Table 12-43** Setting parameters for mounting a hostPath volume

| Parameter   | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
|-------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Volume Type | Select <b>HostPath</b> .                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| 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><br/> <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> |

| Parameter  | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
|------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Mount Path | <p>Enter a mount path, for example, <b>/tmp</b>.</p> <p>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>. 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, causing container startup failures or workload creation failures.</p> <p><b>NOTICE</b><br/>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 machine may be damaged.</p> |
| Subpath    | <p>Enter a subpath, for example, <b>tmp</b>, indicating that data in the mount path of the container will be stored in the <b>tmp</b> folder of the volume.</p> <p>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 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 is not 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 Using kubectl

**Step 1** Use kubectl to connect to 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 # Mount path in the container.
 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

```

## 12.10 StorageClass

### Introduction

StorageClass describes the classification of storage types in a cluster and can be represented as a configuration template for creating PVs. When creating a PVC or PV, specify StorageClass.

As a user, you only need to specify **storageClassName** when defining a PVC to automatically create a PV and underlying storage, significantly reducing the workload of creating and maintaining a PV.

In addition to the [default storage classes](#) provided by CCE, you can also customize storage classes.

- [Application Scenarios of Custom Storage](#)
- [Custom Storage Class](#)
- [Specifying a Default StorageClass](#)
- [Specifying an Enterprise Project for Storage Classes](#)

### CCE Default Storage Classes

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.

Run the following kubectl command to obtain the storage classes that CCE supports. Use the CSI add-on provided by CCE to create a storage class.

```

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 delayed
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 storage class contains the default parameters used for dynamically creating a PV. The following is an example of storage class 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
```

| 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.                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| 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>: indicates that a dynamically created PV will be automatically destroyed.</li> <li><b>Retain</b>: indicates that a dynamically created PV will not be automatically destroyed.</li> </ul> |
| allowVolumeExpansion | Specifies whether the PV of this storage class 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>: PV binding and dynamic creation are completed when a PVC is created.</li> <li><b>WaitForFirstConsumer</b>: PV binding and creation are delayed. The PV creation and binding processes are executed only when the PVC is used in the workload.</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.                                                                                                                                                                                                                                                                                                                                                                   |

## 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.

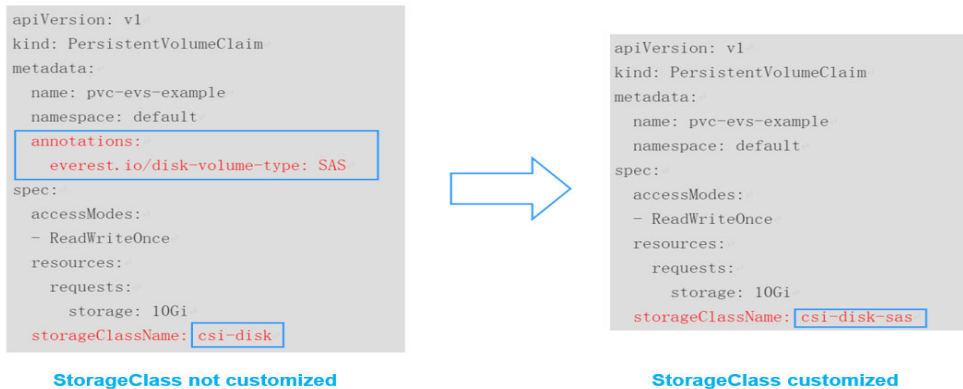
The preceding is a basic method of using StorageClass. In real-world scenarios, you can use StorageClass to perform other operations.

| 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 storage class, respectively. If a storage class named <b>csi-disk-sas</b> is defined, it is used to create SAS storage.</p>                                                                                                                                                                                      | <p><b>Custom Storage Class</b></p> |
| <p>When a user migrates services from a self-built Kubernetes cluster or other Kubernetes services to CCE, the storage class 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 storage class with the same name as that in the original application YAML file in the CCE centralization. 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 storage class 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> storage class in the CCE cluster, change its name to <b>disk-standard</b>, and create another storage class.</p> |                                    |

| Application Scenario                                                                                                  | Solution                                                                                                                                    | Procedure                                         |
|-----------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------|
| <b>storageClassName</b> must be specified in the YAML file to use the storage. If not, the storage cannot be created. | If you set the default StorageClass in the cluster, you can create storage without specifying the <b>storageClassName</b> in the YAML file. | <a href="#">Specifying a Default StorageClass</a> |

### Custom Storage Class

This section uses the custom storage class of EVS disks as an example to describe how to define SAS EVS disk and SSD EVS disk as a storage class, respectively. For example, if you define a storage class named **csi-disk-sas**, which is used to create SAS storage, the differences are shown in the following figure. When compiling a YAML file, you only need to specify **storageClassName**.



- 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 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, the PV and underlying storage resources are not deleted. Instead, you must manually delete these resources. After that, the PV is in the **Released** status 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 **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
```

## Specifying a Default StorageClass

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
```

## 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
```

## 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/
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.

# 13 Observability

---

## 13.1 Logging

### 13.1.1 Overview

CCE allows you to configure policies for collecting, managing, and analyzing workload logs periodically to prevent logs from being over-sized.

- Using ICAgent:

By default, the ICAgent collects container standard outputs (stdout logs). No configuration required.

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**: 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.
- **emptyDir**: 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.

### 13.1.2 Using ICAgent to Collect Container Logs


CCE works with AOM to collect workload logs. When creating a node, CCE installs the ICAgent for you (the DaemonSet named **icagent** in the kube-system namespace of the cluster). After the ICAgent collects workload logs and reports them to AOM, you can view workload logs on the CCE or AOM console.

#### Constraints

The ICAgent only collects **\*.log**, **\*.trace**, and **\*.out** text log files.

## 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 **Host Path** or **Container Path**.

**Table 13-1** Configuring log policies

| Parameter   | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
|-------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Volume Type | <ul style="list-style-type: none"> <li>• <b>Host Path</b> (hostPath): 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>Container Path</b> (emptyDir): 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>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| Host Path   | 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 a volume to a system directory such as <code>/</code> or <code>/var/run</code>. Otherwise, an exception occurs. 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, which leads to a container startup failure or workload creation failure.</li> <li>• 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 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 <code>.log</code>, <code>.trace</code>, and <code>.out</code> text log files in the mount 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> |

| Parameter          | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
|--------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 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>                                                                                                                                                                                                                                                                                                                                                                                            |
| 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>● * 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><br/>Ensure that the ICAgent version is 5.12.22 or later.</p>                                                                                                                                                                                                                                                             |
| 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 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:
 pathPattern: '**'

 volumes:
 - hostPath:
 path: /tmp/log
 name: vol-log
 imagePullSecrets:
 - name: default-secret

```

**Table 13-2** 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> |

| Parameter                           | Description     | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
|-------------------------------------|-----------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 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 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> |
| 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><br/>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)
```

## 13.2 Monitoring

### 13.2.1 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 [Monitoring Custom Metrics on AOM](#).

### Resource Metrics

On the CCE console, you can view the following metrics.

- [Viewing Cluster Monitoring Data](#)
- [Viewing Monitoring Data of Worker Nodes](#)
- [Viewing Workload Monitoring Data](#)
- [Viewing Pod Monitoring Data](#)

On the AOM console, you can view host metrics and container metrics.

### Viewing Cluster Monitoring Data

- Step 1** Log in to the CCE console and click the cluster name to access the cluster console.
- Step 2** CCE allows you to view the monitoring data of all nodes. Choose **Clusters** from the navigation pane. Click the cluster name, and information like **CPU Metrics** and **Memory** of all nodes (excluding master nodes) in the last hour, the **Status**, **AZ** are displayed.



**Table 13-3** Cluster monitoring metrics

| Metric                | Description                                                                                                                                                                                                                                                         |
|-----------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| CPU Allocation (%)    | A metric indicates the percentage of CPUs allocated to workloads.<br><b>CPU Allocation (%)</b> = Sum of CPU quotas requested by running pods in the cluster/Sum of CPU quotas that can be allocated from all nodes (excluding master nodes) to workloads            |
| Memory Allocation (%) | A metric indicates the percentage of memory allocated to workloads.<br><b>Memory Allocation (%)</b> = Sum of memory quotas requested by running pods in the cluster/Sum of memory quotas that can be allocated from all nodes (excluding master nodes) to workloads |
| CPU Usage (%)         | A metric indicates the CPU usage of the cluster.<br>This metric is the average CPU usage of all nodes (excluding master nodes) in a cluster.                                                                                                                        |
| Memory Usage (%)      | A metric indicates the memory usage of your cluster.<br>This metric is the 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 [Node Resource Reservation Policy](#).

----End

## Viewing Monitoring Data of Worker Nodes

CCE also allows you to view monitoring data of a single node.

- Step 1** Log in to the CCE console and click the cluster name to access the cluster console.
- Step 2** Choose **Nodes** in the navigation pane. On the right of the page, click **Monitor** of the target node to view the monitoring data.
- Step 3** You can select statistical **Dimension** and choose time range to view the monitoring data. The data is provided by AOM. You can view the monitoring data of a node, including the CPU, memory, disk, networking, and GPU.

**Table 13-4** Node monitoring metrics

| Metric        | Description                                                                                                      |
|---------------|------------------------------------------------------------------------------------------------------------------|
| CPU Usage (%) | A metric indicates the CPU usage of the node.<br><b>CPU Usage (%)</b> = Used CPU cores/Total number of CPU cores |

| Metric                          | Description                                                                                                                                                                                                                                                                    |
|---------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Used CPU Cores (cores)          | A metric indicates the number of used CPU cores.                                                                                                                                                                                                                               |
| Physical Memory Usage (%)       | A metric indicates the physical memory usage of the node<br><b>Physical Memory Usage (%)</b> = (Physical memory capacity - Available physical memory)/Physical memory capacity                                                                                                 |
| Available Physical Memory (GiB) | A metric indicates the unused physical memory of the node.                                                                                                                                                                                                                     |
| Disk Usage (%)                  | A metric indicates the disk usage of the file system on the data disk of the node. It is calculated based on the file partition. For details, see <a href="#">Data Disk Space Allocation</a> .<br><b>Disk Usage (%)</b> = (Disk capacity - Available disk space)/Disk capacity |
| Available Disk Space (GiB)      | A metric indicates the unused disk space.                                                                                                                                                                                                                                      |
| Downlink Rate (BPS) (KB/s)      | A metric indicates the speed at which data is downloaded from the Internet to the node.                                                                                                                                                                                        |
| Uplink Rate (BPS) (KB/s)        | A metric indicates the speed at which data is uploaded from the node to the Internet.                                                                                                                                                                                          |
| GPU Usage (%)                   | A metric indicates the GPU usage of the node.                                                                                                                                                                                                                                  |
| GPU Memory Usage (%)            | A metric indicates the percentage of the used GPU memory to the GPU memory capacity.<br><b>GPU Memory Usage (%)</b> = Used GPU memory/GPU memory capacity                                                                                                                      |
| Used GPU Memory (GiB)           | A metric indicates the used GPU memory.                                                                                                                                                                                                                                        |

----End

## Viewing Workload Monitoring Data

CCE allows you to view monitoring data of a single workload.

- 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. On the right of the page, click **Monitor** of the target workload. In the window that slides out from the right, the workload monitoring data is displayed.

**Step 3** You can select statistical **Dimension** and choose time range to view the monitoring data. The data is provided by AOM. You can view the monitoring data of a workload, including the CPU, memory, networking, and GPU.

 **NOTE**

If there are multiple pods exist in the workload, the monitoring data may vary according to the statistical **Dimension**. For example, if you select **Maximum** or **Minimum** for **Dimension**, the value of each monitoring data is the maximum or minimum value of all pods under the workload. If **Average** is selected, the value of each monitoring data is the average value of all pods under the workload.

**Table 13-5** Workload monitoring metrics

| Metric                     | Description                                                                                                                                                                                                                                           |
|----------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| CPU Usage (%)              | A metric indicates the CPU usage of the workload.<br><b>CPU Usage (%)</b> = Used CPU cores/Total number of CPU cores of all running pods (If no limit is configured, the total number of the node's CPU cores is used.)                               |
| Used CPU Cores (cores)     | A metric indicates the number of used CPU cores.                                                                                                                                                                                                      |
| Physical Memory Usage (%)  | A metric indicates the physical memory usage of the workload.<br><b>Physical Memory Usage (%)</b> = Used physical memory/Total number of CPU cores of all running pods (If no limit is configured, the total number of the node's CPU cores is used.) |
| Used Physical Memory (GiB) | A metric indicates the amount of the used physical memory.                                                                                                                                                                                            |
| Disk Read Rate             | A metric indicates the data volume read from a disk per second. The unit is KB/s.                                                                                                                                                                     |
| Disk Write Rate            | A metric indicates the data volume written to a disk per second. The unit is KB/s.                                                                                                                                                                    |
| Downlink Rate (BPS) (KB/s) | A metric indicates the speed at which data is downloaded from the Internet.                                                                                                                                                                           |
| Uplink Rate (BPS) (KB/s)   | A metric indicates the speed at which data is uploaded from the node to the Internet.                                                                                                                                                                 |
| GPU Usage (%)              | A metric indicates the GPU usage of the workload.                                                                                                                                                                                                     |
| GPU Memory Usage (%)       | A metric indicates the percentage of the used GPU memory to the GPU memory capacity.<br><b>GPU Memory Usage (%)</b> = Used GPU memory/GPU memory capacity                                                                                             |

| Metric                | Description                             |
|-----------------------|-----------------------------------------|
| Used GPU Memory (GiB) | A metric indicates the used GPU memory. |

----End

## Viewing Pod Monitoring Data

CCE allows you to view the monitoring data of your pods.

- 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. Then click the workload name of the target workload to list the pods.
- Step 3** Click **Monitor** of the target pod to view the monitoring data.
- Step 4** You can select statistical **Dimension** and choose time range to view the monitoring data. The data is provided by AOM. You can view the monitoring data of a pod, including the CPU, memory, disk, networking, and GPU.

### NOTE

If there are multiple containers in a single pod, the monitoring data may vary according to the statistical **Dimension**. For example, if you select **Maximum** or **Minimum** for **Dimension**, the value of each monitoring data is the maximum or minimum value of all containers under the pod. If **Average** is selected, the value of each monitoring data is the average value of all containers in the pod.

**Table 13-6** Pod monitoring metrics

| Metric                     | Description                                                                                                                                                                                                                                                                      |
|----------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| CPU Usage (%)              | A metric indicates the CPU usage of the pod.<br><b>CPU Usage (%)</b> = Used CPU cores/Total number of limited CPU cores of all running containers in the pod (If the limited CPU cores of all running containers are not specified, the number of the node's CPU cores is used.) |
| Used CPU Cores (cores)     | A metric indicates the number of used CPU cores.                                                                                                                                                                                                                                 |
| Physical Memory Usage (%)  | A metric indicates the physical memory usage of the pod.<br><b>Physical Memory Usage (%)</b> = Used physical memory/Sum of physical memory limits of all running containers in the pod (If not specified, the value of the node's physical memory is used.)                      |
| Used Physical Memory (GiB) | A metric indicates the amount of the used physical memory.                                                                                                                                                                                                                       |
| Disk Read Rate             | A metric indicates the data volume read from a disk per second. The unit is KB/s.                                                                                                                                                                                                |

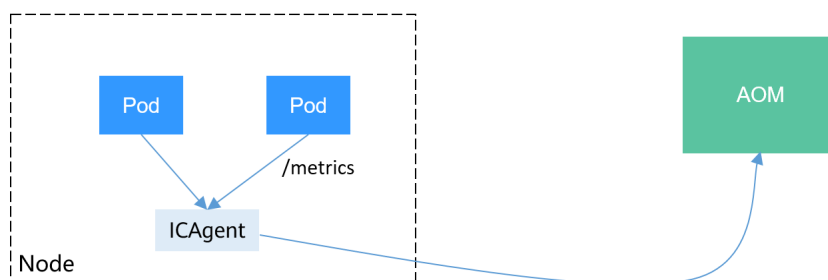
| Metric                     | Description                                                                                                                                               |
|----------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------|
| Disk Write Rate            | A metric indicates the data volume written to a disk per second. The unit is KB/s.                                                                        |
| Downlink Rate (BPS) (KB/s) | A metric indicates the speed at which data is downloaded from the Internet.                                                                               |
| Uplink Rate (BPS) (KB/s)   | A metric indicates the speed at which data is uploaded from the node to the Internet.                                                                     |
| GPU Usage (%)              | A metric indicates the GPU usage of the pod.                                                                                                              |
| GPU Memory Usage (%)       | A metric indicates the percentage of the used GPU memory to the GPU memory capacity.<br><b>GPU Memory Usage (%) = Used GPU memory/GPU memory capacity</b> |
| Used GPU Memory (GiB)      | A metric indicates the used GPU memory of the pod.                                                                                                        |

----End

## 13.2.2 Monitoring Custom Metrics on AOM

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.

**Figure 13-1** 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 monitoring 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

Self-developed applications must provide a metric monitoring API for ICAgent to collect data, and the monitoring data must comply with the Prometheus specifications. For details, see **Prometheus Monitoring Data Collection**.

This document 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 build an image and upload it to SWR. The image name is **nginx:exporter**.

1. In the navigation pane, choose **My Images**. Click **Upload Through Client** in the upper right corner. On the page displayed, click **Generate a temporary login command** and 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.

```
docker tag nginx:exporter {swr-address}/{group}/nginx:exporter
docker push {swr-address}/{group}/nginx:exporter
```

**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
```

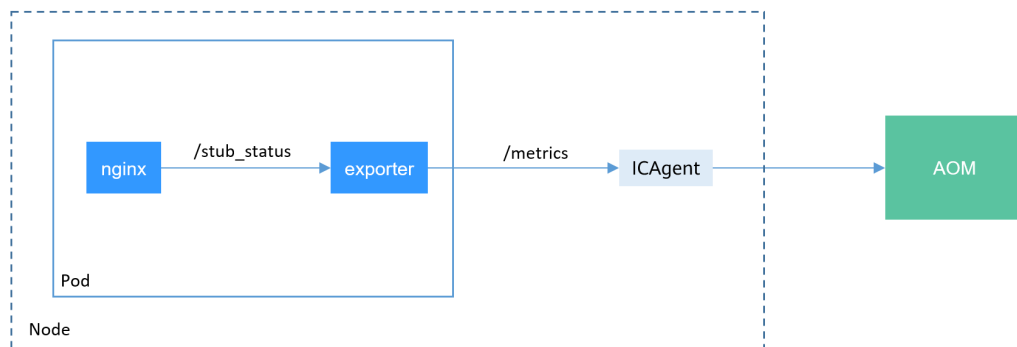
```
146269 146269 212
Reading: 0 Writing: 1 Waiting: 2
```

----End

## Deploying Applications and Converting Nginx Metrics

The data 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 13-2** 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
nginx.org.

Commercial support is available at
nginx.com.</p>

<p>Thank you for using nginx.</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

## 13.3 CTS Logs

### 13.3.1 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 13-7** CCE operations supported by CTS

Operation	Resource Type	Event Name
Creating an agency	Cluster	createUserAgencies
Creating a cluster	Cluster	createCluster
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 cluster	Cluster	resizeCluster
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
Modifying node pool configurations	Node pool	updateConfiguration

Operation	Resource Type	Event Name
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
Creating a PersistentVolume-Claim	Kubernetes resource	createPersistentvolumeclaims

Operation	Resource Type	Event Name
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
Deleting a resource quota	Kubernetes resource	deleteResourcequotas
Deleting a secret	Kubernetes resource	deleteSecrets

Operation	Resource Type	Event Name
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
Replacing a specified service	Kubernetes resource	updateServices
Replacing a specified StatefulSet	Kubernetes resource	updateStatefulsets

Operation	Resource Type	Event Name
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

## 13.3.2 Querying Real-Time Traces


### Scenarios



After you enable CTS and the management tracker is created, CTS starts recording operations on cloud resources. CTS stores operation records generated in the last seven days.

This section describes how to query and export operation records of the last seven days on the CTS console.

- [Viewing Real-Time Traces in the Trace List](#)

### Viewing Real-Time Traces in the Trace List

1. Log in to the management console.
2. Click  in the upper left corner and choose **Management & Deployment > Cloud Trace Service**. The CTS console is displayed.
3. Choose **Trace List** in the navigation pane on the left.
4. Set filters to search for your desired traces. The following filters are available:
  - **Trace Type, Trace Source, Resource Type, and Search By:** Select a filter from the drop-down list.
    - If you select **Resource ID** for **Search By**, specify a resource ID.
    - If you select **Trace name** for **Search By**, specify a trace name.
    - If you select **Resource name** for **Search By**, specify a resource name.

- **Operator:** Select a user.
  - **Trace Status:** Select **All trace statuses**, **Normal**, **Warning**, or **Incident**.
  - Time range: You can query traces generated during any time range in the last seven days.
  - Click **Export** to export all traces in the query result as a CSV file. The file can contain up to 5000 records.
5. Click **Query**.
  6. On the **Trace List** page, you can also export and refresh the trace list.
    - Click **Export** to export all traces in the query result as a CSV file. The file can contain up to 5000 records.
    - Click  to view the latest information about traces.
  7. Click  on the left of a trace to expand its details.
  8. Click **View Trace** in the **Operation** column. The trace details are displayed.

View Trace ×

```
{
 "request": "",
 "trace_id": "XXXXXXXXXXXXXXXXXXXX",
 "code": "200",
 "trace_name": "createDockerConfig",
 "resource_type": "dockerlogincmd",
 "trace_rating": "normal",
 "api_version": "",
 "message": "createDockerConfig, Method: POST Url=/v2/manage/utills/secret, Reason:",
 "source_ip": "XXXXXXXXXX",
 "domain_id": "XXXXXXXXXXXXXXXXXXXX",
 "trace_type": "ApiCall",
 "service_type": "SWR",
 "event_type": "system",
 "project_id": "XXXXXXXXXXXXXXXXXXXX",
 "response": "",
 "resource_id": "",
 "tracker_name": "system",
 "time": "Nov 16, 2023 10:54:04 GMT+08:00",
 "resource_name": "dockerlogincmd",
 "user": {
 "domain": {
 "name": "XXXXXXXXXX",
 "id": "XXXXXXXXXXXXXXXXXXXX"
 }
 }
}
```

9. For details about key fields in the trace structure, see section "Trace References" > "Trace Structure" and section "Trace References" > "Example Traces".

# 14 Namespaces

---

## 14.1 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.

### 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 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 14-1](#).

**Table 14-1** 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
```

## 14.2 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.

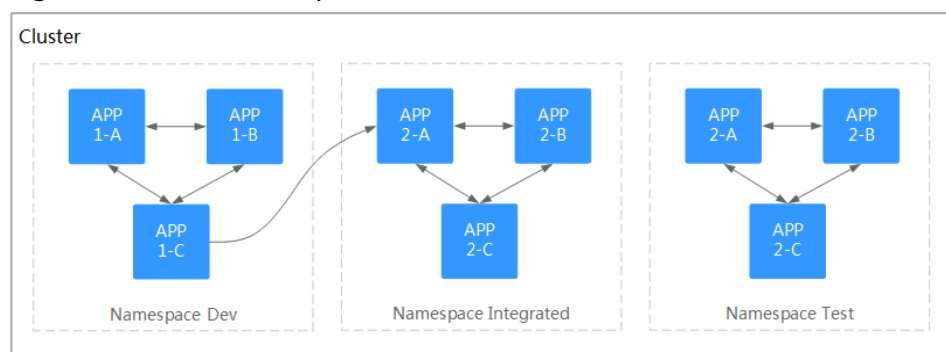
Resources cannot be shared among different clusters. In addition, services in different environments can access each other only through load balancing.

- 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.

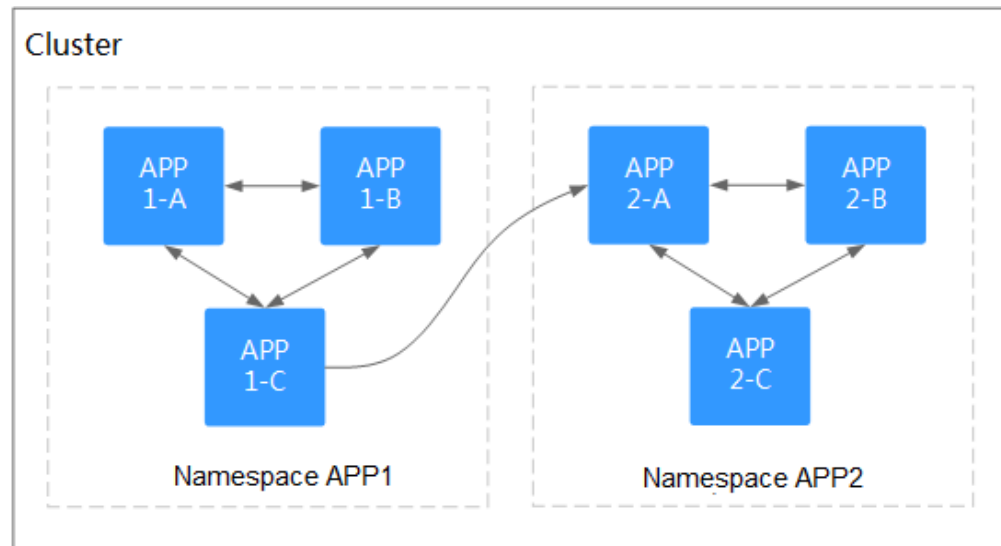
**Figure 14-1** One namespace for one environment




- **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 14-2** 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 **cicd**, indicating that the namespace is used to deploy CICD.
  - Deleting a label: Click  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

## 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 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

## 14.3 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.

### 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 if you have enabled **enable-resource-quota** in [Cluster Configuration Management](#). [Table 14-2](#) lists the resource quotas based on cluster specifications. You can modify them according to your service requirements.

**Table 14-2** Default resource quotas

Cluster Scale	Pod	Deployment	Secret	ConfigMap	Service
50 nodes	2000	1000	1000	1000	1000
200 nodes	2000	1000	1000	1000	1000
1,000 nodes	5000	2000	2000	2000	2000
2,000 nodes	5000	2000	2000	2000	2000

## Constraints

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. The resource quota limits the total resource consumption of each namespace and records the resource information in the cluster. Therefore, after the **enable-resource-quota** option is enabled, the probability of resource creation conflicts increases in large-scale concurrency scenarios, affecting the performance of batch resource creation.

## Procedure

- 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 of the cluster console.
- Step 3** Click **Manage Quota** next to the target namespace.

This operation cannot be performed on system namespaces **kube-system** and **kube-public**.

- Step 4** Set the resource quotas and click **OK**.

---

**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

# 15 ConfigMaps and Secrets

---

## 15.1 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.


### Constraints

- The size of a ConfigMap resource file cannot exceed 2 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** Set parameters.

**Table 15-1** Parameters for creating a ConfigMap

Parameter	Description
Name	Name of a ConfigMap, 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  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>Add</b> .

**Step 4** After the configuration is complete, click **OK**.

The new ConfigMap is displayed in the ConfigMap list.

----End

## 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 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 15-2** 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 configuration item, you can update or delete it as described in [Table 15-3](#).

**Table 15-3** 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 15-1</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.

## 15.2 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 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.

## Setting Workload Environment Variables

### 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**. Then, click **Create Workload**.

When creating a workload, click **Environment Variables** in the **Container Settings** area, and click **+**.

- **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 connect to 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 a 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 a 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 variable 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

## Setting 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 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**.

When creating a workload, click **Environment Variables** in the **Container Settings** area, and click **+**. 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, that is, the key names in the **cce-configmap** ConfigMap.

```
/bin/bash
-c
echo $SPECIAL_LEVEL $SPECIAL_TYPE > /usr/share/nginx/html/index.html
```

**Step 4** Set 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 connect to 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, the `cce-configmap` ConfigMap is imported to the workload. `SPECIAL_LEVEL` and `SPECIAL_TYPE` are the environment variable names in the workload, that is, the 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** After the workload runs properly, the following content is entered into 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
```

## Attaching 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 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**.

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** Configure the parameters.

**Table 15-4** Mounting a ConfigMap volume

Parameter	Description
ConfigMap	Select the desired ConfigMap. A ConfigMap must be created in advance. For details, see <a href="#">Creating a ConfigMap</a> .

Parameter	Description
Add Container Path	<p>Configure the following parameters:</p> <ol style="list-style-type: none"> <li>1. <b>Mount Path:</b> Enter a path of the container, for example, <code>/tmp</code>. 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 <code>/</code> or <code>/var/run</code>; this action may cause container errors. You are advised to 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, causing 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 machine may be damaged.</li> <li>2. <b>Subpath:</b> Enter a subpath, for example, <code>tmp</code>. <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 is 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> </li> <li>3. Set the permission to <b>Read-only</b>. Data volumes in the path are read-only.</li> </ol> <p>You can click  to add multiple paths and subpaths.</p>

----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 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 are 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-*** -- /etc/config/SPECIAL_LEVEL
```

Expected output:

```
Hello
```

----End

## 15.3 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.

### 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** Set parameters.

**Table 15-5** 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>• <b>IngressTLS</b>: 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>
Secret Data	Workload secret data can be used in containers. <ul style="list-style-type: none"> <li>• If <b>Secret Type</b> is <b>Opaque</b>, click . 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 <code>kubernetes.io/dockerconfigjson</code>, enter the account and password for logging in to the private image repository.</li> <li>• If <b>Secret Type</b> is <code>kubernetes.io/tls</code> 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>Add</b> .

**Step 4** After the configuration is complete, 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:

- 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.

- 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=
```

- Use Base64 to encode the following JSON content:

```
echo -n '{"auths":{"address":
{"username":"username","password":"password","auth":"dXNlcm5hbWU6cGFzc3dvcmQ="}}}'
| base64
```

Command output:

```
eyJhdXRocyl6eyJhZGRyZXNzIjp7InVzZXJhZG91bnRlclljb2R2Y21RPSJ9fX0=
```

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** 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**

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 15-6](#).

### NOTE

The secret list contains system secret resources that can be queried only. The system secret resources cannot be updated or deleted.

**Table 15-6** 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 15-5</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

```

## 15.4 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.

- [Setting 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.

## Setting Environment Variables of a Workload

### 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**. Then, click **Create Workload**.

When creating a workload, click **Environment Variables** in the **Container Settings** area, and click **+**.

- **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** Set 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 connect to 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 a 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 a 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
 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 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 on the left, click **Workloads**. 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** Configure the parameters.

**Table 15-7** Mounting a Secret volume

Parameter	Description
Secret	Select the desired secret. A secret must be created in advance. For details, see <a href="#">Creating a Secret</a> .

Parameter	Description
Add Container Path	<p>Configure the following parameters:</p> <ol style="list-style-type: none"> <li>1. <b>Mount Path:</b> Enter a path of the container, for example, <code>/tmp</code>. 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 <code>/</code> or <code>/var/run</code>; this action may cause container errors. You are advised to 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, causing 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 machine may be damaged.</li> <li>2. <b>Subpath:</b> Enter a subpath, for example, <code>tmp</code>. <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 is 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> </li> <li>3. Set the permission to <b>Read-only</b>. Data volumes in the path are read-only.</li> </ol> <p>You can click  to add multiple paths and subpaths.</p>

----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 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:

```
kubect exec nginx-secret-*** -- /etc/foo/username
```

The expected output is the same as the content in the secret.

----End

## 15.5 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 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.

```
$ kubect 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-xxxxx
Tokens: default-token-xxxxx
Events: <none>
```

# 16 Auto Scaling

---

## 16.1 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 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 components are described as follows:

Table 16-1 Workload scaling components

Type	Component Name	Component Description	Reference
HPA	<a href="#">Kubernetes 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="#">HPA</a>

Node scaling components are described as follows:

Table 16-2 Node scaling components

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 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>

## 16.2 Scaling a Workload

### 16.2.1 Workload Scaling Mechanisms

#### 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 is integrated with [Metrics Server](#) to provide such capabilities:

- **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.

#### 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.

## 16.2.2 HPA

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 actual requirements.

- **Kubernetes Metrics Server**: provides basic resource usage metrics, such as container CPU and memory usage. It is supported by all cluster versions.

### 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 volume mounted, a new pod cannot be started because EVS disks cannot be attached.

### Creating an HPA 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 **Workload Scaling**. Then click **Create HPA Policy** in the upper right corner.
- Step 3** Configure the parameters.

**Table 16-3** HPA policy parameters

Parameter	Description
Policy Name	Name of the policy to be created. Set this parameter as required.
Namespace	Namespace to which the workload belongs.
Associated Workload	Workload with which the HPA policy is associated.

Parameter	Description
Pod Range	<p>Minimum and maximum numbers of pods.</p> <p>When a policy is triggered, the workload pods are scaled within this range.</p>
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> <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>



Parameter	Description
System Policy	<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>Desired Value:</b> Enter the desired average resource usage. This parameter indicates the desired value of the selected metric. 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. 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>
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.</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


## 16.2.3 Managing Workload Scaling Policies

### Scenario

After an HPA 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 and click the cluster name to access the cluster console.
- Step 2** In the navigation pane, choose **Workload Scaling**. On the **HPA Policies** tab, click  next to the target HPA policy.
- Step 3** In the expanded area, view the **Rule** and **Status** tabs. Click **View Events** in the **Operation** column. If the policy malfunctions, locate and rectify the fault based on the error message displayed on the page.

 **NOTE**

You can also view the created HPA 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, click the **Auto Scaling** tab to view the HPA policies. You can also view the scaling policies you configured in the **Workload Scaling** page.

**Table 16-4** 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.
	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

## Updating an HPA Policy

An HPA policy is used as an example.

- Step 1** Log in to the CCE console and click the cluster name to access the cluster console.
  - Step 2** On the cluster console, choose **Workload Scaling** in the navigation pane. Locate the row that contains the target policy and choose **More > Edit** in the **Operation** column.
  - Step 3** On the **Edit HPA Policy** page, configure the parameters as listed in [Table 16-3](#).
  - Step 4** Click **OK**.
- End

## Editing the YAML File (HPA 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 **Workload Scaling**, locate the row containing the target policy, and click **Edit YAML** in the **Operation** column.
  - Step 3** In the **Edit YAML** dialog box displayed, edit or download the YAML file.
- End

## Deleting an HPA 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 **Workload Scaling**. Locate the row containing the target policy and choose **More > Delete** in the **Operation** column.
  - Step 3** In the dialog box displayed, click **Yes**.
- End

# 16.3 Scaling a Node

## 16.3.1 Node Scaling Mechanisms

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.

**autoscaler** is an auto 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 **autoscaler** add-on of v1.13.8 or later.

## How autoscaler Works

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 user-defined percentage for scale-in, autoscaler 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.

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)

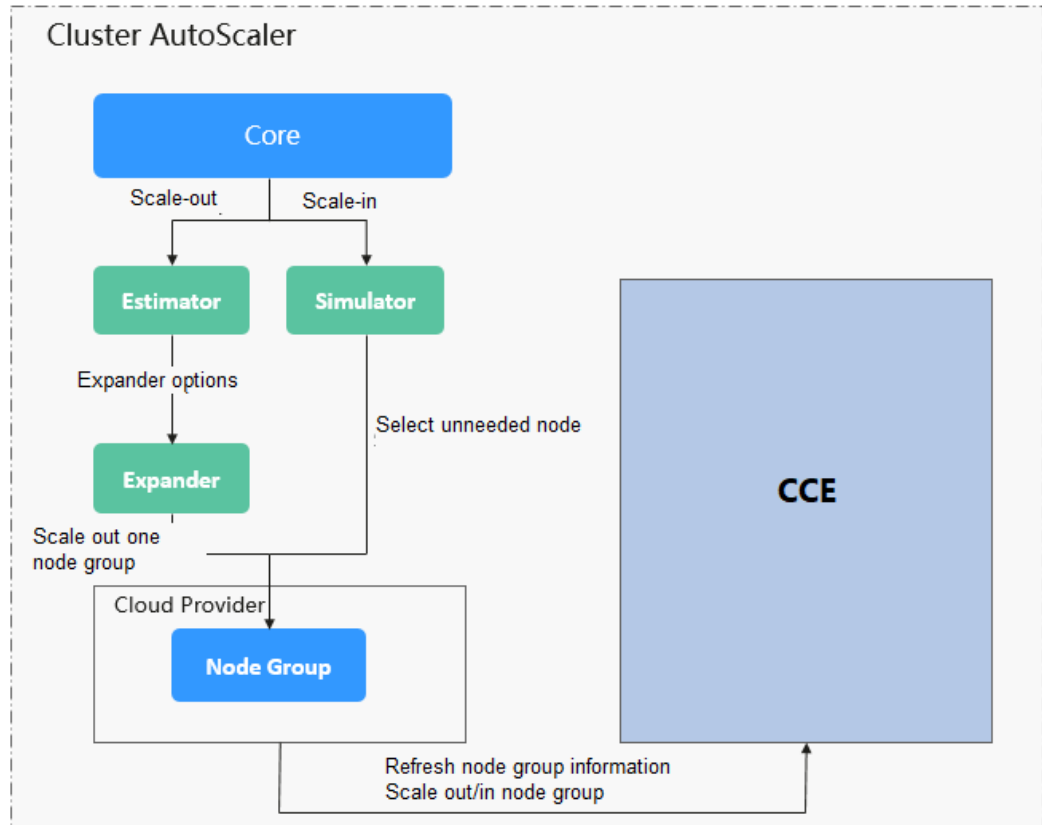
### NOTE

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.

## autoscaler Architecture

[Figure 16-1](#) shows the autoscaler architecture and its core modules:

Figure 16-1 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:

**Table 16-5 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 replicas 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 the 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 replicas 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. 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 replicas 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 and manage the priorities of node pools or scaling groups through the console or API, while the least-waste policy can reduce the resource waste ratio in common scenarios. This policy has wider applicability and is used as the default selection policy.</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 replicas 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>

### 16.3.2 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 in the cluster.

## Constraints

- Auto scaling policies apply to node pools. When the number of nodes in a node pool is 0 and the scaling policy is based on CPU or memory usage, node scaling is not triggered.
- Node scale-in will cause PVC/PV data loss for the [local PVs](#) associated with the node. These PVCs and PVs cannot be restored or used again. In a node scale-in, the pod that uses the local PV is evicted from the node. A new pod is created and stays 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.
- 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](#).

## Procedure

- Step 1** Log in to the CCE console and click the cluster name to access the cluster console.
- Step 2** Choose **Node Scaling** in the navigation pane.
  - If **Uninstalled** 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 **Installed** is displayed next to the add-on name, the add-on has been installed.
- Step 3** Click **Create Node Scaling Policy** in the upper right corner and set the parameters as follows:
  - **Policy Name**: name of the policy to be created, which can be customized.
  - **Associated Node Pools**: Select the node pool to be associated. You can associate multiple node pools to use the same scaling policy.
  - **Rules**: Click **Add Rule**. In the dialog box displayed, set the following parameters:
    - Rule Name**: Enter a rule name.
    - Rule Type**: You can select **Metric-based** or **Periodic**. The differences between the two types are as follows:

– **Metric-based:**

**Condition:** Select **CPU allocation rate** or **Memory allocation rate** 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.

– **Periodic:**

**Trigger Time:** You can select a specific time point every day, every week, every month, or every year.

**Action:** Set an action to be performed when the trigger condition is met.

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 4** Click **OK**.

----End

## Constraints on Scale-in

You can set node scale-in policies only 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).

## Example YAML

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 16-6** 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.

Parameter	Type	Description
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.
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.


### 16.3.3 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** Choose **Node Scaling** in the navigation pane and click  in front of the policy to be viewed.
- Step 3** In the expanded area, the **Associated Node Pools**, **Rules**, and **Scaling History** tab pages are displayed. If the policy is abnormal, locate and rectify the fault based on the error information.

 **NOTE**

You can also disable or enable auto scaling on the **Node Pools** page.

1. Log in to the CCE console and click the cluster name to access the cluster console.
2. In the navigation pane, choose **Nodes** and switch to the **Node Pools** tab.
3. Locate the row containing the target node pool and click **Update** in the **Operation** column. In the window that slides out from the right, enable **Auto Scaling**, and configure **Max. Nodes**, **Min. Nodes**, and **Cooldown Period**.

----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** Choose **Node Scaling** in the navigation pane and choose **More > Delete** next to the policy to be deleted.
- 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** Choose **Node Scaling** in the navigation pane and click **Edit** in the **Operation** column of the policy to be edited.
- Step 3** On the **Edit Node Scaling Policy** page displayed, modify policy parameter values listed in [Table 16-6](#).
- 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** Choose **Node Scaling** in the navigation pane and choose **More > Clone** next to the policy to be cloned.
- Step 3** On the **Clone 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** Choose **Node Scaling** in the navigation pane and click **Disable** in the **Operation** column of the policy to be disabled. If the policy is in the disabled state, click **Enable** in the **Operation** column of the policy.
- Step 3** In the dialog box displayed, confirm whether to disable or enable the node policy.  
----End

## 16.4 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 pods or containers are used for deploying applications, the upper limit of available resources is typically required to set for pods or containers to prevent unlimited usage of node resources during peak hours. However, after the upper limit is reached, an application error may occur. To resolve this issue, scale in the number of pods to share workloads. If the node resource usage increases to a certain extent that newly added pods cannot be scheduled, scale in the number of nodes based on the node resource usage.

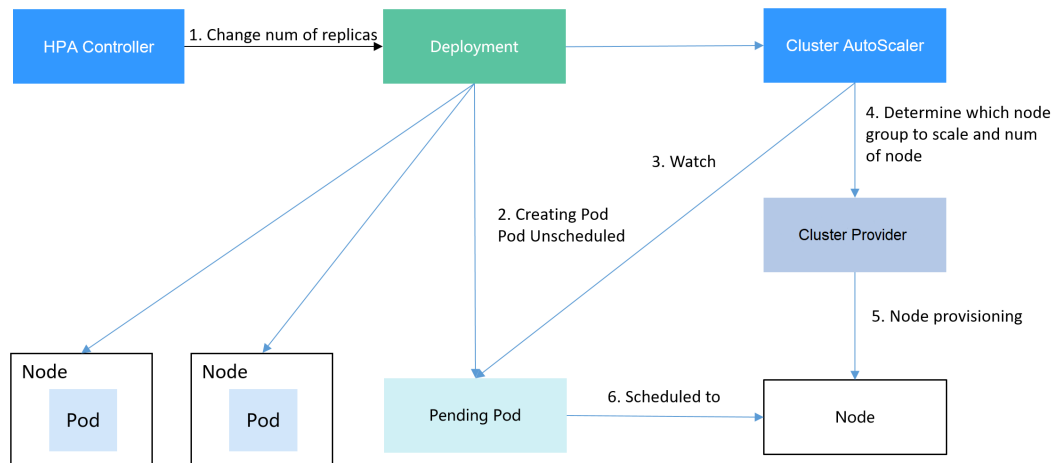
### Solution

Two major auto scaling policies are HPA (Horizontal Pod Autoscaling) and CA (Cluster AutoScaling). HPA is for workload auto scaling and CA is for node auto scaling.

HPA and CA work with each other. HPA requires sufficient cluster resources for successful scaling. When the cluster resources are insufficient, CA is needed to add nodes. If HPA reduces workloads, the cluster will have a large number of idle resources. In this case, CA needs to release nodes to avoid resource waste.

As shown in [Figure 16-2](#), 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 16-2** HPA and CA working flows



Using HPA and CA can easily implement auto scaling in most scenarios. In addition, the scaling process of nodes and pods can be easily observed.

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 CPU 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
```

Example file content:

```
<?php
$x = 0.0001;
for ($i = 0; $i <= 1000000; $i++) {
 $x += sqrt($x);
}
echo "OK!";
?>
```

2. Compile a Dockerfile to build an image.

```
vi Dockerfile
```

Example Dockerfile:




```
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 **Organization Management** in the navigation pane, and click **Create Organization** in the upper right corner to create an organization.

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. Add a tag for 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.

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]
```

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** Set node pool parameters, add a node with 2 vCPUs and 4 GB memory, and enable auto scaling.

- **Nodes:** Set it to **1**, indicating that one node is created by default when a node pool is created.
- **Auto Scaling:** Enable the option, meaning that nodes will be automatically created or deleted in the node pool based on the cluster loads.
- **Max. Nodes:** Set it to **5**, indicating the maximum number of nodes in a node pool.
- **Specifications:** 2 vCPUs | 4 GiB

Retain the defaults for other parameters.

- Step 3** Click **Add-ons** on the left of the cluster console, click **Edit** under the autoscaler add-on, modify the add-on configuration, enable **Auto node scale-in**, and configure scale-in parameters. For example, trigger scale-in when the node resource utilization is less than 50%.

After the preceding configurations, scale-out is performed based on the pending status of the pod and scale-in is triggered when the node resource utilization decreases.

- Step 4** Click **Node Scaling** on the left of the cluster console and click **Create Node Scaling Policy** in the upper right corner. Node scaling policies added here trigger scale-out based on the CPU/memory allocation rate or periodically.

As shown in the following figure, when the cluster CPU allocation rate is greater than 70%, one node will be added. 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.

----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 Pending state. During this period, nodes are added.

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

Using HPA and CA can easily implement auto scaling in most scenarios. In addition, the scaling process of nodes and pods can be easily observed.

# 17 Add-ons

## 17.1 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.

Table 17-1 Add-on list

Add-on Name	Introduction
<b>CoreDNS</b>	CoreDNS is a DNS server that provides domain name resolution for Kubernetes clusters through chain plug-ins.
<b>CCE Container Storage (FlexVolume)</b>	storage-driver is a FlexVolume driver used to support IaaS storage services such as EVS, SFS, and OBS.
<b>CCE Container Storage (Everest)</b>	everest 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).
<b>npd</b>	node-problem-detector (npd for short) 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. The npd add-on can run as a DaemonSet or a daemon.

Add-on Name	Introduction
<b>CCE Cluster Autoscaler</b>	The autoscaler add-on resizes a cluster based on pod scheduling status and resource usage.
<b>Kubernetes Metrics Server</b>	metrics-server is an aggregator for monitoring data of core cluster resources.
<b>gpu-device-plugin</b>	gpu-device-plugin is a device management add-on that supports GPUs in containers. It supports only NVIDIA drivers.
<b>Volcano Scheduler</b>	Volcano provides general-purpose, high-performance computing capabilities, 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>NGINX Ingress Controller</b>	nginx-ingress 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.

## Add-on Lifecycle

An add-on lifecycle involves all the statuses of the add-on from installation to uninstallation.

**Table 17-2** 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.
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.



Status	Attribute	Description
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.

## Related Operations

You can perform the operations described in [Table 17-3](#) on the **Add-ons** page.

**Table 17-3** 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>

Operation	Description	Procedure
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>

## 17.2 CoreDNS

### Introduction

CoreDNS 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 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](#).

## 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 the add-on pods and all the add-on pods 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 and click the cluster name to access the cluster console. Choose **Add-ons** in the navigation pane, locate **CoreDNS** on the right, and click **Install**.
- Step 2** On the **Install Add-on** page, configure the specifications.

**Table 17-4** Add-on configuration

Parameter	Description
Add-on Specifications	Concurrent domain name resolution ability. Select add-on specifications that best fit your needs.  If you select <b>Custom qps</b> , the domain name resolution QPS provided by CoreDNS is positively correlated with the CPU consumption. Adjust the number of pods and container CPU/memory quotas as required.
Pods	Number of pods that will be created to match the selected add-on specifications.  If you select <b>Custom qps</b> , you can adjust the number of pods as required.
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 that AZ.</li> <li>• <b>Required:</b> Deployment pods of the add-on will be forcibly scheduled to nodes in different AZs. If there are fewer AZs than pods, the extra pods will fail to run.</li> </ul>

Parameter	Description
Containers	CPU and memory quotas of the container allowed for the selected add-on specifications. If you select <b>Custom qps</b> , you can adjust the container specifications as required.

**Step 3** Configure the add-on parameters.

**Table 17-5** Add-on parameters

Parameter	Description
Stub domain settings	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
Advanced settings	<ul style="list-style-type: none"> <li>● <b>parameterSyncStrategy</b>: indicates whether to configure consistency check when an 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 recorded in the cluster is inconsistent with the actual configuration, the add-on cannot be upgraded.</li> <li>– <b>force</b>: indicates that the configuration consistency check is ignored during an upgrade. Ensure that the current effective configuration is the same as the original configuration. After the add-on is upgraded, restore the value of <b>parameterSyncStrategy</b> to <b>ensureConsistent</b> and enable the configuration consistency check again.</li> <li>– <b>inherit</b>: indicates that differentiated configurations are automatically inherited during an upgrade. After the add-on is upgraded, restore the value of <b>parameterSyncStrategy</b> to <b>ensureConsistent</b> and enable the configuration consistency check again.</li> </ul> </li> <li>● <b>stub_domains</b>: 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.</li> <li>● <b>upstream_nameservers</b>: IP address of the upstream DNS server.</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 17-6</b> describes common plugins. For details, see <a href="#">Plugins</a>.</p> <p>Example:</p> <pre style="background-color: #f0f0f0; padding: 5px;">{   "servers": [     {       "plugins": [         {           "name": "bind",           "parameters": "\${POD_IP}"         },         {           "name": "cache",           "parameters": 30         }       ]     }   ] }</pre> </li> </ul>

Parameter	Description
	<pre> {   "name": "errors" }, {   "name": "health",   "parameters": "{\$POD_IP}:8080" }, {   "name": "ready",   "{\$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": "."   } ] }, "stub_domains": {   "acme.local": [     "1.2.3.4",     "6.7.8.9"   ] }, "upstream_nameservers": ["8.8.8.8", "8.8.4.4"] } </pre>

**Table 17-6** Default plugin configuration of the active zone of CoreDNS

plugin Name	Description
bind	Host IP address listened by CoreDNS. You are advised to retain the default value <b>{SPOD_IP}</b> . For details, see <a href="#">bind</a> .
cache	DNS cache is enabled. For details, see <a href="#">cache</a> .
errors	Errors are logged to stdout. For details, see <a href="#">errors</a> .

plugin Name	Description
health	Health check configuration. The current listening IP address is <code>{\$POD_IP}:8080</code> . Retain the default setting. Otherwise, the CoreDNS health check fails and CoreDNS restarts repeatedly. For details, see <a href="#">health</a> .
ready	Whether the backend server is ready to receive traffic. The current listening port is <code>{\$POD_IP}:8081</code> . If the backend server is not ready, CoreDNS suspends DNS resolution until the backend server is ready. For details, see <a href="#">ready</a> .
kubernetes	CoreDNS Kubernetes plug-in, which provides the service parsing capability in a cluster. For details, see <a href="#">kubernetes</a> .
loadbalance	Round-robin DNS load balancer that randomizes the order of A, AAAA, and MX records in the answer. For details, see <a href="#">loadbalance</a> .
prometheus	Port for obtaining CoreDNS metrics. The default zone listening IP address is <code>{\$POD_IP}:9153</code> . Retain the default setting. Otherwise, prometheus cannot collect CoreDNS metrics. For details about, see <a href="#">prometheus</a> .
forward	Any queries that are not within the cluster domain of Kubernetes will be forwarded to predefined resolvers ( <code>/etc/resolv.conf</code> ). For details, see <a href="#">forward</a> .
reload	The changed Corefile can be automatically reloaded. After editing the ConfigMap, wait for 2 minutes for the modification to take effect. For details, see <a href="#">reload</a> .

**Step 4** Click **Install**.

----End

## Components

**Table 17-7** CoreDNS components

Container Component	Description	Resource Type
CoreDNS	DNS server for clusters	Deployment

## 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

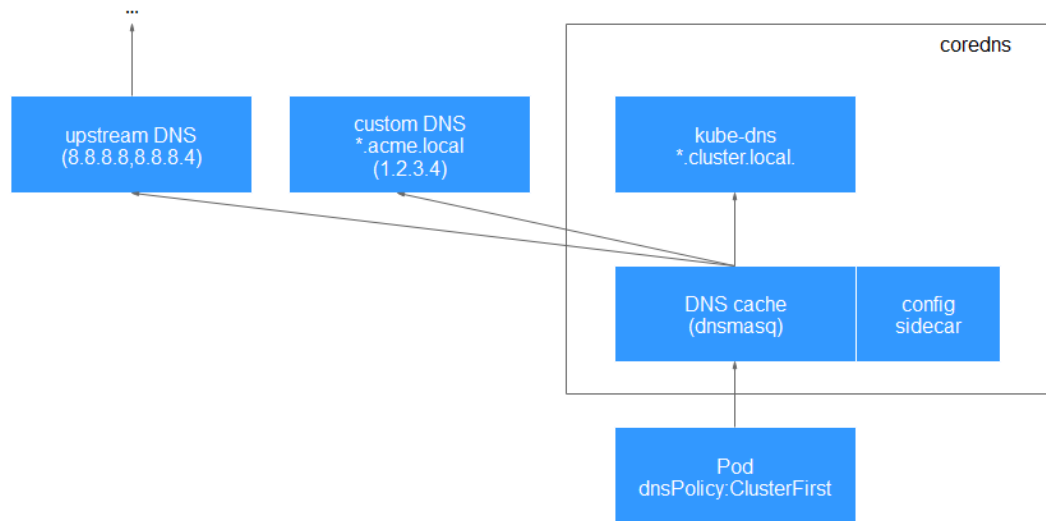
**Without stub domain configurations:** Any query that does not match the configured cluster domain suffix, such as **www.kubernetes.io**, is forwarded to the upstream DNS server inherited from the node.

**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 17-1 Routing



## 17.3 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 Container Storage Interface.

**everest is a system resource add-on. It is installed by default when a cluster of Kubernetes v1.15 or later is created.**

### Constraints

- If your cluster is upgraded from v1.13 to v1.15, **storage-driver** is replaced by everest (v1.1.6 or later) for container storage. The takeover does not affect the original storage functions.
- 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.
- By default, this add-on is installed in **clusters of v1.15 and later**. For clusters of v1.13 and earlier, the **storage-driver** add-on is installed by default.

### 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. Choose **Add-ons** in the navigation pane, locate **CCE Container Storage (Everest)** on the right, and click **Install**.

**Step 2** On the **Install Add-on** page, configure the specifications.

**Table 17-8** Add-on configuration

Parameter	Description
Pods	<p>Number of pods for the add-on.</p> <p>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.</p>
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 that AZ.</li> <li>• <b>Required:</b> Deployment pods of the add-on will be forcibly scheduled to nodes in different AZs. If there are fewer AZs than pods, the extra pods will fail to run.</li> </ul>
Containers	<p>The everest add-on contains the everest-csi-controller and everest-csi-driver components. For details, see <a href="#">Components</a>.</p> <p>After you select the add-on specification, the requested CPU and memory of the add-on component are displayed. For details, see <a href="#">Table 17-9</a>.</p> <p>In non-typical scenarios, the formulas for estimating the limit values are as follows:</p> <ul style="list-style-type: none"> <li>• everest-csi-controller <ul style="list-style-type: none"> <li>- CPU limit: 250m for 200 or fewer nodes, 350m for 1000 nodes, and 500m for 2000 nodes</li> <li>- Memory limit = (200 MiB + Number of nodes x 1 MiB + Number of PVCs x 0.2 MiB) x 1.2</li> </ul> </li> <li>• everest-csi-driver <ul style="list-style-type: none"> <li>- CPU limit: 300 m for 200 or fewer nodes, 500 m for 1000 nodes, and 800 m for 2000 nodes</li> <li>- Memory limit: 300 MiB for 200 or fewer nodes, 600 MiB for 1000 nodes, and 900 MiB for 2000 nodes</li> </ul> </li> </ul>

**Table 17-9** Recommended configuration limits in typical scenarios

Configuration Scenario			everest-csi-controller		everest-csi-driver	
Nodes	PVs/PVCs	Add-on Instances	CPU (The limit value is the same as the requested value.)	Memory (The limit value is the same as the requested value.)	CPU (The limit value is the same as the requested value.)	Memory (The limit value is the same as the requested value.)

Configuration Scenario			everest-csi-controller		everest-csi-driver	
50	1000	2	250m	600Mi	300m	300Mi
200	1000	2	250m	1Gi	300m	300Mi
1000	1000	2	350m	2Gi	500m	600Mi
1000	5000	2	450m	3Gi	500m	600Mi
2000	5000	2	550m	4Gi	800m	900Mi
2000	10,000	2	650m	5Gi	800m	900Mi

**Step 3** Configure the add-on parameters.

**Table 17-10** Add-on parameters

Parameter	Description
csi_attacher_worker_threads	Number of worker nodes that can concurrently attach EVS volumes. The default value is <b>60</b> .
csi_attacher_detach_worker_threads	Number of worker nodes that can concurrently detach EVS volumes. The default value is <b>60</b> .
volume_attaching_flow_ctrl	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.
cluster_id	Cluster ID
default_vpc_id	ID of the VPC to which the cluster belongs
disable_auto_mount_secret	Whether the default AK/SK can be used when an object bucket or parallel file system is mounted. The default value is <b>false</b> .
enable_node_attacher	Whether to enable the attacher on the agent to process the <a href="#">VolumeAttachment</a> .
flow_control	This field is left blank by default. You do not need to configure this parameter.
over_subscription	Overcommitment ratio of the local storage pool ( <b>local_storage</b> ). The default value is <b>80</b> . If the size of the local storage pool is 100 GB, it can be overcommitted to 180 GB.
project_id	ID of the project to which a cluster belongs

 **NOTE**

In everest 1.2.26 or later, the performance of attaching a large number of EVS volumes has been optimized. The following parameters can be configured:

- `csi_attacher_worker_threads`
- `csi_attacher_detach_worker_threads`
- `volume_attaching_flow_ctrl`

The preceding parameters are associated with each other and are constrained by the underlying storage resources in the region where the cluster is located. To attach a large number of volumes (more than 500 EVS volumes per minute), contact customer service and configure the parameters under their guidance to prevent the everest add-on from running abnormally due to improper parameter settings.

**Step 4** Click **Install**.

----End

## Components

**Table 17-11** everest components

Container 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
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

## 17.4 npd

### Introduction

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. The npd add-on can run as a DaemonSet or a daemon.

For more information, see [node-problem-detector](#).

## Constraints

- When using this add-on, do not format or partition node disks.
- Each npd process occupies 30 m CPU and 100 MB memory.

## 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 access the cluster console. Choose **Add-ons** in the navigation pane, locate **npd** on the right, and click **Install**.

**Step 2** On the **Install Add-on** page, configure the specifications.

**Table 17-12** Add-on configuration

Parameter	Description
Add-on Specifications	The specifications can be <b>Custom</b> .
Pods	If you select <b>Custom</b> , you can adjust the number of pods as required.
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 that AZ.</li> <li>• <b>Required</b>: Deployment pods of the add-on will be forcibly scheduled to nodes in different AZs. If there are fewer AZs than pods, the extra pods will fail to run.</li> </ul>
Containers	If you select <b>Custom</b> , you can adjust the container specifications as required.

**Step 3** Configure the add-on parameters.

Only v1.16.0 and later versions support the configurations.

**Table 17-13** Add-on parameters

Parameter	Description
common.image.pullPolicy	An image pulling policy. The default value is <b>IfNotPresent</b> .
feature_gates	A feature gate
npc.maxTaintedNode	The maximum number of nodes that npc can add taints to when a single fault occurs on multiple nodes for minimizing impact The value can be in int or percentage format.
npc.nodeAffinity	Node affinity of the controller

 **NOTE**

Only some parameters are listed here. For more information, see the details of the open-source project node-problem-detector.

**Step 4** Click **Install**.

----End

## Components

**Table 17-14** npd components

Container Component	Description	Resource Type
node-problem-controller	Isolate faults basically based on fault detection results.	Deployment
node-problem-detector	Detect node faults.	Daemon Set

## 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 17-15** 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>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 17-16** 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 CRIProblem	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 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>
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 17-17** Checking system metrics

Check Item	Function	Description
Contrack table full ContrackFullProblem	Check whether the contrack table is full.	<ul style="list-style-type: none"> <li>• Default threshold: 90%</li> <li>• Usage: <b>nf_contrack_count</b></li> <li>• Maximum value: <b>nf_contrack_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>
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 <code>/proc/sys/fs/file-nr</code></li> <li>• Maximum value: the third value in <code>/proc/sys/fs/file-nr</code></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 <code>/proc/meminfo</code></li> <li>• Maximum value: <b>MemTotal</b> in <code>/proc/meminfo</code></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 <code>/proc/loadavg</code></li> <li>• Maximum value: smaller value between <code>/proc/sys/kernel/pid_max</code> and <code>/proc/sys/kernel/threads-max</code>.</li> </ul>

**Table 17-18** Checking the storage

Check Item	Function	Description
Disk read-only DiskReadyonly	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>
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>

Check Item	Function	Description
<p>emptyDir storage pool error</p> <p>EmptyDirVolumeGroupStatus Error</p>	<p>Check whether the ephemeral volume group on the node is normal.</p> <p>Impact: The pod that depends 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 a temporary volume storage pool. The user deletes some data disks 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>
<p>Suspended disk I/O</p> <p>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</li> </ul> <p>Alternatively, you can run the following command:</p> <pre>iostat -xmt 1</pre> <ul style="list-style-type: none"> <li>• Threshold: <ul style="list-style-type: none"> <li>- Average usage: ioutil &gt;= 0.99</li> <li>- Average I/O queue length: avgqu-sz &gt;= 1</li> <li>- Average I/O transfer volume: iops (w/s) + ioth (wMB/s) &lt;= 1</li> </ul> </li> </ul> <p><b>NOTE</b></p> <p>In some OSs, no data changes during I/O. In this case, calculate the CPU I/O time usage. The value of iowait 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: <code>/proc/diskstat</code> Alternatively, you can run the following command: <code>iostat -xmt 1</code></li> <li>• Default threshold: Average I/O latency: <code>await &gt;= 5000 ms</code></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 17-19** Other check items

Check Item	Function	Description
Abnormal NTP NTPProblem	Check whether the node clock synchronization service <code>ntpd</code> or <code>chronyd</code> 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>• <code>/proc/{PID}/stat</code></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
<p>ResolvConf error</p> <p>ResolvConfFileProblem</p>	<p>Check whether the ResolvConf file is lost.</p> <p>Check whether the ResolvConf file is normal.</p> <p>Exceptional definition: No upstream domain name resolution server (nameserver) is included.</p>	<p>Object: <b>/etc/resolv.conf</b></p>
<p>Existing scheduled event</p> <p>ScheduledEvent</p>	<p>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.</p> <p>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.</p>	<p>Source:</p> <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> <p>This check item is an Alpha feature and is disabled by default.</p>

The kubelet component has the following default check items, which have bugs or defects. You can fix them by upgrading the cluster or using npd.

**Table 17-20** Default kubelet check items

Check Item	Function	Description
<p>Insufficient PID resources</p> <p>PIDPressure</p>	<p>Check whether PIDs are sufficient.</p>	<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 plug-in 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 17-21** 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	Check how many nodes can npc add taints to for mitigating the impact when a single fault occurs on multiple nodes. The int format and percentage format are supported.	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

## 17.5 CCE Cluster Autoscaler

### Introduction

CCE Cluster 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 in 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 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.



Auto scale-out will be performed when:

- Node resources are insufficient.
- No node affinity policy is set in the pod scheduling configuration. 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 [Scheduling Policy \(Affinity/Anti-affinity\)](#).
- 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.

## 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. Choose **Add-ons** in the navigation pane, locate **CCE Cluster Autoscaler** on the right, and click **Install**.

**Step 2** On the **Install Add-on** page, configure the specifications.

**Table 17-22** Add-on configuration

Parameter	Description
Add-on Specifications	<p>The add-on can be deployed in the following specifications:</p> <p><b>NOTE</b> When the autoscaler add-on is deployed in HA or customized mode, anti-affinity policies exist between add-on instances and the add-on instances are deployed on different nodes. Therefore, the number of available nodes in the cluster must be greater than or equal to the number of add-on instances to ensure high availability of the add-on.</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 200 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>
Pods	<p>Number of pods that will be created to match the selected add-on specifications.</p> <p>If you select <b>Custom</b>, you can adjust the number of pods as required.</p>
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 that AZ.</li> <li>• <b>Required:</b> Deployment pods of the add-on will be forcibly scheduled to nodes in different AZs. If there are fewer AZs than pods, the extra pods will fail to run.</li> </ul>

Parameter	Description
Containers	CPU and memory quotas of the container allowed for the selected add-on specifications. If you select <b>Custom</b> , you can adjust the container specifications as required.

**Step 3** Configure the add-on parameters.

**Table 17-23** Add-on parameters

Parameter	Description
Scaling	<p>You can select the following options as required:</p> <ul style="list-style-type: none"> <li>● <b>Nodes are automatically added (from the node pool) when pods in the cluster cannot be scheduled.</b> That is, when a pod is in <b>Pending</b> state, automatic scale-out is performed. If a node has been configured as an affinity node for pods, no node will not be automatically added when pods cannot be scheduled. Generally, an HPA policy works with such scaling. For details, see <a href="#">Using HPA and CA for Auto Scaling of Workloads and Nodes</a>. If this parameter is not selected, scaling can be performed only through <a href="#">node scaling policies</a>.</li> <li>● Auto node scale-in <ul style="list-style-type: none"> <li>– <b>Node 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>Scale-in Threshold:</b> If the percentage of both requested CPU and memory 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>Stabilization Window (s)</b> How long after a scale-out that a scale-in evaluation resumes. Default value: 10 minutes.</li> </ul> <p><b>NOTE</b> If both auto scale-out and scale-in exist in a cluster, set <b>How long after a scale-out that a scale-in evaluation resumes</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> <p>How long after the node deletion that a scale-in evaluation resumes. Default value: 10 minutes.</p> <p>How long after a scale-in failure that a scale-in evaluation resumes. 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 Cool-Down Period</a>.</p></li> <li>– <b>Max. Nodes for Batch Deletion:</b> Maximum number of empty nodes that can be deleted at the same time. Default value: 10. This feature applies only to idle nodes. Idle nodes can be concurrently scaled in. Nodes that are not idle can only be scaled in one by one.</li> </ul> <p><b>NOTE</b> During node scale-in, if the pod on the node does not need to be evicted (such as the pods of DaemonSet), the node is idle. Otherwise, the node is not idle.</p>

Parameter	Description
	<ul style="list-style-type: none"> <li>- <b>Check Interval:</b> Interval for checking again a node that could not be removed before. Default value: 5 minutes.</li> </ul>
Total Nodes	Maximum number of nodes that can be managed by the cluster, within which cluster scale-out is performed.
Total CPUs	Maximum sum of CPU cores of all nodes in a cluster, within which cluster scale-out is performed.
Total Memory (GB)	Maximum sum of memory of all nodes in a cluster, within which cluster scale-out is performed.

**Step 4** After the configuration is complete, click **Install**.

----End

## Components

**Table 17-24** autoscaler

Container Component	Description	Resource Type
autoscaler	Auto scaling for Kubernetes clusters	Deployment

## Scale-In Cool-Down Period

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.

## 17.6 NGNIX 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 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 nginx-ingress Works](#).

You can visit the [open source community](#) for more information.

#### NOTE

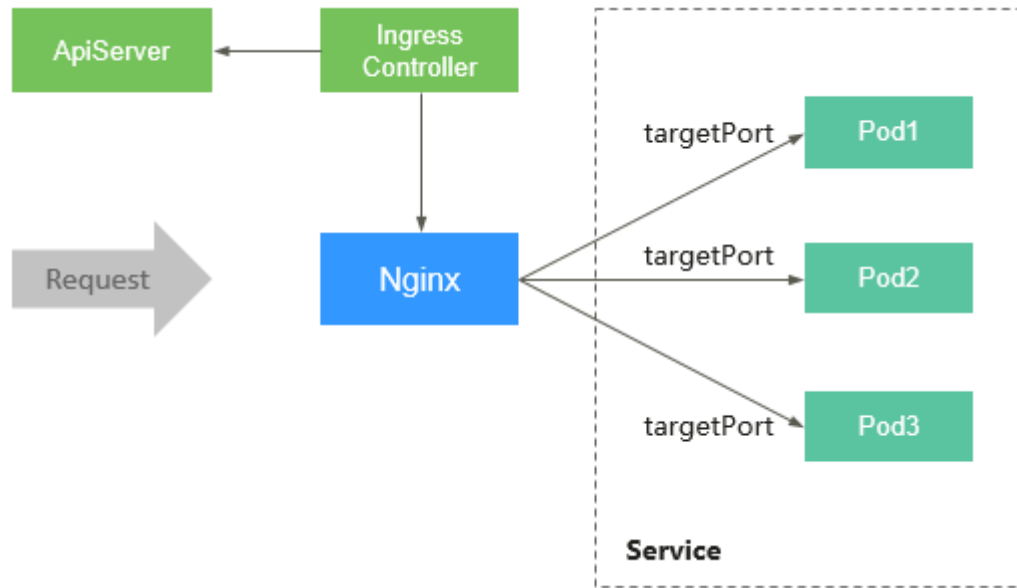
- When installing the add-on, you can add configurations by defining the Nginx configuration. The configurations take effect globally. This parameter is generated by configuring the `nginx.conf` file and affects all managed ingresses. You can search for related parameters in the [ConfigMap](#). If the 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, 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 17-2](#) 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 17-2 Working principles of nginx-ingress



## Constraints

- This add-on can be installed only in CCE clusters of v1.15 or later.
- **kubernetes.io/ingress.class: "nginx"** must be added to the annotation of the Ingress created through the API.
- Dedicated load balancers must be the network type (TCP/UDP) supporting private networks (with a private IP).
- The node where nginx-ingress-controller is running and the containers running on the node cannot access Nginx Ingress. In this case, perform anti-affinity deployment for the workloads and nginx-ingress-controller. For details, see [Anti-affinity Deployment for Workloads and nginx-ingress-controller](#).

## Prerequisites

Before creating a workload, you must have an available cluster. If no cluster is available, create one according to [Creating a Cluster](#).

## Installing the Add-on

### NOTE

- The [CVE-2021-25746](#) vulnerability has been fixed in nginx-ingress-controller of v1.2.0 (corresponding to CCE nginx-ingress add-on 2.1.0). **Rules** are added to disable some annotations prone to unauthorized access.
- The [CVE-2021-25745](#) vulnerability has been fixed in nginx-ingress-controller of v1.2.0 (corresponding to CCE nginx-ingress add-on 2.1.0). **Rules** are added to disable some access paths prone to unauthorized access.

**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, locate **NGINX Ingress Controller** on the right, and click **Install**.

**Step 2** On the **Install Add-on** page, configure the specifications.

**Table 17-25** Add-on configuration

Parameter	Description
Add-on Specifications	The specifications can be <b>Custom</b> .
Pods	If you select <b>Custom</b> , you can adjust the number of pods as required.
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 that AZ.</li> <li>• <b>Required:</b> Deployment pods of the add-on will be forcibly scheduled to nodes in different AZs. If there are fewer AZs than pods, the extra pods will fail to run.</li> </ul>
Containers	If you select <b>Custom</b> , you can adjust the container specifications as required.

**Step 3** Configure the add-on parameters.

- **Load Balancer:** Select a shared or dedicated load balancer. If no load balancer is available, create one first. The load balancer has at least two listeners, and ports 80 and 443 are not occupied by listeners.
- **Nginx Parameters:** Configuring the `nginx.conf` file will affect all managed ingresses. You can search for related parameters through [ConfigMaps](#). If the parameters you configured are not included in the options listed in those ConfigMaps, the parameters will not take effect.

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"
}
```

- **Default 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** Click **Install**.

----End



## Components

**Table 17-26** nginx-ingress component

Container Component	Description	Resource Type
nginx-ingress	Nginx Ingress controller, which provides flexible routing and forwarding for clusters	Deployment

### Anti-affinity Deployment for Workloads and nginx-ingress-controller

The node where nginx-ingress-controller is running and the containers running on the node cannot access Nginx Ingress. To prevent this problem, configure an anti-affinity rule to tell the scheduler not to co-locate the workload and nginx-ingress-controller on the same node.

```

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:
 - key: app
 operator: In
 values:
 - nginx-ingress
 - key: component
 operator: In
 values:
 - controller
 namespaces:
 - kube-system
 topologyKey: kubernetes.io/hostname

```

## 17.7 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 metrics-server is installed, you can create an HPA policy on the **Workload Scaling** tab page of the **Auto Scaling** page. For details, see [HPA](#).

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. Choose **Add-ons** in the navigation pane, locate **Kubernetes Metrics Server** on the right, and click **Install**.
- Step 2** On the **Install Add-on** page, configure the specifications.

**Table 17-27** Add-on configuration

Parameter	Description
Add-on Specifications	Select <b>Single</b> , <b>Custom</b> , or <b>HA</b> for <b>Add-on Specifications</b> .
Pods	Number of pods that will be created to match the selected add-on specifications. If you select <b>Custom</b> , you can adjust the number of pods as required.
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 that AZ.</li> <li>• <b>Required:</b> Deployment pods of the add-on will be forcibly scheduled to nodes in different AZs. If there are fewer AZs than pods, the extra pods will fail to run.</li> </ul>
Containers	CPU and memory quotas of the container allowed for the selected add-on specifications. If you select <b>Custom</b> , you can adjust the container specifications as required.

- Step 3** Click **Install**.

----End

## Components

**Table 17-28** metrics-server components

Container 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

## 17.8 gpu-device-plugin

### Introduction

The `gpu-device-plugin` add-on is a device management add-on that supports GPUs in containers. If GPU nodes are used in the cluster, this add-on must be installed.

### 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 CDUA library version of your application. You need to check the compatibility by yourself.
- If a GPU driver has been added to a custom OS image, CCE cannot ensure that the GPU driver is compatible with other GPU components such as the monitoring components used in CCE.

### Installing the 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, locate **gpu-device-plugin** on the right, and click **Install**.

**Step 2** On the **Install Add-on** page, configure the specifications.

**Table 17-29** Add-on configuration

Parameter	Description
Add-on Specifications	Select <b>Default</b> or <b>Custom</b> .
Containers	CPU and memory quotas of the container allowed for the selected add-on specifications.  If you select <b>Custom</b> , you can adjust the container specifications as required.

**Step 3** Configure the add-on parameters.

- **NVIDIA Driver:** Enter the link for downloading the NVIDIA driver. All GPU nodes in the cluster will use this driver.

---

**NOTICE**

- If the download link is a public network address, for example, **https://us.download.nvidia.com/tesla/470.103.01/NVIDIA-Linux-x86\_64-470.103.01.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.

- **Driver Selection:** 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.

 **NOTE**

- The add-on installs the driver with the version specified by the node pool. The driver takes effect only for new pool nodes.
- After the driver version of new nodes is updated, restart the nodes for the update to take effect. The driver version cannot be updated for non-new nodes.

**Step 4** Click **Install**.

 **NOTE**

Uninstalling the add-on will clear the GPU driver on the nodes. As a result, GPU pods newly scheduled to the nodes cannot run properly, but running GPU pods are not 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 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 17-3](#). **Operating System** must be **Linux 64-bit**.

**Figure 17-3** 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 ?

**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 17-4**. Click **DOWNLOAD**.

**Figure 17-4** 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

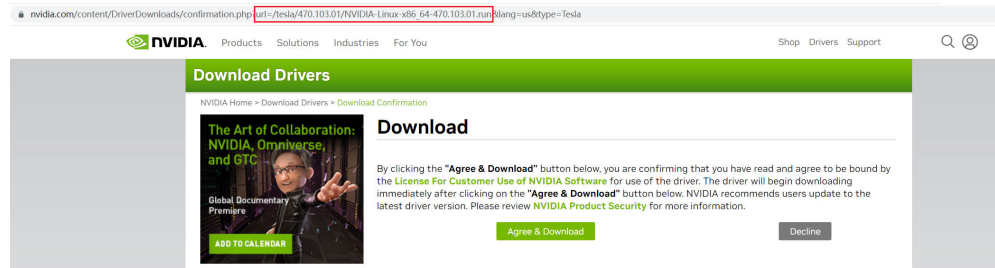
**Download**

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 17-5**, 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 17-5**, 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 17-5 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 17-30 GPU component

Container Component	Description	Resource Type
nvidia-driver-installer	Used for installing an NVIDIA driver on GPU nodes.	DaemonSet

# 17.9 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-purpose, high-performance computing capabilities, 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.

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 plug-ins, 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. Choose **Add-ons** in the navigation pane, locate **Volcano Scheduler** on the right, and click **Install**.

**Step 2** On the **Install Add-on** page, configure the specifications.

**Table 17-31** Add-on configuration

Parameter	Description
Add-on Specifications	Select <b>Standalone</b> , <b>HA</b> , or <b>Custom</b> for <b>Add-on Specifications</b> .
Pods	Number of pods that will be created to match the selected add-on specifications. If you select <b>Custom</b> , you can adjust the number of pods as required.
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 that AZ.</li> <li>• <b>Required:</b> Deployment pods of the add-on will be forcibly scheduled to nodes in different AZs. If there are fewer AZs than pods, the extra pods will fail to run.</li> </ul>



Parameter	Description
Containers	<p>CPU and memory quotas of the container allowed for the selected add-on specifications.</p> <p>If you select <b>Custom</b>, the recommended values for <b>volcano-controller</b> and <b>volcano-scheduler</b> are as follows:</p> <ul style="list-style-type: none"> <li>• If the number of nodes is less than 100, retain the default configuration. The requested CPU is 500 m, and the limit is 2000 m. The requested memory is 500 MiB, and the limit is 2000 MiB.</li> <li>• If the number of nodes is greater than 100, increase the requested CPU by 500 m and the requested memory by 1000 MiB each time 100 nodes (10,000 pods) are added. Increase the CPU limit by 1500 m and the memory limit by 1000 MiB.</li> </ul> <p><b>NOTE</b> Recommended formula for calculating the request value:</p> <ul style="list-style-type: none"> <li>- CPU request value: 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 <a href="#">Table 17-32</a>, 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.</li> <li>- Memory request value: 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 memory request value is the sum of these two values. (The obtained value may be different from the recommended value in <a href="#">Table 17-32</a>. You can use either of them.) Memory request = Number of target nodes/1000 x 2.4 GiB + Number of target pods/10000 x 1 GiB For example, for 2000 nodes and 20,000 pods, the memory request value is 6.8 GiB, that is, 2000/1000 x 2.4 GiB + 20000/10000 x 1 GiB.</li> </ul>

**Table 17-32** Recommended values for volcano-controller and volcano-scheduler

Nodes/Pods in a Cluster	CPU Request (m)	CPU Limit (m)	Memory Request (MiB)	Memory Limit (MiB)
50/5,000	500	2000	500	2000
100/10,000	1000	2500	1500	2500
200/20,000	1500	3000	2500	3500

Nodes/Pods in a Cluster	CPU Request (m)	CPU Limit (m)	Memory Request (MiB)	Memory Limit (MiB)
300/30,000	2000	3500	3500	4500
400/40,000	2500	4000	4500	5500
500/50,000	3000	4500	5500	6500
600/60,000	3500	5000	6500	7500
700/70,000	4000	5500	7500	8500

**Step 3** Configure the add-on parameters.

Configure parameters of the default volcano scheduler. For details, see [Table 17-34](#).

```

colocation_enable: ""
default_scheduler_conf:
 actions: 'allocate, backfill'
 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 17-33** Advanced Volcano configuration parameters

Plug-in	Function	Description	Demonstration
default_scheduler_conf	Used to schedule pods. It consists of a series of actions and plug-ins and features high scalability. You can specify and implement actions and plug-ins 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 plug-in list.</li> </ul>	None
actions	Actions to be executed in each scheduling phase. The configured action sequence is the scheduler execution sequence. For details, see <a href="#">Actions</a> . 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.	The following options are supported: <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>	actions: 'allocate, backfill' <b>NOTE</b> When configuring <b>actions</b> , use either <b>preempt</b> or <b>enqueue</b> .

Plug-in	Function	Description	Demonstration
plugins	Implementation details of algorithms in actions based on different scenarios. For details, see <a href="#">Plugins</a> .	For details, see <a href="#">Table 17-34</a> .	None
tolerations	Tolerance of the add-on to node taints.	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.	<pre> tolerations: - effect: NoExecute   key: node.kubernetes.io/not-ready   operator: Exists   tolerationSeconds: 60 - effect: NoExecute   key: node.kubernetes.io/unreachable   operator: Exists   tolerationSeconds: 60                     </pre>

**Table 17-34** Supported plug-ins

Plug-in	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 plug-in.</li> <li>• <b>binpack.cpu:</b> ratio of CPUs to all resources. The parameter value defaults to <b>1</b>.</li> <li>• <b>binpack.memory:</b> ratio of memory resources to all resources. The parameter value defaults to <b>1</b>.</li> <li>• <b>binpack.resources:</b> other custom resource types requested by the pod, for example, <b>nvdiacom/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.nvdiacom/gpu</b>.</li> </ul>	<pre>- plugins: - name: binpack   arguments:     binpack.weight: 10     binpack.cpu: 1     binpack.memory: 1     binpack.resources:       nvdiacom/gpu,       examplecom/foo  binpack.resources.nvdiacom/ gpu: 2  binpack.resources.exampleco 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>

Plug-in	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: 10     lifecycle.MaxScore: 200.0     lifecycle.SaturatedTresh: 1.0     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>

Plug-in	Function	Description	Demonstration
		<p>Maximum score (equivalent to the weight) of the lifecycle plugin.</p> <ul style="list-style-type: none"> <li> <b>lifecycle.SaturatedTresh</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>.                     </li> </ul> <p>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.</p>	

Plug-in	Function	Description	Demonstration
gang	<p>Consider a group of pods as a whole for resource allocation. This plug-in 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>	<pre>- plugins: - name: priority - name: gang   enablePreemptable: false   enableJobStarving: false - name: conformance</pre>
priority	Schedule based on custom load priorities.	None	<pre>- plugins: - name: priority - name: gang   enablePreemptable: false - name: conformance</pre>



Plug-in	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 plug-in 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>

Plug-in	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>- plugins: - name: 'drf' - <b>name: 'predicates'</b> - name: 'nodeorder'</pre>

Plug-in	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>1</b>.</li> <li>● <b>podaffinity.weight:</b> Pods are scheduled based on pod affinity. This parameter defaults to <b>1</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>1</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>selectorspread.weight:</b> Pods are evenly</li> </ul>	<pre> - plugins: - name: nodeorder arguments:   leastrequested.weight: 1   mostrequested.weight: 0   nodeaffinity.weight: 1   podaffinity.weight: 1   balancedresource.weight: 1   1   tainttoleration.weight: 1   imagelocality.weight: 1   volumebinding.weight: 1   podtopologyspread.weight: 2 </pre>

Plug-in	Function	Description	Demonstration
		<p>scheduled to different nodes. This parameter defaults to <b>0</b>.</p> <ul style="list-style-type: none"> <li>• <b>podtopologyspread.weight</b>: Pods are scheduled based on the pod topology. This parameter defaults to <b>2</b>.</li> </ul>	
cce-gpu-topology-predicate	GPU-topology scheduling preselection algorithm	None	<pre>- plugins: - name: 'cce-gpu-topology-predicate' - name: 'cce-gpu-topology-priority' - name: 'cce-gpu'</pre>
cce-gpu-topology-priority	GPU-topology scheduling priority algorithm	None	<pre>- plugins: - name: 'cce-gpu-topology-predicate' - name: 'cce-gpu-topology-priority' - name: 'cce-gpu'</pre>
cce-gpu	GPU resource allocation that supports decimal GPU configurations by working with the gpu add-on.	None	<pre>- plugins: - name: 'cce-gpu-topology-predicate' - name: 'cce-gpu-topology-priority' - name: 'cce-gpu'</pre>
numa-aware	NUMA affinity scheduling.	<p><b>arguments:</b></p> <ul style="list-style-type: none"> <li>• <b>weight</b>: weight of the numa-aware plug-in</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>
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.	<p><b>arguments:</b></p> <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: <b>networkresource</b> arguments: NetworkType: vpc-router</pre>

Plug-in	Function	Description	Demonstration
nodelocalvolume	Filter out nodes that do not meet local volume requirements.	None	- plugins: - name: 'nodelocalvolume' - name: 'nodeemptydirvolume' - name: 'nodeCSIscheduling' - name: 'networkresource'
nodeemptydirvolume	Filter out nodes that do not meet the emptyDir requirements.	None	- plugins: - name: 'nodelocalvolume' - name: 'nodeemptydirvolume' - name: 'nodeCSIscheduling' - name: 'networkresource'
nodeCSIscheduling	Filter out nodes with malfunctioning everest.	None	- plugins: - name: 'nodelocalvolume' - name: 'nodeemptydirvolume' - name: 'nodeCSIscheduling' - name: 'networkresource'

**Step 4** Click **Install**.

----End

## Components

**Table 17-35** Volcano components

Container 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	Daemon Set
resource-exporter	Report the NUMA topology information of nodes.	Daemon Set

## 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 plug-ins. 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 plug-ins based on your requirements.

Volcano allows you to configure the scheduler during installation, upgrade, and editing. The configuration will be synchronized to volcano-scheduler-configmap.

This section describes how to configure volcano-scheduler.

 **NOTE**

Only Volcano of v1.7.1 and later support this function. On the new plugin page, options such as **plugins.eas\_service** and **resource\_exporter\_enable** are replaced by **default\_scheduler\_conf**.

Log in to the CCE console and access the cluster console. Choose **Add-ons** in the navigation pane. On the right of the page, locate **volcano** and click **Install** or **Upgrade**. In the **Parameters** area, configure the volcano-scheduler parameters.

- Using **resource\_exporter**:

```
{
 "ca_cert": "",
 "default_scheduler_conf": {
 "actions": "allocate, backfill",
 "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"
 }
]
 }
}
"server_cert": "",
"server_key": ""
}

```

After this function is enabled, you can use the functions of the numa-aware plugin and resource\_exporter at the same time.

- Using **eas\_service**:

```

{
 "ca_cert": "",
 "default_scheduler_conf": {
 "actions": "allocate, backfill",
 "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": "eas",
 "custom": {

```

```

 "availability_zone_id": "",
 "driver_id": "",
 "endpoint": "",
 "flavor_id": "",
 "network_type": "",
 "network_virtual_subnet_id": "",
 "pool_id": "",
 "project_id": "",
 "secret_name": "eas-service-secret"
 }
}
],
{
 "plugins": [
 {
 "name": "nodelocalvolume"
 },
 {
 "name": "nodeemptydirvolume"
 },
 {
 "name": "nodeCSIScheduling"
 },
 {
 "name": "networkresource"
 }
]
}
],
"server_cert": "",
"server_key": ""
}

```

- Using **ief**:

```

{
 "ca_cert": "",
 "default_scheduler_conf": {
 "actions": "allocate, backfill",
 "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": "ief",
 "enableBestNode": true
 }
]
 },
 {
 "plugins": [
 {
 "name": "nodelocalvolume"
 },
 {
 "name": "nodeemptydirvolume"
 },
 {
 "name": "nodeCSIScheduling"
 },
 {
 "name": "networkresource"
 }
]
 }
]
},
"server_cert": "",
"server_key": ""
}

```

## Retaining the Original volcano-scheduler-configmap Configurations

If you want to use the original configuration after the plug-in is upgraded, perform the following steps:

**Step 1** Check and back up the original volcano-scheduler-configmap configuration.

Example:

```

kubectl edit cm volcano-scheduler-configmap -n kube-system
apiVersion: v1
data:
 default-scheduler.conf: |-
 actions: "enqueue, allocate, backfill"
 tiers:
 - plugins:
 - name: priority
 - name: gang
 - name: conformance
 - plugins:
 - name: drf
 - name: predicates
 - name: nodeorder
 - name: binpack
 arguments:
 binpack.cpu: 100
 binpack.weight: 10
 binpack.resources: nvidia.com/gpu
 binpack.resources.nvidia.com/gpu: 10000
 - plugins:
 - name: cce-gpu-topology-predicate
 - name: cce-gpu-topology-priority
 - name: cce-gpu

```

- plugins:
- name: nodelocalvolume
- name: nodeemptydirvolume
- name: nodeCSIScheduling
- name: networkresource

**Step 2** Enter the customized content in the **Parameters** area on the console.

```
{
 "ca_cert": "",
 "default_scheduler_conf": {
 "actions": "enqueue, allocate, backfill",
 "tiers": [
 {
 "plugins": [
 {
 "name": "priority"
 },
 {
 "name": "gang"
 },
 {
 "name": "conformance"
 }
]
 }
],
 },
 {
 "plugins": [
 {
 "name": "drf"
 },
 {
 "name": "predicates"
 },
 {
 "name": "nodeorder"
 },
 {
 "name": "binpack",
 "arguments": {
 "binpack.cpu": 100,
 "binpack.weight": 10,
 "binpack.resources": "nvidia.com/gpu",
 "binpack.resources.nvidia.com/gpu": 10000
 }
 }
]
 },
 {
 "plugins": [
 {
 "name": "cce-gpu-topology-predicate"
 },
 {
 "name": "cce-gpu-topology-priority"
 },
 {
 "name": "cce-gpu"
 }
]
 },
 {
 "plugins": [
 {
 "name": "nodelocalvolume"
 },
 {
 "name": "nodeemptydirvolume"
 }
]
 }
}
```

```

 "name": "nodeCSIscheduling"
 },
 {
 "name": "networkresource"
 }
]
 }
}
],
"server_cert": "",
"server_key": ""
}

```

**NOTE**

When this function is used, the original content in volcano-scheduler-configmap will be overwritten. Therefore, you must check whether volcano-scheduler-configmap has been modified during the upgrade. If yes, synchronize the modification to the upgrade page.

----End

## Uninstalling the Volcano Add-on

After the add-on is uninstalled, all custom Volcano resources ([Table 17-36](#)) 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 17-36** 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

## 17.10 CCE Container Storage (FlexVolume)

### Introduction

storage-driver functions as a standard Kubernetes FlexVolume plug-in to allow containers to use EVS, SFS, OBS, and SFS Turbo storage resources. By installing and upgrading storage-driver, you can quickly install and update cloud storage capabilities.

**storage-driver is a system resource add-on. It is installed by default when a cluster of Kubernetes v1.13 or earlier is created.**

## Constraints

- For clusters created in CCE, Kubernetes v1.15.11 is a transitional version in which the FlexVolume add-on (storage-driver) is compatible with the CSI add-on (**everest**). Clusters of v1.17 and later versions do not support FlexVolume anymore. Use the everest add-on.
- The FlexVolume add-on will be maintained by Kubernetes developers, but new functionality will only be added to **CCE Container Storage (Everest)**. Do not create CCE storage that connects to the FlexVolume add-on (storage-driver) anymore. Otherwise, storage may malfunction.
- This add-on can be installed only in **clusters of v1.13 or earlier**. By default, the **everest** add-on is installed when clusters of v1.15 or later are created.

### NOTE

**In a cluster of v1.13 or earlier**, when an upgrade or bug fix is available for storage functionalities, you only need to install or upgrade the storage-driver add-on. Upgrading the cluster or creating a cluster is not required.

## 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:

If storage-driver is not installed in a cluster, perform the following steps to install it:

- 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, locate **CCE Container Storage (Flexvolume)** on the right, and click **Install**.
- Step 2** Click **Install** to install the add-on. Note that the storage-driver has no configurable parameters and can be directly installed.

----End

# 18 Helm Chart

---

## 18.1 Overview

CCE provides a console for managing Helm charts, helping you easily deploy applications using the charts and manage applications on the console.

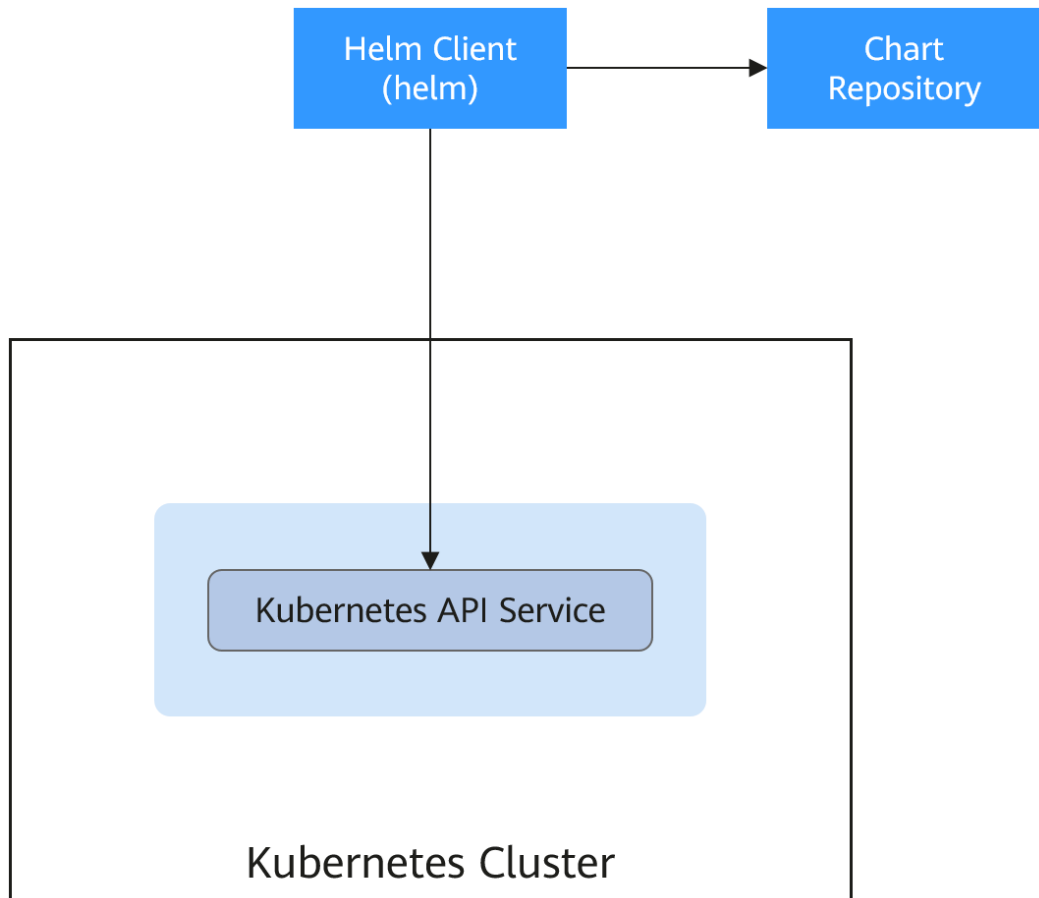
### Helm

**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 releasers 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.

## 18.2 Deploying an Application from a Chart

On the CCE console, you can upload a Helm chart package, deploy it, and manage the deployed pods.

### 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 18-1](#), the parameters marked with \* are mandatory.

**Table 18-1** Parameters in the directory structure of a chart

Parameter	Description
* templates	Stores all templates.
* values.yaml	Describes configuration parameters required by templates. <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.  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.

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 **Charts** 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 **Charts**.

**Step 2** On the **My Charts** tab page, click **Install** of the target chart.

**Step 3** Set workload installation parameters by referring to [Table 18-2](#).

**Table 18-2** 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> 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>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 **Charts** in the navigation pane and 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. 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 and click the cluster name to access the cluster console. Choose **Charts** in the navigation pane and 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 **Charts** in the navigation pane and 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

## 18.3 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 old manifest, its live state, and the new manifest when generating a patch. Helm compares the current live state with the live state of the old 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)

## 18.4 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 -xvzf 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 aerospace/
```
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 , if 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
```

- 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"}
Error: cannot connect to Tiller
```

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](#).

## 18.5 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](#).

### 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 -xvzf 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.

- Default installation: This is the simplest method, which requires only two parameters.

```
helm install {release_name} {chart_name}
```

For example, to install WordPress, the WordPress chart added in [step 1](#) is **bitnami/wordpress**, and the release name is **my-wordpress**.

```
helm install my-wordpress bitnami/wordpress
```

- Custom installation: The default installation uses the default settings in the chart. Use custom installation to custom parameter settings. 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
```

Overwrite specified parameters by running the following commands:

```
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
```

**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 , if 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
```

- 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](#).

## 18.6 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 2to3 conversion plug-in 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 plug-in 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 plug-in.

```
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 plug-in 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
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

# 19 Permissions

---

## 19.1 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 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 non-Kubernetes APIs in CCE clusters and support fine-grained IAM policies.

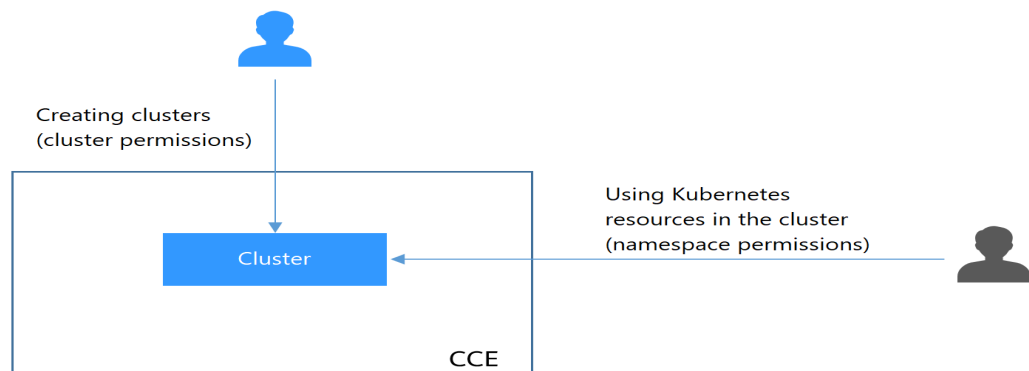
- **Namespace-level permissions:** You can regulate users' or user groups' access to Kubernetes resources in a single namespace based on their Kubernetes RBAC roles. CCE has also been enhanced based on open-source capabilities. It supports RBAC authorization based on IAM user or user group, and RBAC 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 19-1** 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 19-1](#) lists the namespace permissions of different users.

**Table 19-1** 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 19-1](#).

## 19.2 Cluster Permissions (IAM-based)

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 for users to operate cluster-related resources only (such as clusters and nodes). To operate Kubernetes resources like workloads and Services, you must be granted the [namespace permissions](#) at the same time.
  - 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

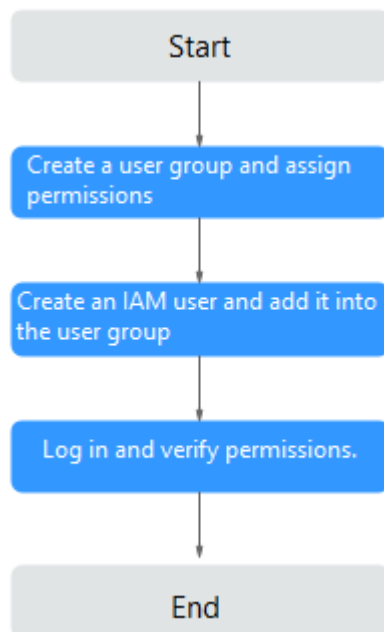
- A user with the Security Administrator role (for example, your account) has all IAM permissions except role switching. Only these users can view user groups and their permissions on the **Permissions** page on the CCE console.

### 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 19-2** 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 **CCEReadOnlyAccess** 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, switch to the CCE console, and buy a cluster. If you fail to do so (assuming that only the **CCEReadOnlyAccess** permission is assigned), the **CCEReadOnlyAccess** policy takes 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 **CCEReadOnlyAccess** policy takes 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 **CCEAdministrator**. 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 **CCEFullAccess** and **CCEReadOnlyAccess**.

- **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.
- 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



Resource Type	Resource Name	Description
	job	Job
	tag	Cluster label
	addonInstance	Add-on instance
	release	Helm version
	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. 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

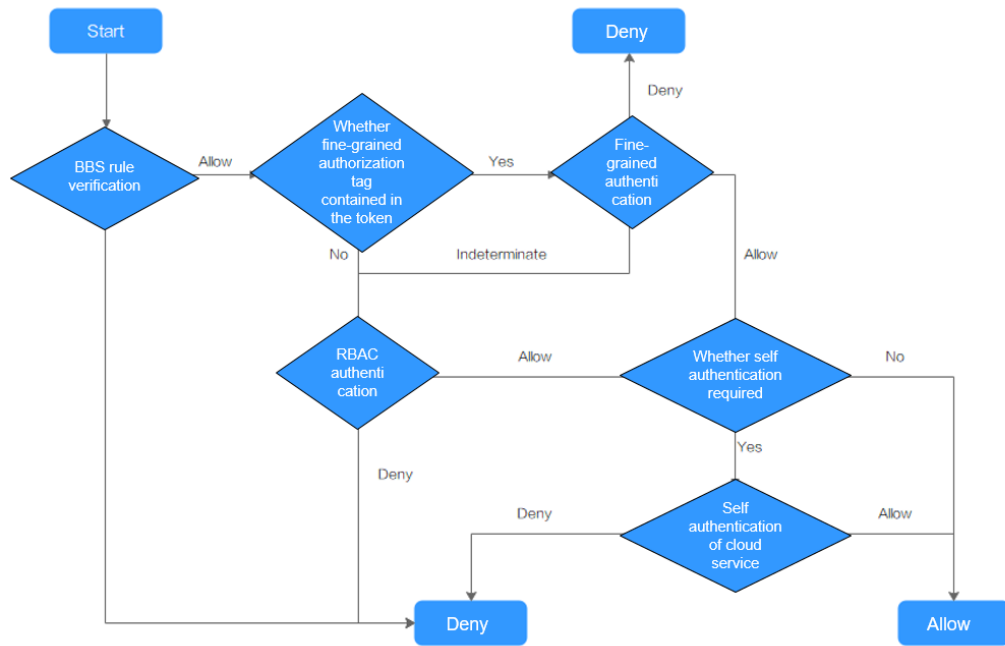
### NOTE

- 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

## 19.3 Namespace Permissions (Kubernetes RBAC-based)

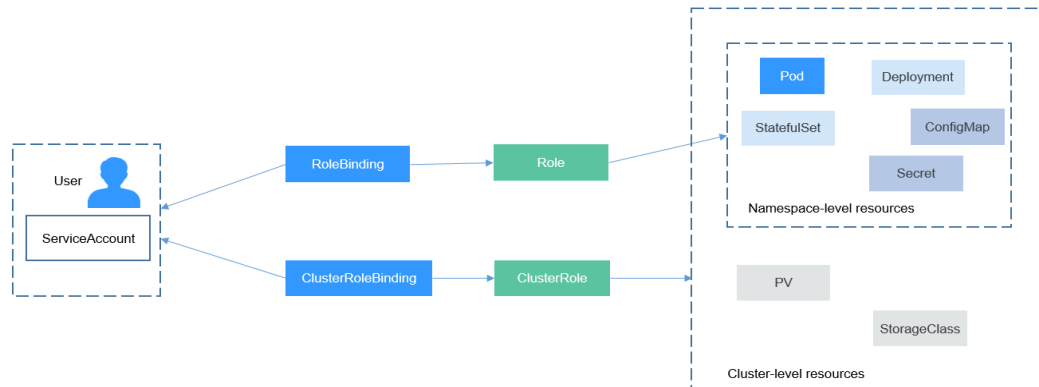
### Namespace Permissions (Kubernetes RBAC-based)

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 19-3** 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.

## 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 19-2](#) lists the namespace permissions of different users.

**Table 19-2** 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.

## 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.

## 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.

- 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**.

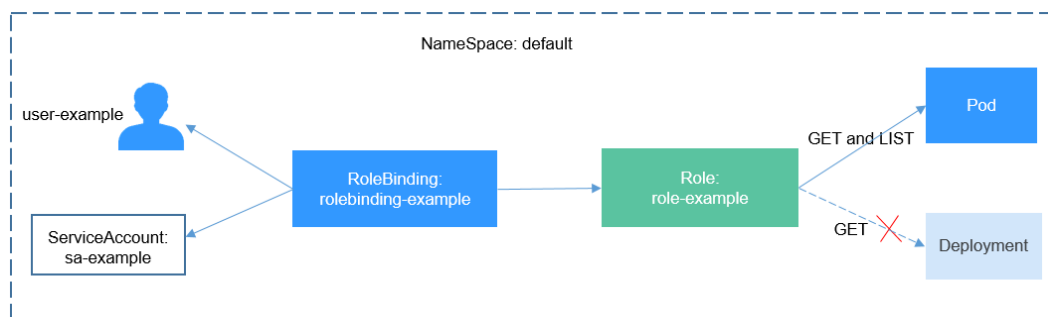
For details, see [Using RBAC Authorization](#).

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 19-4** 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-twvnp 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, 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 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 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
```

### 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 binds the admin role 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](#).

## 19.4 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.

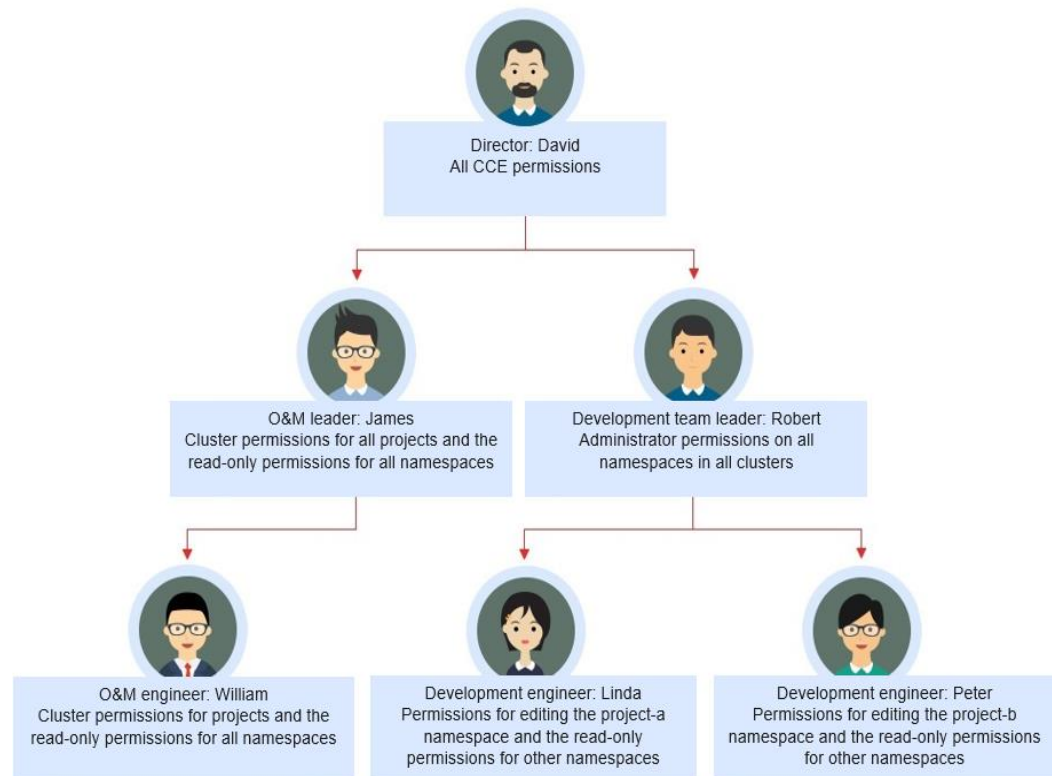
Cluster permissions and namespace permissions are independent of each other but must be used together. The permissions set for a user group apply to all users in the user group. When multiple permissions are added to a user or user group, they take effect at the same time (the union set is used).

### 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.

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.

## 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.

By now, all the required permissions are assigned to the department members.

## 19.5 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 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.
- 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.
  - 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 19-3](#) 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 19-3** Dependency policies

Console Function	Dependent Services	Roles or Policies Required
Cluster overview	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, enterprise projects, 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 SFS FullAccess 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>
Networking	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
Container storage	Object Storage Service (OBS) Scalable File Service (SFS) SFS Turbo	<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, enterprise projects, 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 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	None	None
Chart management	None	Cloud accounts and the IAM users with CCE Administrator assigned can use this function.
Add-on management	None	Cloud accounts and the IAM users with CCE Administrator, CCE FullAccess, or CCE ReadOnlyAccess assigned can use this function.
Permissions management	None	<ul style="list-style-type: none"> <li>For cloud accounts, no additional policy/role is required.</li> <li>IAM users with CCE Administrator or global Security Administrator assigned can use this function.</li> <li>IAM users with the CCE FullAccess or CCE ReadOnlyAccess permission can access the namespace. In addition, the IAM users must have the <b>administrator permissions (cluster-admin)</b> on the namespace.</li> </ul>

Console Function	Dependent Services	Roles or Policies Required
ConfigMaps and Secrets	None	<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	None	None
Switching to other related services	Software Repository for Container (SWR) Application Operations Management (AOM)	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.

## 19.6 Pod Security

### 19.6.1 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 19-4](#).

**Table 19-4** 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.
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 19-4](#).

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
 resourceNameNames:
 - 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
```

## 19.6.2 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 19-5** 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 19-6](#)). 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 19-6** 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 19-6](#).
  - <LEVEL>: must be **privileged**, **baseline**, or **restricted**. For details about the levels, see [Table 19-5](#).
- 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 19-6](#).
- `<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](#)

## 19.7 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 connect to 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
```

# 20 FAQs

---

## 20.1 Common Questions

### Cluster Management

- [Why Can't I Create a CCE Cluster?](#)
- [Is Management Scale of a Cluster Related to the Number of Master Nodes?](#)
- [How Do I Rectify the Fault When the Cluster Status Is Unavailable?](#)

### Node/Node Pool Management

- [What Should I Do If a Cluster Is Available But Some Nodes Are Unavailable?](#)
- [What Should I Do If I/O Suspension Occasionally Occurs When SCSI EVS Disks Are Used?](#)

### Workload Management

- [What Should I Do If Pod Scheduling Fails?](#)
- [What Should I Do If a Pod Fails to Pull the Image?](#)
- [What Should I Do If Container Startup Fails?](#)
- [What Should I Do If Pods in the Terminating State Cannot Be Deleted?](#)
- [What Is the Image Pull Policy for Containers in a CCE Cluster?](#)

### Networking

[Why Does the Browser Return Error Code 404 When I Access a Deployed Application?](#)

[What Should I Do If a Node Fails to Connect to the Internet \(Public Network\)?](#)

[How Do I Optimize the Configuration If the External Domain Name Resolution Is Slow or Times Out?](#)



## 20.2 Billing

### 20.2.1 How Is CCE Billed/Charged?

#### Billing Items

CCE is free of charge. You only pay for the resources (such as nodes) created when you are using CCE. There are two types of billing items:

1. **Clusters:** the cost of resources used by master nodes, varying depending on the cluster type and size. Cluster types include VM cluster and BMS cluster (the number of master nodes determines whether a cluster is highly available). Cluster size (also called management scale) indicates the maximum number of nodes allowed in a cluster.

 **NOTE**

The management scale indicates the number of ECSs or BMSs in a cluster.

2. **IaaS resource costs** are incurred from the IaaS resources you use to run worker nodes in your cluster. These resources, created either manually or automatically, include ECSs, EVS disks, EIPs, bandwidth, and load balancers.

#### Billing Modes

CCE is billed on a pay-per-use basis.

- **Pay-per-use:** It is a pay-after-use mode. Billing starts when a resource is provisioned and stops when the resource is deleted. You can use cloud resources as required and stop paying for them when you no longer need them. There is no upfront payment for excess capacity.

 **NOTE**

The following are pricing principles in the case of CCE cluster hibernation or node shutdown. Note that there are many types of cluster nodes and ECS is used as an example.

- **Cluster hibernation:** For a hibernated cluster, you will not be billed for master nodes, but the resources used by or bound to the cluster, such as the EVS disks, EIPs, and bandwidth, will be billed in their own ways such as pay-per-use.
- **Node shutdown:** Hibernating a cluster will not stop worker nodes in the cluster. You can select **Stop all nodes in the cluster** to do so. You can also stop a node on the ECS console after hibernating a cluster.

---

**NOTICE**

- Clusters follow a tiered pricing plan. Pricing for each tier varies with cluster size and type.
  - Once a pay-per-use resource becomes in arrears, provides a grace period and a retention period during which you can top up your account or renew resources. Within the grace period, you can still access and use your cloud service.
-

## 20.2.2 Will I Be Notified When My Balance Is Insufficient?

In **Billing Center**, on the page, click **Settings** in the **Available Credit** area. Then, you can enable or disable **Balance Alert**. Click **Modify** and set a desired threshold.

- After the function is enabled, if your total balance (including cash balance, credit balance, common vouchers, and flexi-purchase coupons) is lower than the alert threshold, the system sends you SMS and email notifications every day for a maximum of three consecutive days.
- You can change contacts that receive the balance alerts at **SMS & Email Settings > Finance > Account balance** in **Message Center**.

## 20.2.3 Will I Be Notified When My Account Balance Changes?

The system will notify you via email or SMS message of your account balance changes, including whether your online topping up is successful.

## 20.3 Cluster

### 20.3.1 Cluster Creation

#### 20.3.1.1 Why Can't I Create a CCE Cluster?

##### Overview

This section describes how to locate and rectify the fault if you fail to create a CCE cluster.

##### Details

###### Possible causes:

1. The Network Time Protocol daemon (ntpd) is not installed or fails to be installed, Kubernetes components fail to pass the pre-verification, or the disk partition is incorrect. The current solution is to create a cluster again. For details about how to locate the fault, see [Locating the Failure Cause](#).

### Locating the Failure Cause

View the cluster logs to identify the cause and rectify the fault.

**Step 1** Log in to the CCE console. In the navigation pane, click **Operation Records** above the cluster list to view operation records.

**Step 2** Click the record of the **Failed** status to view error information.

**Step 3** Rectify the fault based on the error information and create a cluster again.

----End

### 20.3.1.2 Is Management Scale of a Cluster Related to the Number of Master Nodes?

Management scale indicates the maximum number of nodes that can be managed by a cluster. If you select **50 nodes**, the cluster can manage a maximum of 50 nodes.

The number of master nodes varies according to the cluster specification, but is not affected by the management scale.

After the multi-master node mode is enabled, three master nodes will be created. If a master node is faulty, the cluster can still be available without affecting service functions.

### 20.3.1.3 Which Resource Quotas Should I Pay Attention To When Using CCE?

CCE restricts **only the number of clusters**. However, when using CCE, you may also be using other cloud services, such as Elastic Cloud Server (ECS), Elastic Volume Service (EVS), Virtual Private Cloud (VPC), Elastic Load Balance (ELB), and SoftWare Repository for Containers (SWR).

### What Is a Quota?

A quota is a limit on the quantity or capacity of a certain type of cloud service resources that you can use, for example, the maximum number of ECSs or EVS disks that you can create.

You can request for an increase in quota if an existing quota cannot meet your service requirements.

## 20.3.2 Cluster Running

### 20.3.2.1 How Do I Rectify the Fault When the Cluster Status Is Unavailable?

If the cluster is **Unavailable**, perform the following operations to rectify the fault:

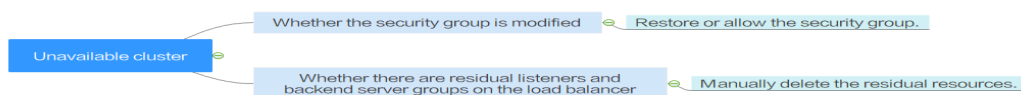
### Fault Locating

Troubleshooting methods are sorted based on the occurrence probability of the possible causes. You are advised to check the possible causes from high probability to low probability to quickly locate the cause of the problem.

If the fault persists after a possible cause is rectified, check other possible causes.

- **Check Item 1: Whether the Security Group Is Modified**
- **Check Item 2: Whether There Are Residual Listeners and Backend Server Groups on the Load Balancer**

Figure 20-1 Fault locating



## Check Item 1: Whether the Security Group Is Modified

- Step 1** Log in to the management console, and choose **Service List > Networking > Virtual Private Cloud**. In the navigation pane on the left, choose **Access Control > Security Groups** to find the security group of the master node in the cluster.

The name of this security group is in the format of *Cluster name-cce-control-ID*.

- Step 2** Click the security group. On the details page displayed, ensure that the security group rules of the master node are correct.

For details about security, see [Configuring Cluster Security Group Rules](#).

----End

## Check Item 2: Whether There Are Residual Listeners and Backend Server Groups on the Load Balancer

### Reproducing the Problem

A cluster exception occurs when a LoadBalancer Service is being created or deleted. After the fault is rectified, the Service is deleted successfully, but there are residual listeners and backend server group.

- Step 1** Pre-create a CCE cluster. In the cluster, use the official Nginx image to create workloads, preset load balancers, Services, and ingresses.
- Step 2** Ensure that the cluster is running properly and the Nginx workload is stable.
- Step 3** Create and delete 10 LoadBalancer Services every 20 seconds.
- Step 4** An injection exception occurs in the cluster. For example, the etcd pod is unavailable or the cluster is hibernated.

----End

### Possible Cause

There are residual listeners and backend server groups on the load balancer.

### Solution

Manually clear residual listeners and backend server groups.

- Step 1** Log in to the management console and choose **Network > Elastic Load Balance** from the service list.
- Step 2** In the load balancer list, click the name of the target load balancer to go to the details page. On the **Listeners** tab page, locate the target listener and delete it.
- Step 3** On the **Backend Server Groups** tab page, locate the target backend server group and delete it.

----End

## 20.3.2.2 How Do I Retrieve Data After a Cluster Is Deleted?

After a cluster is deleted, the workload on the cluster will also be deleted and cannot be restored. Therefore, exercise caution when deleting a cluster.

## 20.3.3 Cluster Deletion

### 20.3.3.1 Failed to Delete a Cluster: Residual ENIs

When deleting a cluster, CCE obtains the cluster's resources through kube-apiserver of the cluster. If the cluster is unavailable, frozen, or hibernated, the resources may fail to be obtained, and the cluster may not be deleted.

#### Symptom

Failed to delete a cluster.

```
Failed Operation:
Resource ID:
Reason: delete failed: {"code":
4967ba194623","action":"SecGrp:DeleteMasterSecGrp:Skip","message":"Expected HTTP response code [200 202 204 404] when accessing [DELETE
https/
instead\n{"NeutronError":{"message":"Security Group
use.,"type":"SecurityGroupInUse","detail":{"}}"}
623 in
```

#### Possible Causes

In this example, the ENI cannot be deleted because kube-apiserver of the cluster fails to obtain the ENI or sub ENI of the cluster. The security group created by CCE for the ENI or sub ENI reports the error code **409**. As a result, the cluster fails to be deleted.

#### Procedure

- Step 1** Copy the resource ID in the error information, go to the **Security Groups** page of the VPC console, and obtain security groups by ID.
- Step 2** Click the security group to view its details, and click the **Associated Instances** tab.  
  
A security group remains after the deletion, because it is attached to an ENI or a sub ENI. Click the **Others** tab to view the residual ENIs. Delete the residual ENIs, and the sub ENI will be automatically deleted.
- Step 3** Choose **Network Interfaces** in the navigation pane to delete the ENIs obtained in the previous step.  
  
You can search for the ENIs to be deleted by ID or name.
- Step 4** Go to the **Security Groups** page, and confirm that *Cluster name-cce-eni-xxx* is not attached to any ENIs. Then, you can delete the cluster on the CCE console.  
  
----End

### 20.3.3.2 How Do I Clear Residual Resources After a Non-Running Cluster Is Deleted?

If a cluster is not in the running state (for example, frozen or unavailable), resources such as PVCs, Services, and ingresses in the cluster cannot be obtained. After the cluster is deleted, residual network and storage resources may exist. In this case, manually delete these resources on their respective service console.

## Deleting Residual ELB Resources

- Step 1** Log in to the ELB console.
  - Step 2** Search for load balancers in the VPC by VPC ID.
  - Step 3** View the listener details of the load balancer. If the description contains the cluster ID and Service ID, the listener is created in the cluster.
  - Step 4** Delete residual load balancer-related resources from the cluster based on the preceding information.
- End

## Deleting Residual EVS Resources

An EVS disk dynamically created using a PVC is named in the format of "pvc-{uid}". The **metadata** field in the API contains the cluster ID. You can use this cluster ID to obtain these EVS disks in the cluster and delete them as required.

- Step 1** Go to the EVS console.
- Step 2** Obtain EVS disks by name ("pvc-{uid}") to obtain all automatically created EVS disks in the CCE clusters.
- Step 3** Press **F12** to open the developer tools. Check whether the **metadata** field in the **detail** interface contains the cluster ID. If yes, the EVS disk is automatically created in this cluster.
- Step 4** Delete the residual EVS resources from the cluster based on the preceding information.

 **NOTE**

Deleted data cannot be restored. Exercise caution when performing this operation.

----End

## Deleting Residual SFS Resources

The name format of SFS systems dynamically created using a PVC is "pvc-{uid}". The **metadata** field in the API contains the cluster ID. You can use this cluster ID to obtain these SFS systems automatically created in the cluster, and delete them as required.

- Step 1** Log in to the SFS console.
- Step 2** Search for SFS systems by name ("pvc-{uid}") to obtain all automatically created SFS systems in CCE clusters.
- Step 3** Press **F12** to open the developer tools. Check whether the **metadata** field in the **detail** interface contains the cluster ID. If yes, the SFS system is automatically created in the cluster.
- Step 4** Delete the residual SFS resources in the cluster based on the preceding information.

 NOTE

Deleted data cannot be restored. Exercise caution when performing this operation.

----End

## 20.3.4 Cluster Upgrade

### 20.3.4.1 What Do I Do If a Cluster Add-On Fails to be Upgraded During the CCE Cluster Upgrade?

#### Overview

This section describes how to locate and rectify the fault if you fail to upgrade an add-on during the CCE cluster upgrade.

#### Procedure

- Step 1** If the add-on fails to be upgraded, try again first. If the retry fails, perform the following steps to rectify the fault.
- Step 2** If a failure message is displayed on the upgrade page, go to the **Add-ons** page to view the add-on status. For an abnormal add-on, click the add-on name to view details.
- Step 3** On the pod details page, click **View Events** in the **Operation** column of the abnormal pod.
- Step 4** Rectify the fault based on the exception information. For example, delete the pod that is not started or restart it.
- Step 5** After the processing is successful, the add-on status changes to **Running**. Ensure that all add-ons are in the **Running** status.
- Step 6** Go to the cluster upgrade page and click **Retry**.

----End

## 20.4 Node

### 20.4.1 Node Creation

## 20.4.1.1 How Do I Troubleshoot Problems Occurred When Adding Nodes to a CCE Cluster?

### Note

- The node images in the same cluster must be the same. Pay attention to this when creating, adding, or accepting nodes in a cluster.
- If you need to allocate user space from the data disk when creating a node, do not set the data storage path to any key directory. For example, to store data in the **/home** directory, set the directory to **/home/test** instead of **/home**.

#### NOTE

Do not set **Path inside a node** to the root directory **/**. Otherwise, the mounting fails. Set **Path inside a node** to any of the following:

- **/opt/xxxx** (excluding **/opt/cloud**)
- **/mnt/xxxx** (excluding **/mnt/paas**)
- **/tmp/xxx**
- **/var/xxx** (excluding key directories such as **/var/lib**, **/var/script**, and **/var/paas**)
- **/xxxx** (It cannot conflict with the system directory, such as **bin**, **lib**, **home**, **root**, **boot**, **dev**, **etc**, **lost+found**, **mnt**, **proc**, **sbin**, **srv**, **tmp**, **var**, **media**, **opt**, **selinux**, **sys**, and **usr**.)

Do not set it to **/home/paas**, **/var/paas**, **/var/lib**, **/var/script**, **/mnt/paas**, or **/opt/cloud**. Otherwise, the system or node installation will fail.

## Check Item 1: Subnet Quota

### Symptom

New nodes cannot be added to a CCE cluster, and a message is displayed indicating that the subnet quota is insufficient.

### Cause Analysis

Example:

VPC CIDR block: 192.168.66.0/24

Subnet CIDR block: 192.168.66.0/24

In 192.168.66.0/24, all 251 private IP addresses have been used.

### Solution

**Step 1** Expand the VPC.

Log in to the console and choose **Virtual Private Cloud** from the service list. Click **Edit CIDR Block** in the **Operation** column of the target VPC.

**Step 2** Change the subnet mask to **16** and click **OK**.

**Step 3** Click the VPC name. On the **Summary** tab page, click the number next to **Subnets** on the right and click **Create Subnet** to create a subnet.



**Step 4** Return to the page for adding a node on the CCE console, and select the newly created subnet.

 **NOTE**

1. Adding subnets to the VPC does not affect the use of the existing 192.168.66.0/24 CIDR block.  
You can select a new subnet when creating a CCE node. The new subnet has a maximum of 251 private IP addresses. If the number of private IP addresses cannot meet service requirements, you can add more subnets.
2. Subnets in the same VPC can communicate with each other.

----End

## Check Item 2: EIP Quota

### Symptom

When a node is added, **EIP** is set to **Automatically assign**. The node cannot be created, and a message indicating that EIPs are insufficient is displayed.

### Solution

Two methods are available to solve the problem.

- **Method 1:** Unbind the VMs bound with EIPs and add a node again.
  - a. Log in to the management console.
  - b. Choose **Computing** > **Elastic Cloud Server**.
  - c. In the ECS list, locate the target ECS and click its name.
  - d. On the ECS details page, click the **EIPs** tab. In the EIP list, click **Unbind** at the row of the target ECS and click **Yes**.
  - e. Return to the page for adding a node on the CCE console, select **Use existing** for **EIP**, and add the node again.
- **Method 2:** Increase the EIP quota.

## Check Item 3: Security Group

### Symptom

A node cannot be added to a CCE cluster.

### Solution

You can click the cluster name to view the cluster details. In the **Networking Configuration** area, click the icon next to **Default security group of the node** to check whether the default security group is deleted and whether the security group rules comply with [Configuring Cluster Security Group Rules](#).

If your account has multiple clusters and you need to manage network security policies of nodes in a unified manner, you can specify custom security groups.

## 20.4.2 Node Running

### 20.4.2.1 What Should I Do If a Cluster Is Available But Some Nodes Are Unavailable?

If the cluster status is available but some nodes in the cluster are unavailable, perform the following operations to rectify the fault:

#### Mechanism for Detecting Node Unavailability

Kubernetes provides the heartbeat mechanism to help you determine node availability. For details about the mechanism and interval, see [Heartbeats](#).

#### Troubleshooting Process

Troubleshooting methods are sorted based on the occurrence probability of the possible causes. You are advised to check the possible causes from high probability to low probability to quickly locate the cause of the problem.

If the fault persists after a possible cause is rectified, check other possible causes.

- [Check Item 1: Whether the Node Is Overloaded](#)
- [Check Item 2: Whether the ECS Is Deleted or Faulty](#)
- [Check Item 3: Whether You Can Log In to the ECS](#)
- [Check Item 4: Whether the Security Group Is Modified](#)
- [Check Item 5: Whether the Security Group Rules Contain the Security Group Policy for the Communication Between the Master Node and the Worker Node](#)
- [Check Item 6: Whether the Disk Is Abnormal](#)
- [Check Item 7: Whether Internal Components Are Normal](#)
- [Check Item 8: Whether the DNS Address Is Correct](#)
- [Check Item 9: Whether the vdb Disk on the Node Is Deleted](#)
- [Check Item 10: Whether the Docker Service Is Normal](#)

**Figure 20-2** Troubleshooting process for an unavailable node



## Check Item 1: Whether the Node Is Overloaded

### Symptom

The node connection in the cluster is abnormal. Multiple nodes report write errors, but services are not affected.

### Fault Locating

**Step 1** Log in to the CCE console, and click the cluster name to access the cluster console. In the navigation pane, choose **Nodes**. Click **Monitor** in the row of the unavailable node.

**Step 2** On the top of the displayed page, click **View More** to go to the AOM console and view historical monitoring records.

A too high CPU or memory usage of the node will result in a high network latency or trigger system OOM. Therefore, the node is displayed as unavailable.

----End

### Solution

1. You are advised to migrate services to reduce the workloads on the node and set the resource upper limit for the workloads.
2. Clear data on the CCE nodes in the cluster.
3. Limit the CPU and memory quotas of each container.
4. Add more nodes to the cluster.
5. You can also restart the node on the ECS console.
6. Add nodes to deploy memory-intensive containers separately.
7. Reset the node.

After the node becomes available, the workload is restored.

## Check Item 2: Whether the ECS Is Deleted or Faulty

**Step 1** Check whether the cluster is available.

Log in to the CCE console and check whether the cluster is available.

- If the cluster is unavailable, for example, an error occurs, perform operations described in [How Do I Rectify the Fault When the Cluster Status Is Unavailable?](#)
- If the cluster is running but some nodes in the cluster are unavailable, go to [Step 2](#).

**Step 2** Log in to the ECS console and view the ECS status.

- If the ECS status is **Deleted**, go back to the CCE console, delete the corresponding node from the node list of the cluster, and then create another one.
- If the ECS status is **Stopped** or **Frozen**, restore the ECS first. It takes about 3 minutes to restore the ECS.
- If the ECS is **Faulty**, restart the ECS to rectify the fault.
- If the ECS status is **Running**, log in to the ECS to locate the fault according to [Check Item 7: Whether Internal Components Are Normal](#).

----End

## Check Item 3: Whether You Can Log In to the ECS

**Step 1** Log in to the ECS console.

**Step 2** Check whether the node name displayed on the page is the same as that on the VM and whether the password or key can be used to log in to the node.

If the node names are inconsistent and the password and key cannot be used to log in to the node, Cloud-Init problems occurred when an ECS was created. In this case, restart the node and submit a service ticket to the ECS personnel to locate the root cause.

----End

## Check Item 4: Whether the Security Group Is Modified

Log in to the VPC console. In the navigation pane, choose **Access Control** > **Security Groups** and locate the security group of the cluster master node.

The name of this security group is in the format of *Cluster name-cce-control-ID*. You can search for the security group by cluster name.

Check whether the rules in the security group are modified. For details, see [Configuring Cluster Security Group Rules](#).

## Check Item 5: Whether the Security Group Rules Contain the Security Group Policy for the Communication Between the Master Node and the Worker Node

Check whether such a security group policy exists.

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.)

For details about security, see [Configuring Cluster Security Group Rules](#).

## Check Item 6: Whether the Disk Is Abnormal

A 100 GiB data disk dedicated for Docker is attached to the new node. If the data disk is uninstalled or damaged, the Docker service becomes abnormal and the node becomes unavailable.

Click the node name to check whether the data disk mounted to the node is uninstalled. If the disk is uninstalled, mount a data disk to the node again and restart the node. Then the node can be recovered.

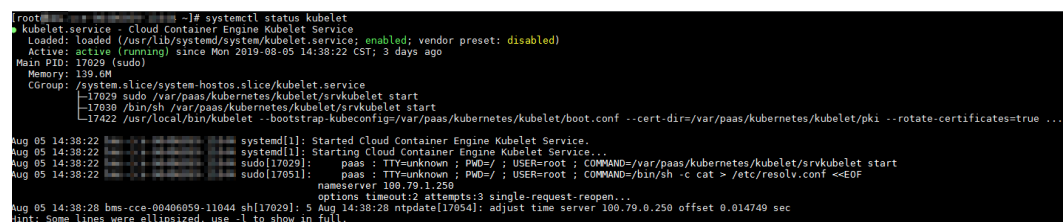
## Check Item 7: Whether Internal Components Are Normal

**Step 1** Log in to the ECS where the unavailable node is located.

**Step 2** Run the following command to check whether the PaaS components are normal:

```
systemctl status kubelet
```

If the command is successfully executed, the status of each component is displayed as **active**, as shown in the following figure.



```
root@bms-ccc-00406059-11044 ~# systemctl status kubelet
● kubelet.service - Cloud Container Engine Kubelet Service
 Loaded: loaded (/usr/lib/systemd/system/kubelet.service; enabled; vendor preset: disabled)
 Active: active (running) since Mon 2019-08-05 14:38:22 CST; 3 days ago
 Main PID: 17029 (sudo)
 Memory: 139.6M
 CGroup: /system.slice/system-hostos.slice/kubelet.service
 └─17029 sudo /var/paas/kubernetes/kubelet/srvkubelet start
 └─17030 /bin/sh /var/paas/kubernetes/kubelet/srvkubelet start
 └─17422 /usr/local/bin/kubelet --bootstrap-kubeconfig=/var/paas/kubernetes/kubelet/boot.conf --cert-dir=/var/paas/kubernetes/kubelet/pki --rotate-certificates=true ...

Aug 05 14:38:22 bms-ccc-00406059-11044 systemd[1]: Started Cloud Container Engine Kubelet Service.
Aug 05 14:38:22 bms-ccc-00406059-11044 systemd[1]: Starting Cloud Container Engine Kubelet Service.
Aug 05 14:38:22 bms-ccc-00406059-11044 sudo[17029]: pass : TTY=unknown ; PWD=/ ; USER=root ; COMMAND=/var/paas/kubernetes/kubelet/srvkubelet start
Aug 05 14:38:22 bms-ccc-00406059-11044 sudo[17051]: pass : TTY=unknown ; PWD=/ ; USER=root ; COMMAND=/bin/sh -c cat > /etc/resolv.conf <<EOF
nameserver 100.79.1.250
options timeout:2 attempts:3 single-request-reopen...
Aug 05 14:38:28 bms-ccc-00406059-11044 sh[17029]: 5 Aug 14:38:28 ntpdate[17054]: adjust time server 100.79.0.250 offset 0.014749 sec
hint: Some lines were ellipsized, use -l to show in full.
```

If the component status is not **active**, run the following commands (using the faulty component **canal** as an example):

Run **systemctl restart canal** to restart the component.

After restarting the component, run **systemctl status canal** to check the status.

**Step 3** If the restart command fails to be run, run the following command to check the running status of the monitrc process:

```
ps -ef | grep monitrc
```

If the monitrc process exists, run the following command to kill this process. The monitrc process will be automatically restarted after it is killed.

```
kill -s 9 `ps -ef | grep monitrc | grep -v grep | awk '{print $2}`
```

----End

## Check Item 8: Whether the DNS Address Is Correct

**Step 1** After logging in to the node, check whether any domain name resolution failure is recorded in the **/var/log/cloud-init-output.log** file.

```
cat /var/log/cloud-init-output.log | grep resolv
```

If the command output contains the following information, the domain name cannot be resolved:

```
Could not resolve host: Unknown error
```

**Step 2** On the node, ping the domain name that cannot be resolved in the previous step to check whether the domain name can be resolved on the node.

- If not, the DNS cannot resolve the IP address. Check whether the DNS address in the **/etc/resolv.conf** file is the same as that configured on the VPC subnet. In most cases, the DNS address in the file is incorrectly configured. As a result, the domain name cannot be resolved. Correct the DNS configuration of the VPC subnet and reset the node.
- If yes, the DNS address configuration is correct. Check whether there are other faults.

----End

## Check Item 9: Whether the vdb Disk on the Node Is Deleted

If the vdb disk on a node is deleted, you can refer to [this topic](#) to restore the node.

## Check Item 10: Whether the Docker Service Is Normal

**Step 1** Run the following command to check whether the Docker service is running:

```
systemctl status docker
```

```
● docker.service - Docker Application Container Engine
 Loaded: loaded (/usr/lib/systemd/system/docker.service; enabled; vendor preset: disabled)
 Active: active (running) since Wed 2021-02-03 16:07:02 CST; 1 day 23h ago
 Docs: https://docs.docker.com
 Main PID: 3673 (dockerd)
 Tasks: 46 (limit: 24004)
 Memory: 491.2M
 CGroup: /system.slice/docker.service
 └─3673 /usr/bin/dockerd --live-restore --log-opt max-size=50m --log-opt max-file=20 --log-driver=json-fil
 └─3680 containerd --config /var/run/docker/containerd/containerd.toml --log-level info
 └─5961 containerd-shim -namespace moby -workdir /var/lib/docker/containerd/daemon/io.containerd.runtime.v
 └─6811 containerd-shim -namespace moby -workdir /var/lib/docker/containerd/daemon/io.containerd.runtime.v

Warning: Journal has been rotated since unit was started. Log output is incomplete or unavailable.
```

If the command fails or the Docker service status is not active, locate the cause or contact technical support if necessary.

**Step 2** Run the following command to check the number of containers on the node:

```
docker ps -a | wc -l
```

If the command is suspended, the command execution takes a long time, or there are more than 1000 abnormal containers, check whether workloads are repeatedly created and deleted. If a large number of containers are frequently created and deleted, a large number of abnormal containers may occur and cannot be cleared in a timely manner.

In this case, stop repeated creation and deletion of the workload or use more nodes to share the workload. Generally, the nodes will be restored after a period of time. If necessary, run the `docker rm {container_id}` command to manually clear abnormal containers.

----End

### 20.4.2.2 How Do I Log In to a Node Using a Password and Reset the Password?

#### Context

When creating a node on CCE, you selected a key pair or specified a password for login. If you forget your key pair or password, you can log in to the ECS console to reset the password of the node. After the password is reset, you can log in to the node using the password.

#### Procedure

- Step 1** Log in to the ECS console.
- Step 2** In the ECS list, select the cloud server type of the node. In the same row as the node, choose **More > Stop**.
- Step 3** After the node is stopped, choose **More > Reset Password**, and follow on-screen prompts to reset the password.
- Step 4** After the password is reset, choose **More > Start**, and click **Remote Login** to log in to the node using the password.

----End

### 20.4.2.3 How Do I Collect Logs of Nodes in a CCE Cluster?

The following tables list log files of CCE nodes.

**Table 20-1** Node logs

Name	Path
kubelet log	<ul style="list-style-type: none"> <li>For clusters of v1.21 or later: <b><code>/var/log/cce/kubernetes/kubelet.log</code></b></li> <li>For clusters of v1.19 or earlier: <b><code>/var/paas/sys/log/kubernetes/kubelet.log</code></b></li> </ul>
kube-proxy log	<ul style="list-style-type: none"> <li>For clusters of v1.21 or later: <b><code>/var/log/cce/kubernetes/kube-proxy.log</code></b></li> <li>For clusters of v1.19 or earlier: <b><code>/var/paas/sys/log/kubernetes/kube-proxy.log</code></b></li> </ul>
yangtse log (networking)	<ul style="list-style-type: none"> <li>For clusters of v1.21 or later: <b><code>/var/log/cce/yangtse</code></b></li> <li>For clusters of v1.19 or earlier: <b><code>/var/paas/sys/log/yangtse</code></b></li> </ul>
canal log	<ul style="list-style-type: none"> <li>For clusters of v1.21 or later: <b><code>/var/log/cce/canal</code></b></li> <li>For clusters of v1.19 or earlier: <b><code>/var/paas/sys/log/canal</code></b></li> </ul>
System logs	<b><code>/var/log/messages</code></b>

**Table 20-2** Add-on logs

Name	Path
everest log	<ul style="list-style-type: none"> <li>For v2.1.41 or later: <ul style="list-style-type: none"> <li>everest-csi-driver: <b><code>/var/log/cce/kubernetes</code></b></li> <li>everest-csi-controller: <b><code>/var/paas/sys/log/kubernetes</code></b></li> </ul> </li> <li>For version earlier than v2.1.41: <ul style="list-style-type: none"> <li>everest-csi-driver: <b><code>/var/log/cce/everest-csi-driver</code></b></li> <li>everest-csi-controller: <b><code>/var/paas/sys/log/everest-csi-controller</code></b></li> </ul> </li> </ul>
npd log	<ul style="list-style-type: none"> <li>For v1.18.16 or later: <b><code>/var/paas/sys/log/kubernetes</code></b></li> <li>For versions earlier than v1.18.16: <b><code>/var/paas/sys/log/cceaddon-npd</code></b></li> </ul>
cce-hpa-controller log	<ul style="list-style-type: none"> <li>For v1.3.12 or later: <b><code>/var/paas/sys/log/kubernetes</code></b></li> <li>For versions earlier than v1.3.12: <b><code>/var/paas/sys/log/ccehpa-controller</code></b></li> </ul>



## 20.4.2.4 What Should I Do If the vdb Disk of a Node Is Damaged and the Node Cannot Be Recovered After Reset?

### Symptom

The vdb disk of a node is damaged and the node cannot be recovered after reset.

### Error Scenarios

- On a normal node, delete the LV and VG. The node is unavailable.
- Reset an abnormal node, and a syntax error is reported. The node is unavailable.

The following figure shows the details.

```

vgcreate VG_new PV ...
create volume group error
, skip pause's work in case of failed dependency docker, skip fuxi's work in case of failed dependency docker, sk
work in case of failed dependency kubelet, skip kube-proxy's work in case of failed dependency config-prepare, sk
ork in case of failed dependency config-prepare, skip canal-agent's work in case of failed dependency fuxi, skip c
work in case of failed dependency config-prepare, skip docker's work in case of failed dependency config-prepare,
s work in case of failed dependency config-prepare]
18525 17:22:55.835685 7116 install.go:36] install failed
Install Failed: [Install config-prepare failed: exit status 1, output: [Mon May 25 17:22:53 CST 2020] start inst
pare
success download the file
success download the file
success download the file
success download the file
success download the file
success download the file
success download the file
success download the file
Checking device: /dev/vda
Raw disk /dev/vda has been partition, will skip this device
Checking device: /dev/vdb
Detected paas disk: /dev/vdb
Use to config lv(eg. docker(direct-lvm),kubelet,user)
No command with matching syntax recognised. Run 'vgcreate --help' for more information.
Correct command syntax is:
vgcreate VG_new PV ...

create volume group error
, skip pause's work in case of failed dependency docker, skip fuxi's work in case of failed dependency docker, sk
work in case of failed dependency kubelet, skip kube-proxy's work in case of failed dependency config-prepare, sk
ork in case of failed dependency config-prepare, skip canal-agent's work in case of failed dependency fuxi, skip c
work in case of failed dependency config-prepare, skip docker's work in case of failed dependency config-prepare,
s work in case of failed dependency config-prepare]

```

### Fault Locating

If the volume group (VG) on the node is deleted or damaged and cannot be identified, you need to manually restore the VG first to prevent your data disks from being formatted by mistake during the reset.

### Solution

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

**Step 2** Create a PV and a VG again. In this example, the following error message is displayed:

```

root@host1:~# pvcreate /dev/vdb
Device /dev/vdb excluded by a filter

```

This is because the added disk is created on another VM and has a partition table. The current VM cannot identify the partition table of the disk. You need to run the **parted** commands for three times to re-create the partition table.

```

root@host1:~# parted /dev/vdb
GNU Parted 3.2
Using /dev/vdb
Welcome to GNU Parted! Type 'help' to view a list of commands.

```

```
(parted) mklabel msdos
Warning: The existing disk label on /dev/vdb will be destroyed and all data on this disk will be lost. Do you
want to continue?
Yes/No? yes
(parted) quit
Information: You may need to update /etc/fstab.
```

Run **pvcreate** again. When the system asks you whether to erase the DOS signature, enter **y**. The disk is created as a PV.

```
root@host1:~# pvcreate /dev/vdb
WARNING: dos signature detected on /dev/vdb at offset 510. Wipe it? [y/n]: y
Wiping dos signature on /dev/vdb.
Physical volume "/dev/vdb" successfully created
```

### Step 3 Create a VG.

Check the Docker disks of the node. If the disks are **/dev/vdb** and **/dev/vdc**, run the following command:

```
root@host1:~# vgcreate vgpaas /dev/vdb /dev/vdc
```

If there is only the **/dev/vdb** disk, run the following command:

```
root@host1:~# vgcreate vgpaas /dev/vdb
```

After the creation is complete, reset the node.

----End

## 20.4.2.5 What Should I Do If I/O Suspension Occasionally Occurs When SCSI EVS Disks Are Used?

### Symptom

When SCSI EVS disks are used and containers are created and deleted on a CentOS node, the disks are frequently mounted and unmounted. The read/write rate of the system disk may instantaneously surge. As a result, the system is suspended, affecting the normal node running.

When this problem occurs, the following information is displayed in the dmesg log:

```
Attached SCSI disk
task jdb2/xxx blocked for more than 120 seconds.
```

Example:

```
1128163.173120] sd 2:0:0:0: [sda] Write Protect IS 011
1128163.173457] sd 2:0:0:0: [sda] Mode Sense: 69 00 00 08
1128163.173573] sd 2:0:0:0: [sda] Write cache: disabled, read cache: enabled, doesn't support DPO or FUA
1128163.176426] sd 2:0:0:0: [sda] Attached SCSI disk
1128350.437941] INFO: task jbd2/dm-1-8:1604 blocked for more than 120 seconds.
1128350.438267] "echo 0 > /proc/sys/kernel/hung_task_timeout_secs" disables this message.
1128350.438564] jbd2/dm-1-8 D ffff9ede7f8420e0 0 1604 2 0x00000000
1128350.438829] Call Trace:
1128350.439120] [<ffffffffffaab5a585>] ? blk_mq_dispatch_rq_list+0x325/0x620
1128350.439394] [<ffffffffffaaf7f229>] schedule+0x29/0x70
```

### Possible Cause

After a PCI device is hot added to BUS 0, the Linux OS kernel will traverse all the PCI bridges mounted to BUS 0 for multiple times, and these PCI bridges cannot work properly during this period. During this period, if the PCI bridge used by the device is updated, due to a kernel defect, the device considers that the PCI bridge

is abnormal, and the device enters a fault mode and cannot work normally. If the front end is writing data into the PCI configuration space for the back end to process disk I/Os, the write operation may be deleted. As a result, the back end cannot receive notifications to process new requests on the I/O ring. Finally, the front-end I/O suspension occurs.

This problem is caused by a Linux kernel defect. For details, see the [defects in Linux distributions](#).

## Impact

CentOS Linux kernels of versions earlier than 3.10.0-1127.el7 are affected.

## Solution

Upgrade the kernel to a later version **by resetting the node**.

### 20.4.2.6 How Do I Fix an Abnormal Container or Node Due to No Thin Pool Disk Space?

#### Problem Description

When the disk space of a thin pool on a node is about to be used up, the following exceptions occasionally occur:

Files or directories fail to be created in the container, the file system in the container is read-only, the node is tainted disk-pressure, or the node is unavailable.

You can run the **docker info** command on the node to view the used and remaining thin pool space to locate the fault. The following figure is an example.

```
Storage Driver: devicemapper
Pool Name: vgpaas-thinpool
Pool Blocksize: 524.3kB
Base Device Size: 10.74GB
Backing Filesystem: ext4
Udev Sync Supported: true
Data Space Used: 7.794GB
Data Space Total: 71.94GB
Data Space Available: 64.15GB
Metadata Space Used: 3.076MB
Metadata Space Total: 3.221GB
Metadata Space Available: 3.218GB
Thin Pool Minimum Free Space: 7.194GB
Deferred Removal Enabled: true
Deferred Deletion Enabled: true
Deferred Deleted Device Count: 0
Library Version: 1.02.146-RHEL7 (2018-01-22)
```

#### Possible Cause

When Docker device mapper is used, although you can configure the **basesize** parameter to 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 only after a while. If **the number of service 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.

## Solution

When the thin pool space of a node is used up, some services can be migrated to other nodes to quickly recover services. But you are advised to use the following solutions to resolve the root cause:

### Solution 1:

Properly plan the service distribution and data plane disk space to avoid the scenario where **the number of service containers multiplied by basesize** is greater than the thin pool size of the node. To expand the thin pool size, perform the following steps:

- Step 1** Expand the capacity of the data disk on the EVS console.
- Step 2** Log in to the CCE console and click the cluster. In the navigation pane, choose **Nodes**. Click **More > Sync Server Data** in the row containing the target node.
- Step 3** Log in to the target node.
- Step 4** Run the **lsblk** command to check the block device information of the node.

A data disk is divided depending on the container storage **Rootfs**:

- **Overlays:** No independent thin pool is allocated. Image data is stored in the **dockersys** disk.

```
lsblk
NAME MAJ:MIN RM SIZE RO TYPE MOUNTPOINT
sda 8:0 0 50G 0 disk
└─sda1 8:1 0 50G 0 part /
sdb 8:16 0 200G 0 disk
├─vgpaas-dockersys 253:0 0 90G 0 lvm /var/lib/docker # Space used by the container
engine
└─vgpaas-kubernetes 253:1 0 10G 0 lvm /mnt/paas/kubernetes/kubelet # Space used by
Kubernetes
```

Run the following commands on the node to add the new disk capacity to the **dockersys** disk:

```
pvresize /dev/sdb
lvextend -l+100%FREE -n vgpaas/dockersys
resize2fs /dev/vgpaas/dockersys
```

- **Devicemapper:** A thin pool is allocated to store image data.

```
lsblk
NAME MAJ:MIN RM SIZE RO TYPE MOUNTPOINT
sda 8:0 0 50G 0 disk
└─sda1 8:1 0 50G 0 part /
sdb 8:16 0 200G 0 disk
├─vgpaas-dockersys 253:0 0 18G 0 lvm /var/lib/docker
├─vgpaas-thinpool_tmeta 253:1 0 3G 0 lvm
└─vgpaas-thinpool 253:3 0 67G 0 lvm # Thin pool space.
```

```

...
vgpaas-thinpool_tdata 253:2 0 67G 0 lvm
└─vgpaas-thinpool 253:3 0 67G 0 lvm
...
vgpaas-kubernetes 253:4 0 10G 0 lvm /mnt/paas/kubernetes/kubelet

```

- Run the following commands on the node to add the new disk capacity to the **thinpool** disk:  

```
pvresize /dev/sdb
lvextend -l+100%FREE -n vgpaas/thinpool
```
- Run the following commands on the node to add the new disk capacity to the **dockersys** disk:  

```
pvresize /dev/sdb
lvextend -l+100%FREE -n vgpaas/dockersys
resize2fs /dev/vgpaas/dockersys
```

----End

### Solution 2:

Create and delete files in service containers in the local storage (such as emptyDir and hostPath) or cloud storage directory mounted to the container. Such files do not occupy the thin pool space.

### Solution 3:

If the OS uses OverlayFS, services can be deployed on such nodes to prevent the problem that the disk space occupied by files created or deleted in the container is not released immediately.

## 20.4.2.7 How Do I Rectify Failures When the NVIDIA Driver Is Used to Start Containers on GPU Nodes?

### Did a Resource Scheduling Failure Event Occur on a Cluster Node?

#### Symptom

A node is running properly and has GPU resources. However, the following error information is displayed:

0/9 nodes are available: 9 insufficient nvidia.com/gpu

#### Analysis

1. Check whether the node is attached with NVIDIA label.

```

[root@chengyindu-test-98835 ~]# kubectl get node --show-labels
NAME STATUS ROLES AGE VERSION LABELS
172.16.0.188 Ready <none> 6h26m v1.13.10-r1-CC2.0.28.B001 accelerator=nvidia-p100 beta.kubernetes.io/arch=amd64,beta.kubernetes.io/os=linux,failure-domain.beta.kubernetes.io/is-baremetal=false,failure-domain.beta.kubernetes.io/region=cn-east-2,failure-domain.beta.kubernetes.io/zone=cn-east-2b,kubernetes.io/availablezone=cn-east-2b,kubernetes.io/eniquota=12,kubernetes.io/hostname=172.16.0.188,node.kubernetes.io/subnetid=4883a3c2-f89f-412d-bd3a-5a2892c5833a,os.architecture=amd64,os.name=EulerOS_2.0_SP5,os.version=3.10.0-862.14.1.2.el249.eulerosv2r7.x86_64
[root@chengyindu-test-98835 ~]#

```

2. Check whether the NVIDIA driver is running properly.

Log in to the node where the add-on is running and view the driver installation log in the following path:

```
/opt/cloud/cce/nvidia/nvidia_installer.log
```

View standard output logs of the NVIDIA container.

Filter the container ID by running the following command:

```
docker ps -a | grep nvidia
```

View logs by running the following command:

```
docker logs Container ID
```

## What Should I Do If the NVIDIA Version Reported by a Service and the CUDA Version Do Not Match?

Run the following command to check the CUDA version in the container:

```
cat /usr/local/cuda/version.txt
```

Check whether the CUDA version supported by the NVIDIA driver version of the node where the container is located contains the CUDA version of the container.

## Helpful Links

[What Should I Do If an Error Occurs When Deploying a Service on the GPU Node?](#)

## 20.4.3 Specification Change

### 20.4.3.1 How Do I Change the Node Specifications in a CCE Cluster?

#### Solution

---

 **CAUTION**

If the node whose specifications need to be changed is accepted into the cluster for management, remove the node from the cluster and then change the node specifications to avoid affecting services.

---

- Step 1** Log in to the CCE console and click the cluster. In the navigation pane, choose **Nodes**. Click the name of the node to display the ECS details page.
- Step 2** In the upper right corner of the ECS details page, click **Stop**. After the ECS is stopped, choose **More > Modify Specifications**.
- Step 3** On the **Modify ECS Specifications** page, select a flavor name and click **Submit** to finish the specification modification. Return to ECS list page and choose **More > Start** to start the ECS.
- Step 4** Log in to the CCE console and click the cluster. In the navigation pane, choose **Nodes**. Locate the target node in the node list, and click **Sync Server Data** in the **Operation** column. After the synchronization is complete, you can view that the node specifications are the same as the modified specifications of the ECS.

----End

## Common Issues

After the specifications of a node configured with CPU management policies are changed, the node may fail to be rebooted or workloads may fail to be created. In

this case, see [What Should I Do If I Fail to Restart or Create Workloads on a Node After Modifying the Node Specifications?](#) to rectify the fault.

### 20.4.3.2 What Should I Do If I Fail to Restart or Create Workloads on a Node After Modifying the Node Specifications?

#### Context

The kubelet option **cpu-manager-policy** defaults to **static**, allowing pods with certain resource characteristics to be granted increased CPU affinity and exclusivity on the node. If you modify CCE node specifications on the ECS console, the original CPU information does not match the new CPU information. As a result, workloads on the node cannot be restarted or created.

For more information, see [Control CPU Management Policies on the Node](#).

#### Impact

The clusters that have enabled a CPU management policy will be affected.

#### Solution

**Step 1** Log in to the CCE node (ECS) and delete the **cpu\_manager\_state** file.

Example command for the file deletion:  

```
rm -rf /mnt/paas/kubernetes/kubelet/cpu_manager_state
```

**Step 2** Restart the node or kubelet. The following is the kubelet restart command:

```
systemctl restart kubelet
```

**Step 3** Verify that workloads on the node can be successfully restarted or created.

----End

## 20.5 Node Pool

### 20.5.1 What Should I Do If No Node Creation Record Is Displayed When the Node Pool Is Being Expanding?

#### Symptom

The node pool keeps being in the expanding state, but no node creation record is displayed in the operation record.

#### Troubleshooting

Check and rectify the following faults:

- Whether the specifications configured for the node pool are insufficient.
- Whether the ECS or memory quota of the tenant is insufficient.
- The ECS capacity verification of the tenant may fail if too many nodes are created at a time.

## Solution

- If the resources of the ECS flavor cannot meet service requirements, use ECSs of another flavor.
- If the ECS or memory quota is insufficient, increase the quota.
- If the ECS capacity verification fails, perform the verification again.

## 20.6 Workload

### 20.6.1 Workload Abnormalities

#### 20.6.1.1 How Do I Use Events to Fix Abnormal Workloads?

If a workload is abnormal, you can first check the pod events to locate the fault and then rectify the fault by referring to [Table 20-3](#).

**Table 20-3** Troubleshooting methods

Event Information	Pod Status	Solution
PodsScheduling failed	Pending	For details, see <a href="#">What Should I Do If Pod Scheduling Fails?</a>
PodsFailed to pull image Failed to re-pull image	FailedPullImage ImagePullBackOff	For details, see <a href="#">What Should I Do If a Pod Fails to Pull the Image?</a>
PodsCreation failed Failed to restart container	CreateContainerError CrashLoopBackOff	For details, see <a href="#">What Should I Do If Container Startup Fails?</a>
The pod status is <b>Evicted</b> , and the pod keeps being evicted.	Evicted	For details, see <a href="#">What Should I Do If a Pod Fails to Be Evicted?</a>
The storage volume fails to be mounted to the pod.	Pending	For details, see <a href="#">What Should I Do If a Storage Volume Cannot Be Mounted or the Mounting Times Out?</a>
The pod stays <b>Creating</b> .	Creating	For details, see <a href="#">What Should I Do If a Workload Remains in the Creating State?</a>
The pod stays <b>Terminating</b> .	Terminating	For details, see <a href="#">What Should I Do If Pods in the Terminating State Cannot Be Deleted?</a>



Event Information	Pod Status	Solution
The pod status is <b>Stopped</b> .	Stopped	For details, see <a href="#">What Should I Do If a Workload Is Stopped Caused by Pod Deletion?</a> .

## Viewing Pod Events

Run the **kubect**l describe pod *{pod-name}* command to view pod events, or log in to the CCE console and view pod events on the workload details page.

```
$ kubect
```

### 20.6.1.2 What Should I Do If Pod Scheduling Fails?

#### Fault Locating

If the pod is in the **Pending** state and the event contains pod scheduling failure information, locate the cause based on the event information. For details about how to view events, see [How Do I Use Events to Fix Abnormal Workloads?](#)

#### Troubleshooting Process

Determine the cause based on the event information, as listed in [Table 20-4](#).

**Table 20-4** Pod scheduling failure

Event Information	Cause and Solution
no nodes available to schedule pods.	No node is available in the cluster. <a href="#">Check Item 1: Whether a Node Is Available in the Cluster</a>
0/2 nodes are available: 2 Insufficient cpu. 0/2 nodes are available: 2 Insufficient memory.	Node resources (CPU and memory) are insufficient. <a href="#">Check Item 2: Whether Node Resources (CPU and Memory) Are Sufficient</a>
0/2 nodes are available: 1 node(s) didn't match node selector, 1 node(s) didn't match pod affinity rules, 1 node(s) didn't match pod affinity/anti-affinity.	The node and pod affinity configurations are mutually exclusive. No node meets the pod requirements. <a href="#">Check Item 3: Affinity and Anti-Affinity Configuration of the Workload</a>

Event Information	Cause and Solution
0/2 nodes are available: 2 node(s) had volume node affinity conflict.	The EVS volume mounted to the pod and the node are not in the same AZ. <a href="#">Check Item 4: Whether the Workload's Volume and Node Reside in the Same AZ</a>
0/1 nodes are available: 1 node(s) had taints that the pod didn't tolerate.	Taints exist on the node, but the pod cannot tolerate these taints. <a href="#">Check Item 5: Taint Toleration of Pods</a>
0/7 nodes are available: 7 Insufficient ephemeral-storage.	The ephemeral storage space of the node is insufficient. <a href="#">Check Item 6: Ephemeral Volume Usage</a>
0/1 nodes are available: 1 everest driver not found at node	The everest-csi-driver on the node is not in the running state. <a href="#">Check Item 7: Whether everest Works Properly</a>
Failed to create pod sandbox: ... Create more free space in thin pool or use dm.min_free_space option to change behavior	The node thin pool space is insufficient. <a href="#">Check Item 8: Thin Pool Space</a>
0/1 nodes are available: 1 Too many pods.	The number of pods scheduled to the node exceeded the maximum number allowed by the node. <a href="#">Check Item 9: Number of Pods Scheduled onto the Node</a>

## Check Item 1: Whether a Node Is Available in the Cluster

Log in to the CCE console and check whether the node status is **Available**. Alternatively, run the following command to check whether the node status is **Ready**:

```
$ kubectl get node
NAME STATUS ROLES AGE VERSION
192.168.0.37 Ready <none> 21d v1.19.10-r1.0.0-source-121-gb9675686c54267
192.168.0.71 Ready <none> 21d v1.19.10-r1.0.0-source-121-gb9675686c54267
```

If the status of all nodes is **Not Ready**, no node is available in the cluster.

### Solution

- Add a node. If an affinity policy is not configured for the workload, the pod will be automatically migrated to the new node to ensure that services are running properly.

- Locate the unavailable node and rectify the fault. For details, see [What Should I Do If a Cluster Is Available But Some Nodes Are Unavailable?](#)
- Reset the unavailable node.

## Check Item 2: Whether Node Resources (CPU and Memory) Are Sufficient

**0/2 nodes are available: 2 Insufficient cpu.** This means insufficient CPUs.

**0/2 nodes are available: 2 Insufficient memory.** This means insufficient memory.

If the resources requested by the pod exceed the allocatable resources of the node where the pod runs, the node cannot provide the resources required to run new pods and pod scheduling onto the node will definitely fail.

If the number of resources that can be allocated to a node is less than the number of resources that a pod requests, the node does not meet the resource requirements of the pod. As a result, the scheduling fails.

### Solution

Add nodes to the cluster. Scale-out is the common solution to insufficient resources.

## Check Item 3: Affinity and Anti-Affinity Configuration of the Workload

Inappropriate affinity policies will cause pod scheduling to fail.

Example:

An anti-affinity relationship is established between workload 1 and workload 2. Workload 1 is deployed on node 1 while workload 2 is deployed on node 2.

When you try to deploy workload 3 on node 1 and establish an affinity relationship with workload 2, a conflict occurs, resulting in a workload deployment failure.

**0/2 nodes are available: 1 node(s) didn't match `node selector`, 1 node(s) didn't match `pod affinity rules`, 1 node(s) didn't match `pod affinity/anti-affinity`.**

- **node selector** indicates that the node affinity is not met.
- **pod affinity rules** indicate that the pod affinity is not met.
- **pod affinity/anti-affinity** indicates that the pod affinity/anti-affinity is not met.

### Solution

- When adding workload-workload affinity and workload-node affinity policies, ensure that the two types of policies do not conflict each other. Otherwise, workload deployment will fail.

- If the workload has a node affinity policy, make sure that **supportContainer** in the label of the affinity node is set to **true**. Otherwise, pods cannot be scheduled onto the affinity node and the following event is generated:

```
No nodes are available that match all of the following predicates: MatchNode Selector, NodeNotSupportsContainer
```

If the value is **false**, the scheduling fails.

## Check Item 4: Whether the Workload's Volume and Node Reside in the Same AZ

**0/2 nodes are available: 2 node(s) had volume node affinity conflict.** An affinity conflict occurs between volumes and nodes. As a result, the scheduling fails.

This is because EVS disks cannot be attached to nodes across AZs. For example, if the EVS volume is located in AZ 1 and the node is located in AZ 2, scheduling fails.

The EVS volume created on CCE has affinity settings by default, as shown below.

```
kind: PersistentVolume
apiVersion: v1
metadata:
 name: pvc-c29bfac7-efa3-40e6-b8d6-229d8a5372ac
spec:
 ...
 nodeAffinity:
 required:
 nodeSelectorTerms:
 - matchExpressions:
 - key: failure-domain.beta.kubernetes.io/zone
 operator: In
 values:
 -
```

### Solution

In the AZ where the workload's node resides, create a volume. Alternatively, create an identical workload and select an automatically assigned cloud storage volume.

## Check Item 5: Taint Toleration of Pods

**0/1 nodes are available: 1 node(s) had taints that the pod didn't tolerate.** This means the node is tainted and the pod cannot be scheduled to the node.

Check the taints on the node. If the following information is displayed, taints exist on the node:

```
$ kubectl describe node 192.168.0.37
Name: 192.168.0.37
...
Taints: key1=value1:NoSchedule
...
```

In some cases, the system automatically adds a taint to a node. The current built-in taints include:

- `node.kubernetes.io/not-ready`: The node is not ready.
- `node.kubernetes.io/unreachable`: The node controller cannot access the node.
- `node.kubernetes.io/memory-pressure`: The node has memory pressure.
- `node.kubernetes.io/disk-pressure`: The node has disk pressure. Follow the instructions described in [Check Item 4: Whether the Node Disk Space Is Insufficient](#) to handle it.
- `node.kubernetes.io/pid-pressure`: The node is under PID pressure.
- `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 **cloud-controller-manager** initializes the node, kubelet deletes the taint.

### Solution

To schedule the pod to the node, use either of the following methods:

- If the taint is added by a user, you can delete the taint on the node. If the taint is **automatically added by the system**, the taint will be automatically deleted after the fault is rectified.
- Specify a toleration for the pod containing the taint. For details, see [Taints and Tolerations](#).

```
apiVersion: v1
kind: Pod
metadata:
 name: nginx
spec:
 containers:
 - name: nginx
 image: nginx:alpine
 tolerations:
 - key: "key1"
 operator: "Equal"
 value: "value1"
 effect: "NoSchedule"
```

## Check Item 6: Ephemeral Volume Usage

**0/7 nodes are available: 7 Insufficient ephemeral-storage.** This means insufficient ephemeral storage of the node.

Check whether the size of the ephemeral volume in the pod is limited. If the size of the ephemeral volume required by the application exceeds the existing capacity of the node, the application cannot be scheduled. To solve this problem, change the size of the ephemeral volume or expand the disk capacity of the node.

```
apiVersion: v1
kind: Pod
metadata:
 name: frontend
spec:
 containers:
 - name: app
 image: images.my-company.example/app:v4
 resources:
 requests:
 ephemeral-storage: "2Gi"
 limits:
 ephemeral-storage: "4Gi"
 volumeMounts:
 - name: ephemeral
 mountPath: "/tmp"
 volumes:
 - name: ephemeral
 emptyDir: {}
```

To obtain the total capacity (**Capacity**) and available capacity (**Allocatable**) of the temporary volume mounted to the node, run the **kubectl describe node** command, and view the application value and limit value of the temporary volume mounted to the node.

The following is an example of the output:

```
...
Capacity:
cpu: 4
ephemeral-storage: 61607776Ki
hugepages-1Gi: 0
hugepages-2Mi: 0
localssd: 0
localvolume: 0
memory: 7614352Ki
pods: 40
Allocatable:
cpu: 3920m
ephemeral-storage: 56777726268
hugepages-1Gi: 0
hugepages-2Mi: 0
localssd: 0
localvolume: 0
memory: 6180752Ki
pods: 40
...
Allocated resources:
(Total limits may be over 100 percent, i.e., overcommitted.)
Resource Requests Limits

cpu 1605m (40%) 6530m (166%)
memory 2625Mi (43%) 5612Mi (92%)
ephemeral-storage 0 (0%) 0 (0%)
hugepages-1Gi 0 (0%) 0 (0%)
hugepages-2Mi 0 (0%) 0 (0%)
localssd 0 0
localvolume 0 0
Events: <none>
```

## Check Item 7: Whether everest Works Properly

**0/1 nodes are available: 1 everest driver not found at node.** This means the everest-csi-driver of everest is not started properly on the node.

Check the daemon named **everest-csi-driver** in the kube-system namespace and check whether the pod is started properly. If not, delete the pod. The daemon will restart the pod.

## Check Item 8: Thin Pool Space

A data disk dedicated for kubelet and the container engine will be attached to a new node. If the data disk space is insufficient, the pod cannot be created.

### Solution 1: Clearing images

Perform the following operations to clear unused images:

- Nodes that use containerd
  - a. Obtain local images on the node.  
cricctl images -v
  - b. Delete the images that are not required by image ID.  
cricctl rmi *Image ID*
- Nodes that use Docker
  - a. Obtain local images on the node.  
docker images

- b. Delete the images that are not required by image ID.  
`docker rmi Image ID`

 **NOTE**

Do not delete system images such as the cce-pause image. Otherwise, pods may fail to be created.

**Solution 2: Expanding the disk capacity**

To expand a disk capacity, perform the following steps:

- Step 1** Expand the capacity of the data disk on the EVS console.
- Step 2** Log in to the CCE console and click the cluster. In the navigation pane, choose **Nodes**. Click **More > Sync Server Data** in the row containing the target node.
- Step 3** Log in to the target node.
- Step 4** Run the **lsblk** command to check the block device information of the node.

A data disk is divided depending on the container storage **Rootfs**:

- **Overlayfs:** No independent thin pool is allocated. Image data is stored in the **dockersys** disk.

```
lsblk
NAME MAJ:MIN RM SIZE RO TYPE MOUNTPOINT
sda 8:0 0 50G 0 disk
├─sda1 8:1 0 50G 0 part /
sdb 8:16 0 200G 0 disk
├─vgpaas-dockersys 253:0 0 90G 0 lvm /var/lib/docker # Space used by the container engine
└─vgpaas-kubernetes 253:1 0 10G 0 lvm /mnt/paas/kubernetes/kubelet # Space used by Kubernetes
```

Run the following commands on the node to add the new disk capacity to the **dockersys** disk:

```
pvresize /dev/sdb
lvextend -l+100%FREE -n vgpaas/dockersys
resize2fs /dev/vgpaas/dockersys
```

- **Devicemapper:** A thin pool is allocated to store image data.

```
lsblk
NAME MAJ:MIN RM SIZE RO TYPE MOUNTPOINT
sda 8:0 0 50G 0 disk
├─sda1 8:1 0 50G 0 part /
sdb 8:16 0 200G 0 disk
├─vgpaas-dockersys 253:0 0 18G 0 lvm /var/lib/docker
├─vgpaas-thinpool_tmeta 253:1 0 3G 0 lvm
├─vgpaas-thinpool 253:3 0 67G 0 lvm # Thin pool space.
├─...
├─vgpaas-thinpool_tdata 253:2 0 67G 0 lvm
├─vgpaas-thinpool 253:3 0 67G 0 lvm
├─...
└─vgpaas-kubernetes 253:4 0 10G 0 lvm /mnt/paas/kubernetes/kubelet
```

- Run the following commands on the node to add the new disk capacity to the **thinpool** disk:

```
pvresize /dev/sdb
lvextend -l+100%FREE -n vgpaas/thinpool
```

- Run the following commands on the node to add the new disk capacity to the **dockersys** disk:

```
pvresize /dev/sdb
lvextend -l+100%FREE -n vgpaas/dockersys
resize2fs /dev/vgpaas/dockersys
```

----End

## Check Item 9: Number of Pods Scheduled onto the Node

**0/1 nodes are available: 1 Too many pods.** indicates excessive number of pods have been scheduled to the node.

When creating a node, configure **Max. Pods** in **Advanced Settings** to specify the maximum number of pods that can run properly on the node. The default value varies with the node flavor. You can change the value as needed.

On the **Nodes** page, obtain the **Pods (Allocated/Total)** value of the node, and check whether the number of pods scheduled onto the node has reached the upper limit. If so, add nodes or change the maximum number of pods.

To change the maximum number of pods that can run on a node, do as follows:

- For nodes in the default node pool: Change the **Max. Pods** value when resetting the node.
- For nodes in a customized node pool: Change the value of the node pool parameter **max-pods**.

### 20.6.1.3 What Should I Do If a Pod Fails to Pull the Image?

#### Fault Locating

When a workload enters the state of "Pod not ready: Back-off pulling image "xxxxx", a Kubernetes event of **PodsFailed to pull image** or **Failed to re-pull image** will be reported. For details about how to view Kubernetes events, see [Viewing Pod Events](#).

#### Troubleshooting Process

Determine the cause based on the event information, as listed in [Table 20-5](#).

**Table 20-5** FailedPullImage

Event Information	Cause and Solution
Failed to pull image "xxx": rpc error: code = Unknown desc = Error response from daemon: Get xxx: denied: You may not login yet	You have not logged in to the image repository. <a href="#">Check Item 1: Whether imagePullSecret Is Specified When You Use kubectl to Create a Workload</a>
Failed to pull image "nginx:v1.1": rpc error: code = Unknown desc = Error response from daemon: Get https://registry-1.docker.io/v2/: dial tcp: lookup registry-1.docker.io: no such host	The image address is incorrectly configured. <a href="#">Check Item 2: Whether the Image Address Is Correct When a Third-Party Image Is Used</a> <a href="#">Check Item 3: Whether an Incorrect Secret Is Used When a Third-Party Image Is Used</a>



Event Information	Cause and Solution
Failed create pod sandbox: rpc error: code = Unknown desc = failed to create a sandbox for pod "nginx-6dc48bf8b6-l8xrw": Error response from daemon: mkdir xxxxx: no space left on device	The disk space is insufficient. <a href="#">Check Item 4: Whether the Node Disk Space Is Insufficient</a>
Failed to pull image "xxx": rpc error: code = Unknown desc = error pulling image configuration: xxx x509: certificate signed by unknown authority	An unknown or insecure certificate is used by the third-party image repository from which the image is pulled. <a href="#">Check Item 5: Whether the Remote Image Repository Uses an Unknown or Insecure Certificate</a>
Failed to pull image "xxx": rpc error: code = Unknown desc = context canceled	The image size is too large. <a href="#">Check Item 6: Whether the Image Size Is Too Large</a>
Failed to pull image "docker.io/bitnami/nginx:1.22.0-debian-11-r3": rpc error: code = Unknown desc = Error response from daemon: Get https://registry-1.docker.io/v2/: net/http: request canceled while waiting for connection (Client.Timeout exceeded while awaiting headers)	<a href="#">Check Item 7: Connection to the Image Repository</a>
ERROR: toomanyrequests: Too Many Requests. Or you have reached your pull rate limit, you may increase the limit by authenticating an upgrading	The rate is limited because the number of image pull times reaches the upper limit. <a href="#">Check Item 8: Whether the Number of Public Image Pull Times Reaches the Upper Limit</a>

## Check Item 1: Whether imagePullSecret Is Specified When You Use kubectl to Create a Workload

If the workload status is abnormal and a Kubernetes event is displayed indicating that the pod fails to pull the image, check whether the **imagePullSecrets** field exists in the YAML file.

### Items to Check

- If an image needs to be pulled from SWR, the **name** parameter must be set to **default-secret**.

```
apiVersion: extensions/v1beta1
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

```

- If an image needs to be pulled from a third-party image repository, the **imagePullSecrets** parameter must be set to the created secret name.

When you use kubectl to create a workload from a third-party image, specify the **imagePullSecret** field, in which **name** indicates the name of the secret used to pull the image.

## Check Item 2: Whether the Image Address Is Correct When a Third-Party Image Is Used

CCE allows you to create workloads using images pulled from third-party image repositories.

Enter the third-party image address according to requirements. The format must be **ip:port/path/name:version** or **name:version**. If no tag is specified, **latest** is used by default.

- For a private repository, enter an image address in the format of **ip:port/path/name:version**.
- For an open-source Docker repository, enter an image address in the format of **name:version**, for example, **nginx:latest**.

The following information is displayed when you fail to pull an image due to incorrect image address provided.

```
Failed to pull image "nginx:v1.1": rpc error: code = Unknown desc = Error response from daemon: Get https://registry-1.docker.io/v2/: dial tcp: lookup registry-1.docker.io: no such host
```

### Solution

You can either edit your YAML file to change the image address or log in to the CCE console to replace the image on the **Upgrade** tab on the workload details page.

## Check Item 3: Whether an Incorrect Secret Is Used When a Third-Party Image Is Used

Generally, a third-party image repository can be accessed only after authentication (using your account and password). CCE uses the secret authentication mode to pull images. Therefore, you need to create a secret for an image repository before pulling images from the repository.

### Solution

If your secret is incorrect, images will fail to be pulled. In this case, create a new secret.

## Check Item 4: Whether the Node Disk Space Is Insufficient

If the Kubernetes event contains information "no space left on device", there is no disk space left for storing the image. As a result, the image will fail to be pulled. In this case, clear the image or expand the disk space to resolve this issue.

Failed create pod sandbox: rpc error: code = Unknown desc = failed to create a sandbox for pod "nginx-6dc48bf8b6-l8xrw": Error response from daemon: mkdir xxxxx: no space left on device

Run the following command to obtain the disk space for storing images on a node:

lvs

```
[root@zhouxu-20650 ~]# lvs
LV VG Attr LSize Pool Origin Data% Meta% Move Log Cpy%Sync Convert
kubernetes vgpaas -wi-ao--- <10.00g
thinpool vgpaas twi-aot--- 84.00g
 5.05 0.07
```

### Solution 1: Clearing images

Perform the following operations to clear unused images:

- Nodes that use containerd
  - a. Obtain local images on the node.  
crictl images -v
  - b. Delete the images that are not required by image ID.  
crictl rmi *Image ID*
- Nodes that use Docker
  - a. Obtain local images on the node.  
docker images
  - b. Delete the images that are not required by image ID.  
docker rmi *Image ID*

#### NOTE

Do not delete system images such as the cce-pause image. Otherwise, pods may fail to be created.

### Solution 2: Expanding the disk capacity

To expand a disk capacity, perform the following steps:

- Step 1** Expand the capacity of the data disk on the EVS console.
- Step 2** Log in to the CCE console and click the cluster. In the navigation pane, choose **Nodes**. Locate the row containing the target node and choose **More > Sync Server Data** in the **Operation** column.
- Step 3** Log in to the target node.
- Step 4** Run **lsblk** to view the block device information of the node.

A data disk is divided depending on the container storage **Rootfs**:

- **Overlayfs**: No independent thin pool is allocated. Image data is stored in the **dockersys** disk.

```
lsblk
NAME MAJ:MIN RM SIZE RO TYPE MOUNTPOINT
sda 8:0 0 50G 0 disk
├─sda1 8:1 0 50G 0 part /
sdb 8:16 0 200G 0 disk
├─vgpaas-dockersys 253:0 0 90G 0 lvm /var/lib/docker # Space used by the container
engine
└─vgpaas-kubernetes 253:1 0 10G 0 lvm /mnt/paas/kubernetes/kubelet # Space used by
Kubernetes
```

Run the following commands on the node to add the new disk capacity to the **dockersys** disk:

```
pvresize /dev/sdb
lvextend -l+100%FREE -n vgpaas/dockersys
resize2fs /dev/vgpaas/dockersys
```

- **Devicemapper: A thin pool is allocated to store image data.**

```
lsblk
NAME MAJ:MIN RM SIZE RO TYPE MOUNTPOINT
sda 8:0 0 50G 0 disk
├─sda1 8:1 0 50G 0 part /
sdb 8:16 0 200G 0 disk
├─vgpaas-dockersys 253:0 0 18G 0 lvm /var/lib/docker
├─vgpaas-thinpool_tmeta 253:1 0 3G 0 lvm
├─vgpaas-thinpool 253:3 0 67G 0 lvm # Thin pool space.
...
├─vgpaas-thinpool_tdata 253:2 0 67G 0 lvm
├─vgpaas-thinpool 253:3 0 67G 0 lvm
...
└─vgpaas-kubernetes 253:4 0 10G 0 lvm /mnt/paas/kubernetes/kubelet
```

- Run the following commands on the node to add the new disk capacity to the **thinpool** disk:

```
pvresize /dev/sdb
lvextend -l+100%FREE -n vgpaas/thinpool
```

- Run the following commands on the node to add the new disk capacity to the **dockersys** disk:

```
pvresize /dev/sdb
lvextend -l+100%FREE -n vgpaas/dockersys
resize2fs /dev/vgpaas/dockersys
```

----End

## Check Item 5: Whether the Remote Image Repository Uses an Unknown or Insecure Certificate

When a pod pulls an image from a third-party image repository that uses an unknown or insecure certificate, the image fails to be pulled from the node. The pod event list contains the event "Failed to pull the image" with the cause "x509: certificate signed by unknown authority".

### NOTE

The security of EulerOS 2.9 images is enhanced. Some insecure or expired certificates are removed from the system. It is normal that this error is reported in EulerOS 2.9 but not on some third-party images on other types of nodes. You can also perform the following operations to rectify the fault.

### Solution

- Step 1** Check the IP address and port number of the third-party image server for which the error message "unknown authority" is displayed.

You can see the IP address and port number of the third-party image server for which the error is reported in the event information "Failed to pull image".

```
Failed to pull image "bitnami/redis-cluster:latest": rpc error: code = Unknown desc = error pulling image configuration: Get https://production.cloudflare.docker.com/registry-v2/docker/registry/v2/blobs/sha256/e8/e83853f03a2e792614e7c1e6de75d63e2d6d633b4e7c39b9d700792ee50f7b56/data?verify=1636972064-AQbL5RActnudzV%2F3EShZwnqOe8%3D: x509: certificate signed by unknown authority
```

The IP address of the third-party image server is *production.cloudflare.docker.com*, and the default HTTPS port number is *443*.

**Step 2** Load the root certificate of the third-party image server to the node where the third-party image is to be downloaded.

Run the following commands on the EulerOS and CentOS nodes with *{server\_url}*: *{server\_port}* replaced with the IP address and port number obtained in Step 1, for example, **production.cloudflare.docker.com:443**:

If the container engine of the node is containerd, replace **systemctl restart docker** with **systemctl restart containerd**.

```
openssl s_client -showcerts -connect {server_url}:{server_port} < /dev/null | sed -ne '/-BEGIN CERTIFICATE-/,/-END CERTIFICATE-/p' > /etc/pki/ca-trust/source/anchors/tmp_ca.crt
update-ca-trust
systemctl restart docker
```

Run the following command on Ubuntu nodes:

```
openssl s_client -showcerts -connect {server_url}:{server_port} < /dev/null | sed -ne '/-BEGIN CERTIFICATE-/,/-END CERTIFICATE-/p' > /usr/local/share/ca-certificates/tmp_ca.crt
update-ca-trust
systemctl restart docker
```

----End

## Check Item 6: Whether the Image Size Is Too Large

The pod event list contains the event "Failed to pull image". This may be caused by a large image size.

```
Failed to pull image "XXX": rpc error: code = Unknown desc = context canceled
```

Log in to the node and run the **docker pull** command to manually pull the image. The image is successfully pulled.

### Possible Causes

The default value of **image-pull-progress-deadline** is 1 minute. If the image pull progress is not updated within 1 minute, image pull is canceled. If the node performance is poor or the image size is too large, the image may fail to be pulled and the workload may fail to be started.

### Solution

- (Recommended) Method 1: Log in to the node, run the **docker pull** command to manually pull the image, and check whether **imagePullPolicy** of the workload is **IfNotPresent** (default policy configuration). In this case, the image that has been pulled to the local host is used to create the workload.
- Method 2: Modify the kubelet configuration parameters.

For a cluster of v1.15 or later, run the following command:

```
vi /opt/cloud/cce/kubernetes/kubelet/kubelet
```

For a cluster earlier than v1.15, run the following command:

```
vi /var/paas/kubernetes/kubelet/kubelet
```



### Solution

Push the frequently used image to SWR and then pull the image from SWR.

## 20.6.1.4 What Should I Do If Container Startup Fails?

### Fault Locating

On the details page of a workload, if an event is displayed indicating that the container fails to be started, perform the following steps to locate the fault:

**Step 1** Log in to the node where the abnormal workload is located.

**Step 2** Check the ID of the container where the workload pod exits abnormally.

```
docker ps -a | grep $podName
```

**Step 3** View the logs of the corresponding container.

```
docker logs $containerID
```

Rectify the fault of the workload based on logs.

**Step 4** Check the error logs.

```
cat /var/log/messages | grep $containerID | grep oom
```

Check whether the system OOM is triggered based on the logs.

----End

### Troubleshooting Process

Determine the cause based on the event information, as listed in [Table 20-6](#).

**Table 20-6** Container startup failure

Log or Event	Cause and Solution
The log contains <b>exit(0)</b> .	No process exists in the container. Check whether the container is running properly. <b>Check Item 1: Whether There Are Processes that Keep Running in the Container (Exit Code: 0)</b>
Event information: <b>Liveness probe failed: Get http...</b> The log contains <b>exit(137)</b> .	Health check fails. <b>Check Item 2: Whether Health Check Fails to Be Performed (Exit Code: 137)</b>

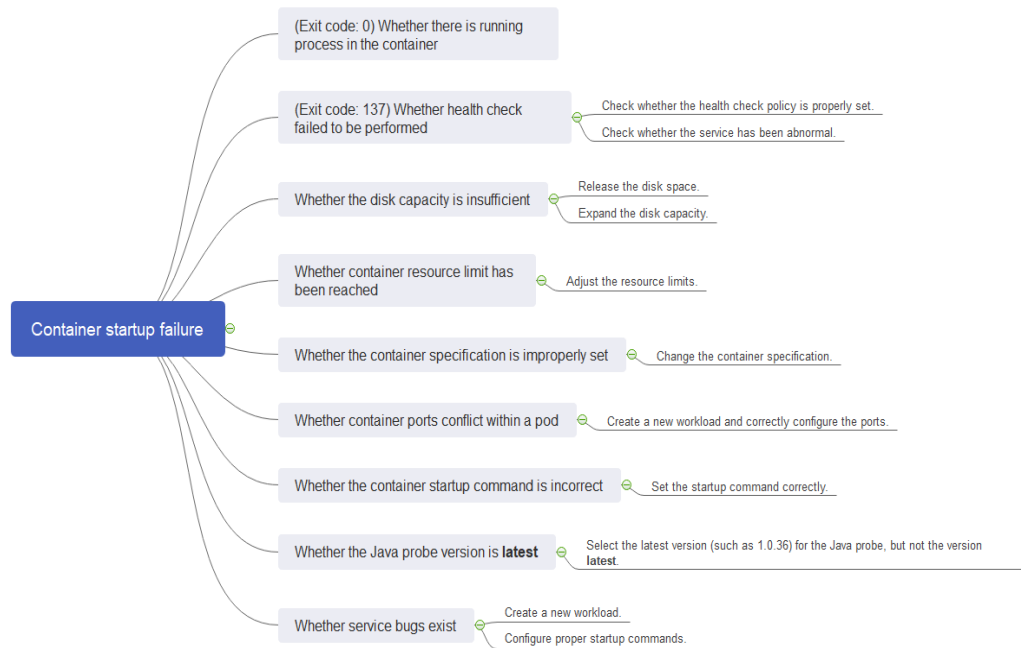
Log or Event	Cause and Solution
<p>Event information: Thin Pool has 15991 free data blocks which are less than minimum required 16383 free data blocks. Create more free space in thin pool or use dm.min_free_space option to change behavior</p>	<p>The disk space is insufficient. Clear the disk space. <b>Check Item 3: Whether the Container Disk Space Is Insufficient</b></p>
<p>The keyword OOM exists in the log.</p>	<p>The memory is insufficient. <b>Check Item 4: Whether the Upper Limit of Container Resources Has Been Reached</b> <b>Check Item 5: Whether the Resource Limits Are Improperly Configured for the Container</b></p>
<p>Address already in use</p>	<p>A conflict occurs between container ports in the pod. <b>Check Item 6: Whether the Container Ports in the Same Pod Conflict with Each Other</b></p>

In addition to the preceding possible causes, there are some other possible causes:

- **Check Item 7: Whether the Container Startup Command Is Correctly Configured**
- **Check Item 8: Whether the User Service Has a Bug**
- Use the correct image when you create a workload on an Arm node.



Figure 20-3 Troubleshooting process of the container restart failure



## Check Item 1: Whether There Are Processes that Keep Running in the Container (Exit Code: 0)

**Step 1** Log in to the node where the abnormal workload is located.

**Step 2** View the container status.

```
docker ps -a | grep $podName
```

Example:

```

[root@xxx ~]# docker ps -a | grep test
1f59a7f4c777 613855f01959 "/bin/bash" 10 seconds ago Exited (0) 10 seconds ago
k8s_container-0_test-66b79cbdb7-htcjf_default_5c388617-ac32-11e9-9168-fa163ec28742_1 2c73ac8717cc cce-pause:2.0 12 seconds ago Up 12 seconds
k8s_POD_test-66b79cbdb7-htcjf_default_5c388617-ac32-11e9-9168-fa163ec28742_0

```

If no running process exists in the container, the status code **Exited (0)** is displayed.

----End

## Check Item 2: Whether Health Check Fails to Be Performed (Exit Code: 137)

The health check configured for a workload is performed on services periodically. If an exception occurs, the pod reports an event and the pod fails to be restarted.

If the liveness-type (workload liveness probe) health check is configured for the workload and the number of health check failures exceeds the threshold, the containers in the pod will be restarted. On the workload details page, if Kubernetes events contain **Liveness probe failed: Get http...**, the health check fails.

**Solution**

Click the workload name to go to the workload details page, click the **Containers** tab. Then select **Health Check** to check whether the policy is proper or whether services are running properly.

### Check Item 3: Whether the Container Disk Space Is Insufficient

The following message refers to the thin pool disk that is allocated from the Docker disk selected during node creation. You can run the **lvs** command as user **root** to view the current disk usage.

Thin Pool has 15991 free data blocks which are less than minimum required 16383 free data blocks. Create more free space in thin pool or use dm.min\_free\_space option to change behavior

```
lvs
LV VG Attr LSize Pool Origin Data% Meta% Move Log Cpy%Sync Convert
dockersys vgpas -wi-ao---- <18.00g
kubernetes vgpas -wi-ao---- <18.00g
thinpool vgpas twi-aot--- 67.00g 98.04 1.32
```

#### Solution

##### Solution 1: Clearing images

Perform the following operations to clear unused images:

- Nodes that use containerd
  - a. Obtain local images on the node.
 

```
crictl images -v
```
  - b. Delete the images that are not required by image ID.
 

```
crictl rmi Image ID
```
- Nodes that use Docker
  - a. Obtain local images on the node.
 

```
docker images
```
  - b. Delete the images that are not required by image ID.
 

```
docker rmi Image ID
```

#### NOTE

Do not delete system images such as the cce-pause image. Otherwise, pods may fail to be created.

##### Solution 2: Expanding the disk capacity

To expand a disk capacity, perform the following steps:

- Step 1** Expand the capacity of the data disk on the EVS console.
- Step 2** Log in to the CCE console and click the cluster. In the navigation pane, choose **Nodes**. Click **More > Sync Server Data** in the row containing the target node.
- Step 3** Log in to the target node.
- Step 4** Run the **lsblk** command to check the block device information of the node.

A data disk is divided depending on the container storage **Rootfs**:

- **Overlayfs**: No independent thin pool is allocated. Image data is stored in the **dockersys** disk.

```
lsblk
NAME MAJ:MIN RM SIZE RO TYPE MOUNTPOINT
sda 8:0 0 50G 0 disk
```

```
└─sda1 8:1 0 50G 0 part /
sdb 8:16 0 200G 0 disk
├─vgpaas-dockersys 253:0 0 90G 0 lvm /var/lib/docker # Space used by the container
engine
└─vgpaas-kubernetes 253:1 0 10G 0 lvm /mnt/paas/kubernetes/kubelet # Space used by
Kubernetes
```

Run the following commands on the node to add the new disk capacity to the **dockersys** disk:

```
pvresize /dev/sdb
lvextend -l+100%FREE -n vgpaas/dockersys
resize2fs /dev/vgpaas/dockersys
```

- **Devicemapper:** A thin pool is allocated to store image data.

```
lsblk
NAME MAJ:MIN RM SIZE RO TYPE MOUNTPOINT
sda 8:0 0 50G 0 disk
├─sda1 8:1 0 50G 0 part /
└─sdb 8:16 0 200G 0 disk
 ├─vgpaas-dockersys 253:0 0 18G 0 lvm /var/lib/docker
 ├─vgpaas-thinpool_tmeta 253:1 0 3G 0 lvm
 │ └─vgpaas-thinpool 253:3 0 67G 0 lvm # Thin pool space.
 │ ...
 └─vgpaas-thinpool_tdata 253:2 0 67G 0 lvm
 └─vgpaas-thinpool 253:3 0 67G 0 lvm
 ...
 └─vgpaas-kubernetes 253:4 0 10G 0 lvm /mnt/paas/kubernetes/kubelet
```

- Run the following commands on the node to add the new disk capacity to the **thinpool** disk:

```
pvresize /dev/sdb
lvextend -l+100%FREE -n vgpaas/thinpool
```

- Run the following commands on the node to add the new disk capacity to the **dockersys** disk:

```
pvresize /dev/sdb
lvextend -l+100%FREE -n vgpaas/dockersys
resize2fs /dev/vgpaas/dockersys
```

----End

## Check Item 4: Whether the Upper Limit of Container Resources Has Been Reached

If the upper limit of container resources has been reached, OOM will be displayed in the event details as well as in the log:

```
cat /var/log/messages | grep 96feb0a425d6 | grep oom
```

```
[root@xxx ~]#
[root@xxx ~]# cat /var/log/messages | grep 96feb0a425d6 | grep oom
2019-07-22T11:57:49.441756+08:00 xxx dockerd: time="2019-07-22T11:57:49.440755329+08:00" level=info msg=event OOMKilled=true containerID=96feb0a425d6669f8f062cf3a6096868617a10711334fd5bce4a6ee6eadc82d module=libcontainerd namespace=moby topic=/tasks/oom
2019-07-22T11:59:55.028162+08:00 xxx [/bin/bash]: [2019-07-22T11:57:49.441756+08:00 xxx dockerd: time="2019-07-22T11:57:49.440755329+08:00" level=info msg=event OOMKilled=true containerID=96feb0a425d6669f8f062cf3a6096868617a10711334fd5bce4a6ee6eadc82d module=libcontainerd namespace=moby topic=/tasks/oom] return code=[127], execute failed by [root(uid=0)] from [pts/0 (192.168.0.7)]
2019-07-22T12:01:47.621029+08:00 xxx [/bin/bash]: [cat /var/log/messages | grep 96feb0a425d6 | grep oom] return code=[0], execute success by [root(uid=0)] from [pts/0 (192.168.0.7)]
[root@xxx ~]#
```

When a workload is created, if the requested resources exceed the configured upper limit, the system OOM is triggered and the container exits unexpectedly.

## Check Item 5: Whether the Resource Limits Are Improperly Configured for the Container

If the resource limits set for the container during workload creation are less than required, the container fails to be restarted.

## Check Item 6: Whether the Container Ports in the Same Pod Conflict with Each Other

- Step 1** Log in to the node where the abnormal workload is located.
- Step 2** Check the ID of the container where the workload pod exits abnormally.

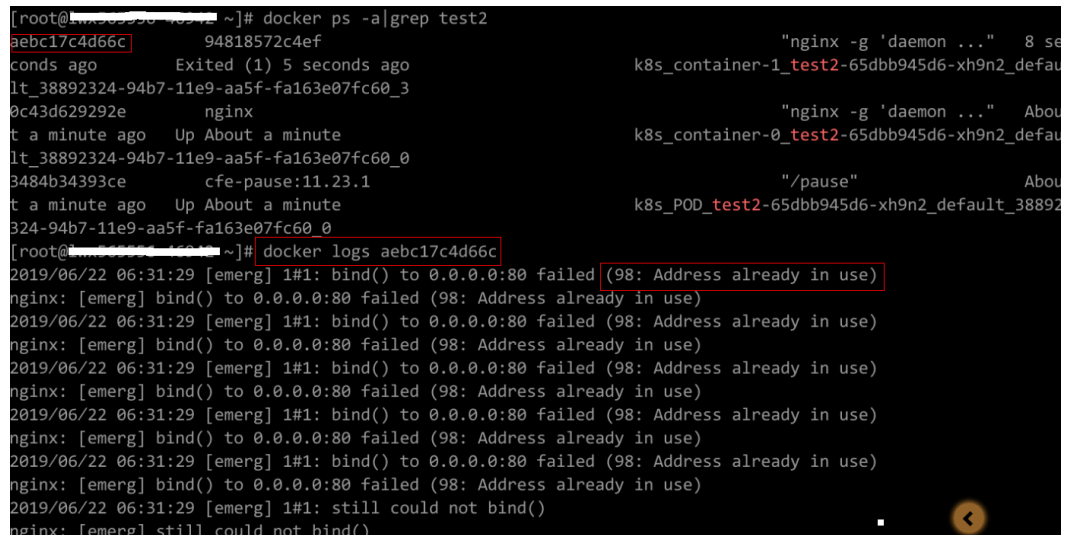
```
docker ps -a | grep $podName
```

- Step 3** View the logs of the corresponding container.

```
docker logs $containerID
```

Rectify the fault of the workload based on logs. As shown in the following figure, container ports in the same pod conflict. As a result, the container fails to be started.

Figure 20-4 Container restart failure due to a container port conflict



----End

### Solution

Re-create the workload and set a port number that is not used by any other pod.

## Check Item 7: Whether the Container Startup Command Is Correctly Configured

The error messages are as follows:



### Solution

Click the workload name to go to the workload details page, click the **Containers** tab. Choose **Lifecycle** , click **Startup Command**, and ensure that the command is correct.

### Check Item 8: Whether the User Service Has a Bug

Check whether the workload startup command is correctly executed or whether the workload has a bug.

**Step 1** Log in to the node where the abnormal workload is located.

**Step 2** Check the ID of the container where the workload pod exits abnormally.

```
docker ps -a | grep $podName
```

**Step 3** View the logs of the corresponding container.

```
docker logs $containerID
```

Note: In the preceding command, *containerID* indicates the ID of the container that has exited.

**Figure 20-5** Incorrect startup command of the container

```
[root@dcb-ha-11638 ~]# docker ps -a | grep nginx
cf0357f617f9 3f8a4339aadd "/bin/bash /tmp/test." 2 minutes ago
 Exited (127) 2 minutes ago k8s_container-0_nginx-267
0177225-kt929_test_d6402ef7-4e0f-11e8-b4f7-fa163e74044e_5
c2176ce394a1 cfe-pause:3.7.6 "/pause" 5 minutes ago
 Up 5 minutes k8s_POD_nginx-2670177225-
kt929_test_d6402ef7-4e0f-11e8-b4f7-fa163e74044e_0
[root@dcb-ha-11638 ~]# docker logs cf035
/bin/bash: /tmp/test.sh: No such file or directory
[root@dcb-ha-11638 ~]#
```

As shown in the figure above, the container fails to be started due to an incorrect startup command. For other errors, rectify the bugs based on the logs.

----End

#### Solution

Create a new workload and configure a correct startup command.

## 20.6.1.5 What Should I Do If a Pod Fails to Be Evicted?

### What Is Eviction

When an exception occurs on a node, Kubernetes evicts the pods on the node to ensure the workload availability.

In Kubernetes, both kube-controller-manager and kubelet can evict pods.

- **Eviction implemented by kube-controller-manager**

kube-controller-manager consists of multiple controllers, and eviction is implemented by the node controller. The controller periodically checks the status of all nodes. When a node is in the **NotReady** state for a period of time, all pods on the node are evicted.

kube-controller-manager provides the following startup parameters to control evictions:

- **pod-eviction-timeout**: an interval when a node is down, after which pods on that node are evicted. The default interval is 5 minutes.
  - **node-eviction-rate**: a rate at which nodes are evicted, which is implemented by the token bucket traffic control algorithm. The default value is **0.1**, indicating that 0.1 nodes are evicted per second. Note that this rate is not the rate at which pods are evicted, but the rate at which nodes are evicted. That is, one node is cleared every 10 seconds.
  - **secondary-node-eviction-rate**: secondary eviction rate. When a large number of nodes are down in the cluster, the eviction rate decreases. The default value is **0.01**.
  - **unhealthy-zone-threshold**: threshold for a zone to be considered unhealthy. This parameter determines when to enable the secondary eviction rate. The default value is **0.55**. That is, if the percentage of down nodes in a zone exceeds 55%, the zone is unhealthy.
  - **large-cluster-size-threshold**: threshold for a cluster to be considered large. When the number of nodes in a zone exceeds this threshold, the zone is considered as a large cluster. If the percentage of down nodes in a large cluster exceeds 55%, the eviction rate is reduced to 0.01. If the cluster is a small one, the eviction rate is reduced to 0.
- **Eviction implemented by kubelet**

If resources of a node are to be used up, kubelet executes the eviction policy based on the pod priority, resource usage, and resource request. If pods have the same priority, the pod that uses the most resources or requests for the most resources will be evicted first.

kube-controller-manager evicts all pods on a node, whereas kubelet evicts certain pods on a node. Pods to be evicted are determined by the pod QoS. kubelet periodically checks the memory and disk resources of the node. If the resources are insufficient, pods are evicted based on the priority.

There are soft eviction thresholds and hard eviction thresholds.

- **Soft eviction threshold**: A grace period is set for node resources. kubelet will reclaim node resources associated with this threshold if that grace period is exceeded. If the node resource usage reaches this threshold but falls below it before the grace period is exceeded, kubelet will not evict pods on the node.
- **Hard eviction threshold**: Pods are immediately evicted once this threshold is reached.

kubelet provides the following parameters to control evictions:

- **eviction-soft**: soft eviction thresholds. If, for example, **memory.available** is less than 1.5 GiB, pods will be evicted only after the grace period specified by **eviction-soft-grace-period** is exceeded.
- **eviction-soft-grace-period**: an eviction grace period. It is used to terminate the grace period of a pod to be evicted by **eviction-soft**. The default value is 90 seconds.
- **eviction-max-pod-grace-period**: the maximum allowed grace period to use when terminating pods in response to a soft eviction threshold being met
- **eviction-pressure-transition-period**: a duration for which kubelet has to wait before transitioning out of an eviction pressure condition. The default value is 5 minutes. If the time exceeds the threshold, the node is

set to **memory pressure** or **disk pressure**, and then pod eviction is started.

- **eviction-minimum-reclaim**: the minimum number of resources that must be reclaimed in each eviction
- **eviction-hard**: hard eviction thresholds. If, for example, **memory.available** is less than 1 GiB, the pod will be evicted.

## Fault Locating

If the pods are not evicted when the node is faulty, perform the following steps to locate the fault:

After the following command is run, the command output shows that many pods are in the **Evicted** state.

```
kubectl get pods
```

Check results will be recorded in kubelet logs of the node. You can run the following command to search for the information:

```
cat /var/paas/sys/log/kubernetes/kubelet.log | grep -i Evicted -C3
```

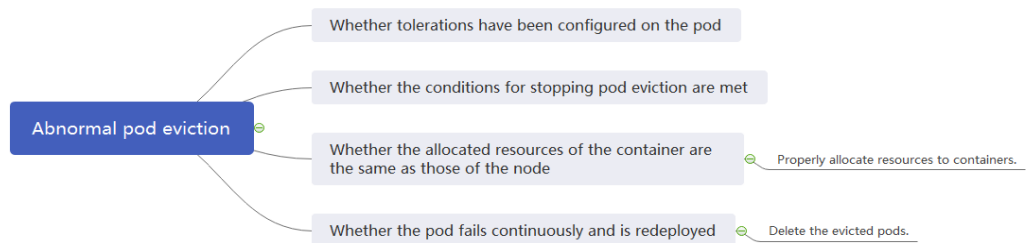
## Troubleshooting Process

Troubleshooting methods are sorted based on the occurrence probability of the possible causes. You are advised to check the possible causes from high probability to low probability to quickly locate the cause of the problem.

If the fault persists after a possible cause is rectified, check other possible causes.

- **Check Item 1: Whether Tolerations Have Been Configured on the Pod**
- **Check Item 2: Whether the Conditions for Stopping Pod Eviction Are Met**
- **Check Item 3: Whether the Allocated Resources of the Container Are the Same as Those of the Node**
- **Check Item 4: Whether the Pod Fails Continuously and Is Redeployed**

Figure 20-6 Troubleshooting process for pod eviction exception



### Check Item 1: Whether Tolerations Have Been Configured on the Pod

Use `kubectl`, or locate the row containing the target workload and choose **More > Edit YAML** in the **Operation** column to check whether tolerance is configured for the workload. For details, see [Taints and Tolerations](#).

## Check Item 2: Whether the Conditions for Stopping Pod Eviction Are Met

If the number of nodes in a cluster is smaller than 50 and the number of faulty nodes accounts for over 55% of the total nodes, the pod eviction will be suspended. In this case, Kubernetes will not attempt to evict the workload on the faulty node. For details, see [Rate limits on eviction](#).

## Check Item 3: Whether the Allocated Resources of the Container Are the Same as Those of the Node

An evicted container is frequently scheduled to the original node.

### Possible Causes

A node evicts a container based on the node resource usage. The evicted container is scheduled based on the allocated node resources. Eviction and scheduling are based on different rules. Therefore, an evicted container may be scheduled to the original node again.

### Solution

Properly allocate resources to each container.

## Check Item 4: Whether the Pod Fails Continuously and Is Redeployed

A pod in the workload fails and is being redeployed constantly.

### Analysis

After a pod is evicted and scheduled to a new node, if pods in that node are also being evicted, the pod will be evicted again. Pods may be evicted repeatedly.

If a pod is evicted by kube-controller-manager, it would be in the **Terminating** state. This pod will be automatically deleted only after the node where the container is located is restored. If the node has been deleted or cannot be restored due to other reasons, you can forcibly delete the pod.

If a pod is evicted by kubelet, it would be in the **Evicted** state. This pod is only used for subsequent fault locating and can be directly deleted.

### Solution

Run the following command to delete the evicted pods:

```
kubectl get pods <namespace> | grep Evicted | awk '{print $1}' | xargs kubectl delete pod <namespace>
```

In the preceding command, *<namespace>* indicates the namespace name. Configure it based on your requirements.

## References

[Kubelet does not delete evicted pods](#)



## 20.6.1.6 What Should I Do If a Storage Volume Cannot Be Mounted or the Mounting Times Out?

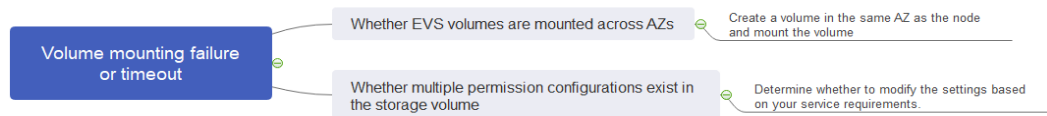
### Troubleshooting Process

Troubleshooting methods are sorted based on the occurrence probability of the possible causes. You are advised to check the possible causes from high probability to low probability to quickly locate the cause of the problem.

If the fault persists after a possible cause is rectified, check other possible causes.

- [Check Item 1: Whether EVS Volumes Are Mounted Across AZs](#)
- [Check Item 2: Whether Multiple Permission Configurations Exist in the Storage Volume](#)
- [Check Item 3: Whether There Is More Than One Replica for a Deployment with EVS Volumes](#)
- [Check Item 4: Whether the EVS Disk File System Is Damaged](#)

**Figure 20-7** Troubleshooting for storage volume mounting failure or mounting timeout



### Check Item 1: Whether EVS Volumes Are Mounted Across AZs

#### Symptom

Mounting an EVS volume to a StatefulSet times out.

#### Fault Locating

If your node is in **AZ 1** but the volume to be mounted is in **AZ 2**, the mounting times out and the volume cannot be mounted.

#### Solution

Create a volume in the same AZ as the node and mount the volume.

### Check Item 2: Whether Multiple Permission Configurations Exist in the Storage Volume

If the volume to be mounted stores too many data and involves permission-related configurations, the file permissions need to be modified one by one, which results in mounting timeout.

#### Fault Locating

- Check whether the **securityContext** field contains **runAsuser** and **fsGroup**. **securityContext** is a Kubernetes field that defines the permission and access control settings of pods or containers.
- Check whether the startup commands contain commands used to obtain or modify file permissions, such as **ls**, **chmod**, and **chown**.

### Solution

Determine whether to modify the settings based on your service requirements.

## Check Item 3: Whether There Is More Than One Replica for a Deployment with EVS Volumes

### Symptom

The pod fails to be created, and an event indicating that the storage fails to be added is reported.

```
Multi-Attach error for volume "pvc-62a7a7d9-9dc8-42a2-8366-0f5ef9db5b60" Volume is already used by pod(s) testttt-7b774658cb-lc98h
```

### Fault Locating

Check whether the number of replicas of the Deployment is greater than 1.

If the Deployment uses an EVS volume, the number of replicas can only be 1. If you specify more than two pods for the Deployment on the backend, CCE does not restrict the creation of the Deployment. However, if these pods are scheduled to different nodes, some pods cannot be started because the EVS volumes used by the pods cannot be mounted to the nodes.

### Solution

Set the number of replicas of the Deployment that uses an EVS volume to 1 or use other volume types.

## Check Item 4: Whether the EVS Disk File System Is Damaged

### Symptom

The pod fails to be created, and information similar to the following is displayed, indicating that the disk file system is damaged.

```
MountVolume.MountDevice failed for volume "pvc-08178474-c58c-4820-a828-14437d46ba6f" : rpc error: code = Internal desc = [09060def-afd0-11ec-9664-fa163eef47d0] /dev/sda has file system, but it is detected to be damaged
```

### Solution

Back up the disk in EVS and run the following command to restore the file system:

```
fsck -y {Drive letter}
```

## 20.6.1.7 What Should I Do If a Workload Remains in the Creating State?

### Symptom

The workload remains in the creating state.

### Troubleshooting Process

Troubleshooting methods are sorted based on the occurrence probability of the possible causes. You are advised to check the possible causes from high probability to low probability to quickly locate the cause of the problem.

If the fault persists after a possible cause is rectified, check other possible causes.

- [Check Item 1: Whether the cce-pause Image Is Deleted by Mistake](#)
- [Check Item 2: Modifying Node Specifications After the CPU Management Policy Is Enabled in the Cluster](#)

## Check Item 1: Whether the cce-pause Image Is Deleted by Mistake

### Symptom

When creating a workload, an error message indicating that the sandbox cannot be created is displayed. This is because the **cce-pause:3.1** image fails to be pulled.

```
Failed to create pod sandbox: rpc error: code = Unknown desc = failed to get sandbox image "cce-pause:3.1": failed to pull image "cce-pause:3.1": failed to pull and unpack image "docker.io/library/cce-pause:3.1": failed to resolve reference "docker.io/library/cce-pause:3.1": pulling from host **** failed with status code [manifests 3.1]: 400 Bad Request
```

### Possible Causes

The image is a system image added during node creation. If the image is deleted by mistake, the workload cannot be created.

### Solution

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

**Step 2** Decompress the cce-pause image installation package.

```
tar -xzf /opt/cloud/cce/package/node-package/pause-*.tgz
```

**Step 3** Import the image.

- For a node which uses a Docker container runtime:  

```
docker load ./pause/package/image/cce-pause-3.1.tar
```
- For a node which uses a containerd container runtime:  

```
ctr -n k8s.io image import ./pause/package/image/cce-pause-3.1.tar
```

**Step 4** Create a workload.

----End

## Check Item 2: Modifying Node Specifications After the CPU Management Policy Is Enabled in the Cluster

The kubelet option **cpu-manager-policy** defaults to **static**. This allows granting enhanced CPU affinity and exclusivity to pods with certain resource characteristics on the node. If you modify CCE node specifications on the ECS console, the original CPU information does not match the new CPU information. As a result, workloads on the node cannot be restarted or created.

**Step 1** Log in to the CCE node (ECS) and delete the **cpu\_manager\_state** file.

Example command for deleting the file:

```
rm -rf /mnt/paas/kubernetes/kubelet/cpu_manager_state
```

**Step 2** Restart the node or kubelet. The following is the kubelet restart command:

```
systemctl restart kubelet
```

Verify that workloads on the node can be successfully restarted or created.

For details, see [What Should I Do If I Fail to Restart or Create Workloads on a Node After Modifying the Node Specifications?](#).

----End

### 20.6.1.8 What Should I Do If Pods in the Terminating State Cannot Be Deleted?

#### Symptom

When a node is in the Unavailable state, CCE migrates container pods on the node and sets the pods running on the node to the **Terminating** state.

After the node is restored, the pods in the **Terminating** state are automatically deleted.

However, some pods remain in the **Terminating** state.

```
#kubectl get pod -n aos
NAME READY STATUS RESTARTS AGE
aos-apiserver-5f8f5b5585-s9l92 1/1 Terminating 0 3d1h
aos-cmdbserver-789bf5b497-6rwrq 1/1 Running 0 3d1h
aos-controller-545d78bs8d-vm6j9 1/1 Running 3 3d1h
```

Running **kubectl delete pods <podname> -n <namespace>** cannot delete the pods.

```
kubectl delete pods aos-apiserver-5f8f5b5585-s9l92 -n aos
```

#### Solution

You can run the following command to forcibly delete the pods created in any ways:

```
kubectl delete pods <pod> --grace-period=0 --force
```

Therefore, run the following command to delete the pod:

```
kubectl delete pods aos-apiserver-5f8f5b5585-s9l92 --grace-period=0 --force
```

### 20.6.1.9 What Should I Do If a Workload Is Stopped Caused by Pod Deletion?

#### Problem

A workload is in **Stopped** state.

#### Cause:

The **metadata.enable** field in the YAML file of the workload is **false**. As a result, the pod of the workload is deleted and the workload is in the stopped status.

```
kind: Deployment
apiVersion: apps/v1
metadata:
 name: test
 namespace: default
 selfLink: /apis/apps/v1/namespaces/default/deployments/test
 uid: b130db9f-9306-11e9-a2a9-fa163eaff9f7
 resourceVersion: '7314771'
 generation: 1
 creationTimestamp: '2019-06-20T02:54:16Z'
 labels:
 appgroup: ''
 annotations:
 deployment.kubernetes.io/revision: '1'
 description: ''
 enable: false
spec:
```

## Solution

Delete the **enable** field or set it to **true**.

### 20.6.1.10 What Should I Do If an Error Occurs When Deploying a Service on the GPU Node?

#### Symptom

The following exceptions occur when services are deployed on the GPU nodes in a CCE cluster:

1. The GPU memory of containers cannot be queried.
2. Seven GPU services are deployed, but only two of them can be accessed properly. Errors are reported during the startup of the remaining five services.
  - The CUDA versions of the two services that can be accessed properly are 10.1 and 10.0, respectively.
  - The CUDA versions of the failing services are also 10.0 and 10.1.
3. Files named **core.\*** are found in the GPU service containers. No such files existed in any of the previous deployments.

#### Fault Locating

1. The driver version of the gpu add-on is too old. After a new driver is downloaded and installed, the fault is rectified.
2. The workloads do not declare that GPU resources are required.

#### Suggested Solution

After you install `gpu-beta` (`gpu-device-plugin`) on a node, `nvidia-smi` will be automatically installed. If an error is reported during GPU deployment, this issue is

typically caused by an NVIDIA driver installation failure. Check whether the NVIDIA driver has been downloaded.

- 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.

If the driver address is incorrect, uninstall the add-on, reinstall it, and configure the correct address.

 NOTE

You are advised to store the NVIDIA driver in the OBS bucket and set the bucket policy to public read.

## Helpful Links

- [How Do I Rectify Failures When the NVIDIA Driver Is Used to Start Containers on GPU Nodes?](#)

### 20.6.1.11 What Should I Do If Sandbox-Related Errors Are Reported When the Pod Remains in the Creating State?

#### Symptom

The pod remains in the creating state for a long time, and the sandbox-related errors are reported.

Failed create pod sandbox: rpc error: code = Unknown desc = [ failed to setup sandbox .....

#### Solution

Select a troubleshooting method for your cluster:

##### Clusters of V1.13

 NOTE

This method applies only to clusters of v1.13.

1. Sandbox errors are generally caused by the abnormal startup of container components on the node. You can run the **systemctl status canal** command to check the container components and run the **systemctl restart canal** command to restart the abnormal component.
2. All container components on the node are normal, but the **loopback** file of the CNI is missing. The error is as follows: **network: failed to find plugin "loopback" in path [/opt/cni/bin]**. You can copy a complete version of the **loopback** file from the current region or other regions which share the same

cluster versions (minor releases can be ignored), and put the **loopback** file in the path **/opt/cni/bin/**. Then, restart the canal component.

### Clusters earlier than V1.13

1. Sandbox errors are generally caused by the abnormal startup of container components on the node. You can run the **su paas -c '/var/paas/monit/bin/monit summary'** command to check the container components and run the **su paas -c '/var/paas/monit/bin/monit restart canal'** command to restart the abnormal component.
2. All container components on the node are normal, but the **loopback** file of the CNI is missing. The error is as follows: **network: failed to find plugin "loopback" in path [/opt/cni/bin]**. You can copy a complete version of the **loopback** file from the current region or other regions which share the same cluster versions (minor releases can be ignored), and put the **loopback** file in the path **/opt/cni/bin/**. Then, restart the canal component.

## 20.6.2 Container Configuration

### 20.6.2.1 When Is Pre-stop Processing Used?

Service processing takes a long time. Pre-stop processing makes sure that during an upgrade, a pod is killed only when the service in the pod has been processed.

### 20.6.2.2 How Do I Set an FQDN for Accessing a Specified Container in the Same Namespace?

#### Context

When creating a workload, users can specify a container, pod, and namespace as an FQDN for accessing the container in the same namespace.

FQDN stands for Fully Qualified Domain Name, which contains both the host name and domain name. These two names are combined using a period (.).

For example, if the host name is **bigserver** and the domain name is **mycompany.com**, the FQDN is **bigserver.mycompany.com**.

#### Solution

Solution 1: Use the domain name for service discovery. The host name and namespace must be pre-configured. The domain name of the registered service is in the format of *service name.namespace name.svc.cluster.local*. The limitation of this solution is that the registration center must be deployed using containers.

Solution 2: Use the host network to deploy containers and then configure affinity between the containers and a node in the cluster. In this way, the service address (that is, the node address) of the containers can be determined. The registered address is the IP address of the node where the service is located. This solution allows you to deploy the registration center using VMs, whereas the disadvantage is that the host network is not as efficient as the container network.

### 20.6.2.3 What Should I Do If Health Check Probes Occasionally Fail?

When the liveness and readiness probes fail to perform the health check, locate the service fault first.

Common causes are as follows:

- The service processing takes a long time. As a result, the response times out.
- The Tomcat connection setup and waiting time are too long (for example, too many connections or threads). As a result, the response times out.
- The performance of the node where the container is located, such as the disk I/O, reaches the bottleneck. As a result, the service processing times out.

### 20.6.2.4 How Do I Set the umask Value for a Container?

#### Symptom

A container is started in **tailf /dev/null** mode and the directory permission is **700** after the startup script is manually executed. If the container is started by Kubernetes itself without **tailf**, the obtained directory permission is **751**.

#### Solution

The reason is that the umask values set in the preceding two startup modes are different. Therefore, the permissions on the created directories are different.

The umask value is used to set the default permission for a newly created file or directory. If the umask value is too small, group users or other users will have excessive permissions, posing security threats to the system. Therefore, the default umask value for all users is set to **0077**. That is, the default permission on directories created by users is **700**, and the default permission on files is **600**.

You can add the following content to the startup script to set the permission on the created directory to **700**:

1. Add **umask 0077** to the **/etc/bashrc** file and all files in **/etc/profile.d/**.
2. Run the following command:  

```
echo "umask 0077" >> $FILE
```

#### NOTE

*FILE* indicates the file name, for example, **echo "umask 0077" >> /etc/bashrc**.

3. Set the owner and group of the **/etc/bashrc** file and all files in **/etc/profile.d/** to **root**.
4. Run the following command:  

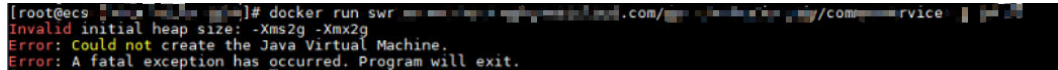
```
chown root.root $FILE
```



### 20.6.2.5 What Can I Do If an Error Is Reported When a Deployed Container Is Started After the JVM Startup Heap Memory Parameter Is Specified for ENTRYPOINT in Dockerfile?

#### Symptom

After the JVM startup heap memory parameter is specified for ENTRYPOINT in the Dockerfile, an error message "invalid initial heap size" is displayed during the deployed container's startup, as shown in the following figure:



```
[root@ecs ~]# docker run swr.../com...ervice
Invalid initial heap size: -Xms2g -Xmx2g
Error: Could not create the Java Virtual Machine.
Error: A fatal exception has occurred. Program will exit.
```

#### Answer

Check the ENTRYPOINT settings. The following settings are incorrect:

```
ENTRYPOINT ["java","-Xms2g -Xmx2g","-jar","xxx.jar"]
```

You can use either of the following methods to solve the problem:

- **(Recommended)** Write the container startup command in **Workloads > Container Settings > Lifecycle > Startup Command**, then the container can be started properly.
- Change the format of the **ENTRYPOINT** startup command to the following:  

```
ENTRYPOINT exec java -Xmx2g -Xms2g -jar xxx.jar
```

### 20.6.2.6 What Is the Retry Mechanism When CCE Fails to Start a Pod?

CCE is a fully managed Kubernetes service and is fully compatible with Kubernetes APIs and kubectl.

In Kubernetes, the spec of a pod contains a **restartPolicy** field. The value of **restartPolicy** can be **Always**, **OnFailure**, or **Never**. The default value is **Always**.

- **Always**: When a container fails, kubelet automatically restarts the container.
- **OnFailure**: When a container stops running and the exit code is not **0** (indicating normal exit), kubelet automatically restarts the container.
- **Never**: kubelet does not restart the container regardless of the container running status.

**restartPolicy** applies to all containers in a pod.

**restartPolicy** only refers to restarts of the containers by kubelet on the same node. When containers in a pod exit, kubelet restarts them with an exponential back-off delay (10s, 20s, 40s, ...), which is capped at five minutes. Once a container has been running for 10 minutes without any problems, kubelet resets the restart backoff timer for the container.

The settings of **restartPolicy** vary depending on the controller:

- **Replication Controller (RC)** and **DaemonSet**: **restartPolicy** must be set to **Always** to ensure continuous running of the containers.
- **Job**: **restartPolicy** must be set to **OnFailure** or **Never** to ensure that containers are not restarted after being executed.

## 20.6.3 Scheduling Policies

### 20.6.3.1 How Do I Evenly Distribute Multiple Pods to Each Node?

The kube-scheduler component in Kubernetes is responsible pod scheduling. For each newly created pod or other unscheduled pods, kube-scheduler selects an optimal node from them to run on. kube-scheduler selects a node for a pod in a 2-step operation: filtering and scoring. In the filtering step, all nodes where it is feasible to schedule the pod are filtered out. In the scoring step, kube-scheduler ranks the remaining nodes to choose the most suitable pod placement. Finally, kube-scheduler schedules the pod to the node with the highest score. If there is more than one node with the equal scores, kube-scheduler selects one of them at random.

BalancedResourceAllocation is only one of the scoring priorities. Other scoring items may also cause uneven distribution. For details about scheduling, see [Kubernetes Scheduler](#) and [Scheduling Policies](#).

You can configure pod anti-affinity policies to evenly distribute pods onto different nodes.

Example:

```
kind: Deployment
apiVersion: apps/v1
metadata:
 name: nginx
 namespace: default
spec:
 replicas: 2
 selector:
 matchLabels:
 app: nginx
 template:
 metadata:
 labels:
 app: nginx
 spec:
 containers:
 - name: container-0
 image: nginx:alpine
 resources:
 limits:
 cpu: 250m
 memory: 512Mi
 requests:
 cpu: 250m
 memory: 512Mi
 affinity:
 podAntiAffinity:
 preferredDuringSchedulingIgnoredDuringExecution:
 - podAffinityTerm:
 labelSelector:
 matchExpressions:
 - key: app
 operator: In
 values:
 - nginx
 namespaces:
 - default
 topologyKey: kubernetes.io/hostname # It takes effect on the node.
workload.
 # Workload anti-affinity
 # Ensure that the following conditions are met:
 # Select the label of the pod, which is anti-affinity with the
```

## 20.6.3.2 How Do I Prevent a Container on a Node from Being Evicted?

### Context

During workload scheduling, two containers on a node may compete for resources. As a result, kubelet evicts both containers. This section describes how to set a policy to retain one of the containers.

### Solution

kubelet uses the following criteria to evict a pod:

- Quality of Service (QoS) class: **BestEffort**, **Burstable**, and **Guaranteed**
- Consumed resources based on the pod scheduling request

Pods of different QoS classes are evicted in the following sequence:

BestEffort -> Burstable -> Guaranteed

- BestEffort pods: These pods have the lowest priority. They will be the first to be killed if the system runs out of memory.
- Burstable pods: These pods will be killed if the system runs out of memory and no BestEffort pods exist.
- Guaranteed pods: These pods will be killed if the system runs out of memory and no Burstable or BestEffort pods exist.

#### NOTE

- If processes in a pod are killed because of excessive resource usage (while the node resources are still sufficient), the system tends to restart the container or create a pod.
- If resources are sufficient, you can assign the QoS class of Guaranteed to all pods. In this way, more compute resources are used to improve service performance and stability, reducing troubleshooting time and costs.
- To improve resource utilization, assign the QoS class of Guaranteed to service pods and Burstable or BestEffort to other pods (for example, filebeat).

## 20.6.3.3 Why Are Pods Not Evenly Distributed to Nodes?

The kube-scheduler component in Kubernetes is responsible for pod scheduling. For each newly created pod or other unscheduled pods, kube-scheduler selects an optimal node from them to run on. kube-scheduler selects a node for a pod in a 2-step operation: filtering and scoring. In the filtering step, all nodes where it is feasible to schedule the pod are filtered out. In the scoring step, kube-scheduler ranks the remaining nodes to choose the most suitable pod placement. Finally, kube-scheduler schedules the pod to the node with the highest score. If there is more than one node with the equal scores, kube-scheduler selects one of them at random.

BalancedResourceAllocation is only one of the scoring priorities. Other scoring items may also cause uneven distribution. For details about scheduling, see [Kubernetes Scheduler](#) and [Scheduling Policies](#).

## 20.6.3.4 How Do I Evict All Pods on a Node?

You can run the **kubectl drain** command to safely evict all pods from a node.

 **NOTE**

By default, the **kubectl drain** command retains some system pods, for example, everest-csi-driver.

**Step 1** Use **kubectl** to connect to the cluster.

**Step 2** Check the nodes in the cluster.

```
kubectl get node
```

**Step 3** Select a node and view all pods on the node.

```
kubectl get pod --all-namespaces -owide --field-selector spec.nodeName=192.168.0.160
```

The pods on the node before eviction are as follows:

NAMESPACE	NAME	READY	STATUS	RESTARTS	AGE	IP
NODE	NOMINATED NODE	READINESS GATES				
default	nginx-5bcc57c74b-lgcvh	1/1	Running	0	7m25s	10.0.0.140
192.168.0.160	<none>	<none>				
kube-system	coredns-6fcd88c4c-97p6s	1/1	Running	0	3h16m	10.0.0.138
192.168.0.160	<none>	<none>				
kube-system	everest-csi-controller-56796f47cc-99dtm	1/1	Running	0	3h16m	10.0.0.139
192.168.0.160	<none>	<none>				
kube-system	everest-csi-driver-dpfzl	2/2	Running	2	12d	192.168.0.160
192.168.0.160	<none>	<none>				
kube-system	icagent-tpfpv	1/1	Running	1	12d	192.168.0.160
192.168.0.160	<none>	<none>				

**Step 4** Evict all pods on the node.

```
kubectl drain 192.168.0.160
```

If a pod mounted with local storage or controlled by a DaemonSet set exists on the node, the message "error: unable to drain node "192.168.0.160", aborting command..." will be displayed. The eviction command does not take effect. You can add the following parameters to the end of the preceding command to forcibly evict the pod:

- **--delete-emptydir-data**: forcibly evicts pods mounted with local storage, for example, coredns.
- **--ignore-daemonsets**: forcibly evicts the DaemonSet pods, for example, everest-csi-driver.

In the example, both types of pods exist on the node. Therefore, the eviction command is as follows:

```
kubectl drain 192.168.0.160 --delete-emptydir-data --ignore-daemonsets
```

**Step 5** After the eviction, the node is automatically marked as unschedulable. That is, the node is tainted **node.kubernetes.io/unschedulable = : NoSchedule**.

After the eviction, only system pods are retained on the node.

NAMESPACE	NAME	READY	STATUS	RESTARTS	AGE	IP	NODE
NOMINATED NODE	READINESS GATES						
kube-system	everest-csi-driver-dpfzl	2/2	Running	2	12d	192.168.0.160	192.168.0.160
<none>	<none>						
kube-system	icagent-tpfpv	1/1	Running	1	12d	192.168.0.160	192.168.0.160
<none>	<none>						

----End

## Related Operations

Drain, cordon, and uncordon operations of kubectl:

- **drain**: Safely evicts all pods from a node and marks the node as unschedulable.
- **cordon**: Marks the node as unschedulable. That is, the node is tainted `node.kubernetes.io/unschedulable = : NoSchedule`.
- **uncordon**: Marks the node as schedulable.

For more information, see the [kubectl documentation](#).

## 20.6.4 Others

### 20.6.4.1 What Should I Do If a Scheduled Task Cannot Be Restarted After Being Stopped for a Period of Time?

If a scheduled task is stopped during running, before its restart, the system calculates the difference between the last time the task was successfully executed and the current time and compares the time difference with the scheduled task period multiplied by 100. If the time difference is greater than the period multiplied by 100, the scheduled task will not be triggered again. For details, see [CronJob Limitations](#).

For example, assume that a cron job is set to create a job every minute from 08:30:00 and the **startingDeadlineSeconds** field is not set. If the cron job controller stops running from 08:29:00 to 10:21:00, the job will not be started because the time difference between 08:29:00 and 10:21:00.00 exceeds 100 minutes, that is, the number of missed scheduling times exceeds 100 (in the example, a scheduling period is 1 minute).

If the **startingDeadlineSeconds** field is set, the controller calculates the number of missed jobs in the last  $x$  seconds ( $x$  indicates the value of **startingDeadlineSeconds**). For example, if **startingDeadlineSeconds** is set to **200**, the controller counts the number of jobs missed in the last 200 seconds. In this case, if the cron job controller stops running from 08:29:00 to 10:21:00, the job will start again at 10:22:00, because only three scheduling requests are missed in the last 200 seconds (in the example, one scheduling period is 1 minute).

## Solution

Configure the **startingDeadlineSeconds** parameter in a cron job. This parameter can be created or modified only by using kubectl or APIs.

Example YAML:

```
apiVersion: batch/v1
kind: CronJob
metadata:
 name: hello
spec:
 startingDeadlineSeconds: 200
 schedule: "* * * * *"
 jobTemplate:
 spec:
 template:
```

```
spec:
 containers:
 - name: hello
 image: busybox:1.28
 imagePullPolicy: IfNotPresent
 command:
 - /bin/sh
 - -c
 - date; echo Hello
 restartPolicy: OnFailure
```

If you create a cron job again, you can temporarily avoid this issue.

### 20.6.4.2 What Is a Headless Service When I Create a StatefulSet?

The inter-pod discovery service of CCE corresponds to the headless Service of Kubernetes. Headless Services specify **None** for the cluster IP (spec:clusterIP) in YAML, which means no cluster IP is allocated.

### Differences Between Headless Services and Common Services

- **Common Services:**  
One Service may be backed by multiple endpoints (pods). A client accesses the cluster IP address and the request is forwarded to the real server based on the iptables or IPVS rules to implement load balancing. For example, a Service has two endpoints, but only the Service address is returned during DNS query. The iptables or IPVS rules determine the real server that the client accesses. The client cannot access the specified endpoint.
- **Headless Services:**  
When a headless Service is accessed, the actual endpoint (pod IP addresses) is returned. The headless Service points directly to each endpoint, that is, each pod has a DNS domain name. In this way, pods can access each other, achieving inter-pod discovery and access.

### Headless Service Application Scenarios

If there is no difference between multiple pods of a workload, you can use a common Service and use the cluster kube-proxy to implement load balancing, for example, an Nginx Deployment.

However, in some application scenarios, pods of a workload have different roles. For example, in a Redis cluster, each Redis pod is different. They have a master/slave relationship and need to communicate with each other. In this case, a common Service cannot access a specified pod through the cluster IP address. Therefore, you need to allow the headless Service to directly access the real IP address of the pod to implement mutual access among pods.

Headless Services work with [StatefulSet](#) to deploy stateful applications, such as Redis and MySQL.

### 20.6.4.3 What Should I Do If Error Message "Auth is empty" Is Displayed When a Private Image Is Pulled?

#### Problem Description

When you replace the image of a container in a created workload and use an uploaded image on the CCE console, an error message "Auth is empty, only accept X-Auth-Token or Authorization" is displayed when the uploaded image is pulled.

```
Failed to pull image "IP address:Port number /magicdoom/tidb-operator:latest": rpc error: code = Unknown desc = Error response from daemon: Get https://IP address:Port number /v2/magicdoom/tidb-operator/manifests/latest: error parsing HTTP 400 response body: json: cannot unmarshal number into Go struct field Error.code of type errcode.ErrorCode: "{\"errors\": [{\"code\":400,\"message\": \"Auth is empty, only accept X-Auth-Token or Authorization.\"}]}"
```

#### Solution

You can select a private image to create an application on the CCE console. In this case, CCE automatically carries the secret. This problem will not occur during the upgrade.

When you create a workload using an API, you can include the secret in Deployments to avoid this problem during the upgrade.

```
imagePullSecrets:
- name: default-secret
```

### 20.6.4.4 Why Cannot a Pod Be Scheduled to a Node?

- Step 1** Check whether the node and Docker are normal. For details, see [Check Item 7: Whether Internal Components Are Normal](#).
- Step 2** If the node and Docker are normal, check whether an affinity policy is configured for the pod. For details, see [Check Item 3: Affinity and Anti-Affinity Configuration of the Workload](#).
- Step 3** Check whether the resources on the node are sufficient. If the resources are insufficient, expand the capacity or add nodes.

----End

### 20.6.4.5 What Is the Image Pull Policy for Containers in a CCE Cluster?

A container image is required to create a container. Images may be stored locally or in a remote image repository.

The **imagePullPolicy** field in the Kubernetes configuration file is used to describe the image pull policy. This field has the following value options:

- **Always:** Always force a pull.  
imagePullPolicy: Always
- **IfNotPresent:** The image is pulled only if it is not already present locally.  
imagePullPolicy: IfNotPresent
- **Never:** The image is assumed to exist locally. No attempt is made to pull the image.  
imagePullPolicy: Never

#### Description

1. If this field is set to **Always**, the image is pulled from the remote repository each time a container is started or restarted.  
If **imagePullPolicy** is left blank, the policy defaults to **Always**.
2. If the policy is set to **IfNotPresent**:
  - a. If the required image does not exist locally, it will be pulled from the remote repository.
  - b. If the content, except the tag, of the required image is the same as that of the local image, and the image with that tag exists only in the remote repository, Kubernetes will not pull the image from the remote repository.

### 20.6.4.6 What Can I Do If a Layer Is Missing During Image Pull?

#### Symptom

When containerd is used as the container engine, there is a possibility that the image layer is missing when an image is pulled to a node. As a result, the workload container fails to be created.

```

events:
Type Reason Age From Message
---- -
Normal Scheduled 54s default-scheduler Successfully assigned cattle-prometheus/prometheus-server-6c69469c-f4-nfs7f to 10.14.11.139
Normal SuccessfulMountVolume 55s kubelet Successfully mounted volumes for pod "prometheus-server-6c69469c-f4-nfs7f_cattle-prometheus(48ac202a-649a-429c-91ca-573dbaacb72)"
Normal SuccessfulUpdateSecurityGroup 52s yamglse-controller Successfully updated security group to "c8d07f89-0f46-421a-0901-ed072805106c"
Normal Failed 8s (x6 over 53s) kubelet Container image "199.125.0.255:20202/prometheus/prometheus:v2.27.0" already present on machine
Warning FailedCreate 7s (x6 over 50s) kubelet Error: failed to create container container: error unpacking image: failed to extract layer sha256:f9d9e4e62f0689cd752390e14de48b0ec6f2488a05af5ab2f0cca154c299d: failed to get reader from content store: content digest sha256:8c5a7d1a1fbc62695fcb2c95445743cc51f22053ea29ea9e0d0773b7068105: not found

```

#### Possible Cause

Docker earlier than v1.10 supports the layer whose **mediaType** is **application/octet-stream**. However, containerd does not support **application/octet-stream**. As a result, the layer is not pulled.

#### Solution

You can use either of the following methods to solve this problem:

- Use Docker v1.11 or later to repackage the image.
- Manually pull the image.
  - a. Log in to the node.
  - b. Run the following command to pull the image:  
**ctr -n k8s.io images pull --user u:p images**
  - c. Use the newly pulled image to create a workload.

## 20.7 Networking

### 20.7.1 Network Planning

#### 20.7.1.1 What Is the Relationship Between Clusters, VPCs, and Subnets?

A Virtual Private Cloud (VPC) is similar to a private local area network (LAN) managed by a home gateway whose IP address is 192.168.0.0/16. A VPC is a



private network built on the cloud and provides basic network environment for running elastic cloud servers (ECSs), elastic load balances (ELBs), and middleware. Networks of different scales can be configured based on service requirements. Generally, you can set the CIDR block to 10.0.0.0/8–24, 172.16.0.0/12–24, or 192.168.0.0/16–24. The largest CIDR block is 10.0.0.0/8, which corresponds to a class A network.

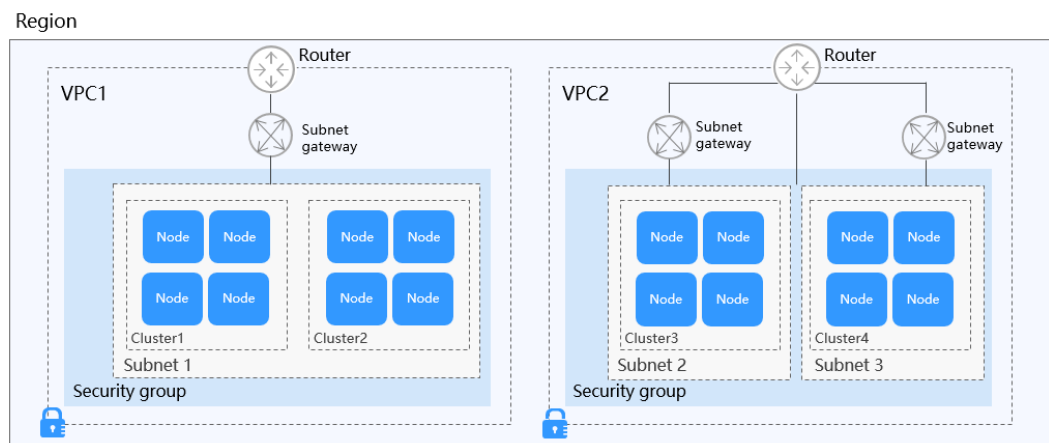
A VPC can be divided into multiple subnets. Security groups are configured to determine whether these subnets can communicate with each other. This ensures that subnets can be isolated from each other, so that you can deploy different services on different subnets.

A cluster is one or a group of cloud servers (also known as nodes) in the same VPC. It provides computing resource pools for running containers.

As shown in [Figure 20-8](#), a region may comprise of multiple VPCs. A VPC consists of one or more subnets. The subnets communicate with each other through a subnet gateway. A cluster is created in a subnet. There are three scenarios:

- Different clusters are created in different VPCs.
- Different clusters are created in the same subnet.
- Different clusters are created in different subnets.

**Figure 20-8** Relationship between clusters, VPCs, and subnets



### 20.7.1.2 Configuring Cluster Security Group Rules

CCE is a universal container platform. Its default security group rules apply to common scenarios. When a cluster is created, a security group is automatically created for the master node and worker node, separately. The security group name of the master node is *{Cluster name}-cce-control-{Random ID}*, and the security group name of the worker node is *{Cluster name}-cce-node-{Random ID}*. If a CCE Turbo cluster is used, an additional ENI security group named *{Cluster name}-cce-eni-{Random ID}* will be created.

You can log in to the management console, choose **Service List > Networking > Virtual Private Cloud > Access Control > Security Groups**, locate the security group of the cluster, and modify the security group rules as required.

The default security group rules of the clusters using different networks are as follows:

- [Security Group Rules of a Cluster Using a VPC Network](#)
- [Security Group Rules of a Cluster Using the Tunnel Network](#)
- [Security Group Rules of a CCE Turbo Cluster Using the Cloud Native Network 2.0](#)

**NOTICE**

Modifying or deleting rules in a security group may affect cluster running. Exercise caution when performing this operation. If you need to modify security group rules, do not modify the rules of the port on which CCE running depends.

## Security Group Rules of a Cluster Using a VPC Network

### Security group of a worker node

A security group named *{Cluster name}-cce-node-{Random ID}* is automatically created for each worker node. For details about the default ports, see [Table 20-7](#).

**Table 20-7** Default ports in the security group for a worker node that uses a VPC network

Direction	Port	Default Source Address	Description	Modifiable	Modification Suggestion
Inbound rules	All UDP ports	VPC CIDR block	Allow access between worker nodes and between worker nodes and the master node.	No	N/A
	All TCP ports				
	All ICMP ports	Security group of the master node	Allow the master node to access worker nodes.	No	N/A
	TCP port range: 30000 to 32767	All IP addresses: <b>0.0.0.0/0</b>	Allow access from NodePort.	Yes	Allow access from VPC, container, and ELB CIDR blocks.
	UDP port range: 30000 to 32767				
	All ports	Container CIDR block	Allow access between nodes and containers.	No	N/A

Direction	Port	Default Source Address	Description	Modifiable	Modification Suggestion
	All ports	Security group of worker nodes	Allow access between worker nodes.	No	N/A
	TCP port 22	All IP addresses: <b>0.0.0.0/0</b>	Allow SSH access to ECSs.	<b>Recommended</b>	N/A
Outbound rule	All ports	All IP addresses: <b>0.0.0.0/0</b>	Allow traffic on all ports by default. You are advised to retain this setting.	Yes	If you want to harden security by allowing traffic only on specific ports, remember to allow such ports. For details, see <a href="#">Hardening Outbound Rules</a> .

### Security group of the master node

A security group named *{Cluster name}-cce-control-{Random ID}* is automatically created for the master node. For details about the default ports, see [Table 20-8](#).

**Table 20-8** Default ports in the security group for the master node that uses a VPC network

Direction	Port	Default Source Address	Description	Modifiable	Modification Suggestion
Inbound rules	TCP port 5444	VPC CIDR block	Allow access from kube-apiserver, which provides lifecycle management for Kubernetes resources.	No	N/A
	TCP port 5444	Container CIDR block			

Direction	Port	Default Source Address	Description	Modifiable	Modification Suggestion
	TCP port 9443	VPC CIDR block	Allow the network add-on of a worker node to access the master node.	No	N/A
	TCP port 5443	All IP addresses: <b>0.0.0.0/0</b>	Allow kube-apiserver of the master node to listen to worker nodes.	<b>Recommended</b>	The port must allow traffic from the CIDR blocks of the VPC, container, and the control plane of the hosted service mesh.
	TCP port 8445	VPC CIDR block	Allow the storage add-on of a worker node to access the master node.	No	N/A
	All ports	IP addresses of this security group	Allow traffic from all IP addresses of this security group.	No	N/A
Outbound rule	All ports	All IP addresses: <b>0.0.0.0/0</b>	Allow traffic on all ports by default.	No	N/A

## Security Group Rules of a Cluster Using the Tunnel Network

### Security group of a worker node

A security group named *{Cluster name}-cce-node-{Random ID}* is automatically created for each worker node. For details about the default ports, see [Table 20-9](#).

**Table 20-9** Default ports in the security group for a worker node that uses the tunnel network

Direction	Port	Default Source Address	Description	Modifiable	Modification Suggestion
Inbound rules	UDP port 4789	All IP addresses: <b>0.0.0.0/0</b>	Allow access between containers.	No	N/A
	TCP port 10250	CIDR block of the master node	Allow the master node to access kubelet on a worker node, for example, by running <b>kubectl exec {pod}</b> .	No	N/A
	TCP port range: 30000 to 32767	All IP addresses: <b>0.0.0.0/0</b>	Allow access from NodePort.	Yes	Allow access from VPC, container, and ELB CIDR blocks.
	UDP port range: 30000 to 32767				
	TCP port 22	All IP addresses: <b>0.0.0.0/0</b>	Allow SSH access to ECSs.	<b>Recommended</b>	N/A
	All ports	IP addresses of this security group	Allow traffic from all IP addresses of this security group.	No	N/A
Outbound rule	All ports	All IP addresses: <b>0.0.0.0/0</b>	Allow traffic on all ports by default. You are advised to retain this setting.	Yes	If you want to harden security by allowing traffic only on specific ports, remember to allow such ports. For details, see <a href="#">Hardening Outbound Rules</a> .

### Security group of the master node

A security group named *{Cluster name}-cce-control-{Random ID}* is automatically created for the master node. For details about the default ports, see [Table 20-10](#).

**Table 20-10** Default ports in the security group for the master node that uses the Tunnel network

Direction	Port	Default Source Address	Description	Modifiable	Modification Suggestion
Inbound rules	UDP port 4789	All IP addresses: <b>0.0.0.0/0</b>	Allow access between containers.	No	N/A
	TCP port 5444	VPC CIDR block	Allow access from kube-apiserver, which provides lifecycle management for Kubernetes resources.	No	N/A
	TCP port 5444	Container CIDR block			
	TCP port 9443	VPC CIDR block	Allow the network add-on of a worker node to access the master node.	No	N/A
	TCP port 5443	All IP addresses: <b>0.0.0.0/0</b>	Allow kube-apiserver of the master node to listen to worker nodes.	<b>Recommended</b>	The port must allow traffic from the CIDR blocks of the VPC, container, and the control plane of the hosted service mesh.
	TCP port 8445	VPC CIDR block	Allow the storage add-on of a worker node to access the master node.	No	N/A
	All ports	IP addresses of this security group	Allow traffic from all IP addresses of this security group.	No	N/A

Direction	Port	Default Source Address	Description	Modifiable	Modification Suggestion
Outbound rule	All ports	All IP addresses: <b>0.0.0.0/0</b>	Allow traffic on all ports by default.	No	N/A

## Security Group Rules of a CCE Turbo Cluster Using the Cloud Native Network 2.0

### Security group of a worker node

A security group named *{Cluster name}-cce-node-{Random ID}* is automatically created for each worker node. For details about the default ports, see [Table 20-11](#).

**Table 20-11** Default ports in the security group for a worker node

Direction	Port	Default Source Address	Description	Modifiable	Modification Suggestion
Inbound rules	TCP port 10250	CIDR block of the master node	Allow the master node to access kubelet on a worker node, for example, by running <b>kubect exec {pod}</b> .	No	N/A
	TCP port range: 30000 to 32767	All IP addresses: <b>0.0.0.0/0</b>	Allow access from NodePort.	Yes	Allow access from VPC, container, and ELB CIDR blocks.
	UDP port range: 30000 to 32767				
	TCP port 22	All IP addresses: <b>0.0.0.0/0</b>	Allow SSH access to ECSs.	<b>Recommended</b>	N/A
	All ports	IP addresses of this security group	Allow traffic from all IP addresses of this security group.	No	N/A

Direction	Port	Default Source Address	Description	Modifiable	Modification Suggestion
	All ports	Subnet CIDR block of the master node	Allow traffic from all source IP addresses in the subnet CIDR block of the master node.	No	N/A
Outbound rule	All ports	All IP addresses: <b>0.0.0.0/0</b>	Allow traffic on all ports by default. You are advised to retain this setting.	Yes	If you want to harden security by allowing traffic only on specific ports, remember to allow such ports. For details, see <a href="#">Hardening Outbound Rules</a> .

### Security group of the master node

A security group named *{Cluster name}-cce-control-{Random ID}* is automatically created for the master node. For details about the default ports, see [Table 20-12](#).

**Table 20-12** Default ports in the security group for the master node

Direction	Port	Default Source Address	Description	Modifiable	Modification Suggestion
Inbound rules	TCP port 5444	All IP addresses: <b>0.0.0.0/0</b>	Allow access from kube-apiserver, which provides lifecycle management for Kubernetes resources.	No	N/A
	TCP port 5444	VPC CIDR block		No	N/A
	TCP port 9443	VPC CIDR block	Allow the network add-on of a worker node to access the master node.	No	N/A



Direction	Port	Default Source Address	Description	Modifiable	Modification Suggestion
	TCP port 5443	All IP addresses: <b>0.0.0.0/0</b>	Allow kube-apiserver of the master node to listen to worker nodes.	<b>Recommended</b>	The port must allow traffic from the CIDR blocks of the VPC, container, and the control plane of the hosted service mesh.
	TCP port 8445	VPC CIDR block	Allow the storage add-on of a worker node to access the master node.	No	N/A
	All ports	IP addresses of this security group	Allow traffic from all IP addresses of this security group.	No	N/A
	All ports	Subnet CIDR block of the master node	Allow traffic from all source IP addresses in the subnet CIDR block of the master node.	No	N/A
Outbound rule	All ports	All IP addresses: <b>0.0.0.0/0</b>	Allow traffic on all ports by default.	No	N/A

### Security group of an ENI

An additional security group named *{Cluster name}-cce-eni-{Random ID}* is created for a CCE Turbo cluster. For details about the default ports, see [Table 20-13](#).

**Table 20-13** Default ports in the security group for an ENI

Direction	Port	Default Source Address	Description	Modifiable	Modification Suggestion
Inbound rules	All ports	IP addresses of this security group	Allow traffic from all IP addresses of this security group.	No	N/A
		VPC CIDR block	Allow traffic from all IP addresses of the VPC CIDR block.	No	N/A
Outbound rule	All ports	All IP addresses: <b>0.0.0.0/0</b>	Allow traffic on all ports by default.	No	N/A

## Hardening Outbound Rules

By default, all security groups created by CCE allow all the **outbound** traffic. You are advised to retain this configuration. If you want to harden security by allowing traffic only on specific ports, remember to allow ports listed in the following table.

**Table 20-14** Ports that must be allowed in outbound rules for a worker node

Port	Allowed CIDR	Description
UDP port 53	DNS server of the subnet	Allow traffic on the port for domain name resolution.
UDP port 4789 (required only for clusters that use the Tunnel network)	All IP addresses	Allow access between containers.
TCP port 5443	CIDR block of the master node	Allow kube-apiserver of the master node to listen to worker nodes.
TCP port 5444	CIDR blocks of the VPC and container	Allow access from kube-apiserver, which provides lifecycle management for Kubernetes resources.
TCP port 6443	CIDR block of the master node	None

Port	Allowed CIDR	Description
TCP port 8445	VPC CIDR block	Allow the storage add-on of a worker node to access the master node.
TCP port 9443	VPC CIDR block	Allow the network add-on of a worker node to access the master node.

## 20.7.2 Network Fault

### 20.7.2.1 How Do I Locate a Workload Networking Fault?

#### Troubleshooting Process

Troubleshooting methods are sorted based on the occurrence probability of the possible causes. You are advised to check the possible causes from high probability to low probability to quickly locate the cause of the problem.

If the fault persists after a possible cause is rectified, check other possible causes.

- [Check Item 1: Container and Container Port](#)
- [Check Item 2: Node IP Address and Node Port](#)
- [Check Item 3: ELB IP Address and Port](#)
- [Check Item 4: NAT Gateway + Port](#)
- [Check Item 5: Whether the Security Group of the Node Where the Container Is Located Allows Access](#)

#### Check Item 1: Container and Container Port

Log in to the CCE console or use `kubectl` to query the IP address of the pod. Then, log in to the node or container in the cluster and run the `curl` command to manually call the API. Check whether the expected result is returned.

If `<container IP address>:<port>` cannot be accessed, you are advised to log in to the application container and access `<127.0.0.1>:<port>` to locate the fault.

##### Common issues:

1. The container port is incorrectly configured (the container does not listen to the access port).
2. The URL does not exist (no related path exists in the container).
3. A Service exception (a Service bug in the container) occurs.
4. Check whether the cluster network kernel component is abnormal (container tunnel network model: openswitch kernel component; VPC network model: ipvlan kernel component).

## Check Item 2: Node IP Address and Node Port

Only NodePort or LoadBalancer Services can be accessed using the node IP address and node port.

- **NodePort Services:**

The access port of a node is the port exposed externally by the node.

- **LoadBalancer Service:**

You can view the node port of a LoadBalancer Service by editing the YAML file.

Example:

**nodePort: 30637** indicates the exposed node port. **targetPort: 80** indicates the exposed pod port. **port: 123** is the exposed Service port. LoadBalancer Services also use this port to configure the ELB listener.

```
spec:
 ports:
 - name: cce-service-0
 protocol: TCP
 port: 123
 targetPort: 80
 nodePort: 30637
```

After finding the node port (nodePort), access <IP address>:<port> of the node where the container is located and check whether the expected result is returned.

### Common issues:

1. The service port is not allowed in the inbound rules of the node.
2. A custom route is incorrectly configured for the node.
3. The label of the pod does not match that of the Service (created using kubectl or API).

## Check Item 3: ELB IP Address and Port

There are several possible causes if <IP address>:<port> of the ELB cannot be accessed, but <IP address>:<port> of the node can be accessed.

### Possible causes:

- The backend server group of the port or URL does not meet the expectation.
- The security group on the node has not exposed the related protocol or port to the ELB.
- The health check of the layer-4 load balancing is not enabled.
- The certificate used for Services of layer-7 load balancing has expired.

### Common issues:

1. When exposing a layer-4 ELB load balancer, if you have not enabled health check on the console, the load balancer may route requests to abnormal nodes.

2. For UDP access, the ICMP port of the node has not been allowed in the inbound rules.
3. The label of the pod does not match that of the Service (created using kubectl or API).

#### Check Item 4: NAT Gateway + Port

Generally, no EIP is configured for the backend server of NAT. Otherwise, exceptions such as network packet loss may occur.

#### Check Item 5: Whether the Security Group of the Node Where the Container Is Located Allows Access

Log in to the management console, choose **Service List > Networking > Virtual Private Cloud**. On the Network console, choose **Access Control > Security Groups**, locate the security group rule of the CCE cluster, and modify and harden the security group rule.

- CCE cluster:  
The security group name of the node is **{Cluster name}-cce-node-{Random characters}**.

##### Check the following:

- IP address, port, and protocol of an external request to access the workloads in the cluster. They must be allowed in the inbound rule of the cluster security group.
- IP address, port, and protocol of a request by a workload to visit external applications outside the cluster. They must be allowed in the outbound rule of the cluster security group.

For details about security group configuration, see [Configuring Cluster Security Group Rules](#).

#### 20.7.2.2 Why Does the Browser Return Error Code 404 When I Access a Deployed Application?

CCE does not return any error code when you fail to access your applications using a browser. Check your services first.

##### 404 Not Found

If the error code shown in the following figure is returned, it indicates that the ELB cannot find the corresponding forwarding policy. Check the forwarding policies.

Figure 20-9 404:ALB



# 404 Not Found

ALB

If the error code shown in the following figure is returned, it indicates that errors occur on Nginx (your services). In this case, check your services.

**Figure 20-10** 404:nginx/1.\*\*.\*

---

# 404 Not Found

---

nginx/1.14.0

### 20.7.2.3 What Should I Do If a Container Fails to Access the Internet?

If a container cannot access the Internet, check whether the node where the container is located can access the Internet. Then check whether the network configuration of the container is correct. For example, check whether the DNS configuration can resolve the domain name.

#### Check Item 1: Whether the Node Can Access the Internet

**Step 1** Log in to the ECS console.

**Step 2** Check whether the ECS corresponding to the node has been bound to an EIP or has a NAT gateway configured.

An EIP has been bound. If no EIP is displayed, bind an EIP to the ECS.

----End

#### Check Item 2: Whether a Network ACL Has Been Configured for the Node

**Step 1** Log in to the VPC console.

**Step 2** In the navigation pane on the left, choose **Access Control > Network ACLs**.

**Step 3** Check whether a network ACL has been configured for the subnet where the node is located and whether external access is restricted.

----End

#### Check Item 3: Whether the DNS Configuration of the Container Is Correct

Run the **cat /etc/resolv.conf** command in the container to check the DNS configuration. An example is as follows:

```
nameserver 10.247.x.x
search default.svc.cluster.local svc.cluster.local cluster.local
options ndots:5
```

If **nameserver** is set to **10.247.x.x**, DNS is connected to the CoreDNS of the cluster. Ensure that the CoreDNS of the cluster is running properly. If another IP address is displayed, an in-cloud or on-premises DNS server is used. Ensure that the domain name resolution is correct.

## 20.7.2.4 What Should I Do If a Node Fails to Connect to the Internet (Public Network)?

If a node fails to be connected to the Internet, perform the following operations:

### Check Item 1: Whether an EIP Has Been Bound to the Node

Log in to the ECS console and check whether an EIP has been bound to the ECS corresponding to the node.

If there is an IP address in the EIP column, an EIP has been bound. If there is no IP address in that column, bind one.

### Check Item 2: Whether a Network ACL Has Been Configured for the Node

Log in to the VPC console. In the navigation pane, choose **Access Control > Network ACLs**. Check whether a network ACL has been configured for the subnet where the node is located and whether external access is restricted.

## 20.7.2.5 What Should I Do If an Nginx Ingress Access in the Cluster Is Abnormal After the Add-on Is Upgraded?

### Symptom

An Nginx Ingress whose type is not specified (**kubernetes.io/ingress.class: nginx** is not added to annotations) exists in the cluster. After the nginx-ingress add-on is upgraded from 1.x to 2.x, services are interrupted.

### Fault Locating

For an 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 ELB 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@xxxxxxxxx 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={{IPs:+++++ Hostname: Ports:[]}} {IPs:+++++ Hostname: Ports:[]}}
```

If the fault persists, contact technical support personnel.

----End

## Solution

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.

## Possible Causes

The nginx-ingress add-on is developed based on the Nginx Ingress Controller template and image of the open source community.

For the Nginx Ingress Controller of an earlier version (community version v0.49 or earlier, corresponding to CCE nginx-ingress version v1.x.x), the Ingress type is not specified as nginx during Ingress creation, which is, **kubernetes.io/ingress.class: nginx** is not added to annotations. This Ingress can also be managed by Nginx Ingress Controller. For details, see the [GitHub code](#).

For the Nginx Ingress Controller of a later version (community version v1.0.0 or later, corresponding to CCE nginx-ingress version 2.x.x), if the Ingress type is not specified as nginx during Ingress creation, this Ingress will be ignored by the Nginx Ingress Controller and the Ingress rules become invalid. The services will be interrupted. For details, see the [GitHub code](#).

Related link: <https://github.com/kubernetes/ingress-nginx/pull/7341>

You can specify the Ingress type in either of the following ways:

- Add the **kubernetes.io/ingress.class: nginx** annotation to the Ingresses.
- Use spec. Set the **.spec.ingressClassName** field to **nginx**. IngressClass resources are required.

An example is as follows:

```
apiVersion: networking.k8s.io/v1beta1
kind: Ingress
metadata:
 name: test
 namespace: default
 annotations:
 kubernetes.io/ingress.class: nginx
spec:
```



```
ingressClassName: nginx
rules:
...
status:
loadBalancer: {}
```

## 20.8 Storage

### 20.8.1 What Are the Differences Among CCE Storage Classes in Terms of Persistent Storage and Multi-node Mounting?

Container storage provides storage for container workloads. It supports multiple storage classes. A pod can use any amount of storage.

Currently, CCE supports local, EVS, SFS, SFS Turbo, and OBS volumes.

The following table lists the differences among these storage classes.

**Table 20-15** Differences among storage classes

Storage Class	Persistent Storage	Automatic Migration with Containers	Multi-node Mounting
Local disks	Supported	Not supported	Not supported
EVS	Supported	Supported	Not supported
OBS	Supported	Supported	Supported. This type of volumes can be shared among multiple nodes or workloads.
SFS	Supported	Supported	Supported. This type of volumes can be shared among multiple nodes or workloads.
SFS Turbo	Supported	Supported	Supported. This type of volumes can be shared among multiple nodes or workloads.

### Selecting a Storage Class

You can use the following types of storage volumes when creating a workload. You are advised to store workload data on EVS volumes. If you store workload data on a local volume, the data cannot be restored when a fault occurs on the node.

- Local volumes: Mount the file directory of the host where a container is located to a specified container path (corresponding to hostPath in Kubernetes). Alternatively, you can leave the source path empty (corresponding to emptyDir in Kubernetes). If the source path is left empty, a temporary directory of the host will be mounted to the mount point of the

container. A specified source path is used when data needs to be persistently stored on the host, while `emptyDir` is used when temporary storage is needed. A `ConfigMap` is a type of resource that stores configuration data required by a workload. Its contents are user-defined. A `Secret` is an object that contains sensitive data such as workload authentication information and keys. Information stored in a `Secret` is determined by users.

- **EVS volumes:** Mount an EVS volume to a container path. When the container is migrated, the mounted EVS volume is migrated together. This storage class is applicable when data needs to be stored permanently.
- **SFS volumes:** Create SFS volumes and mount them to a container path. The file system volumes created by the underlying SFS service can also be used. SFS volumes are applicable to persistent storage for frequent read/write in multiple workload scenarios, including media processing, content management, big data analysis, and workload analysis.
- **OBS volumes:** Create OBS volumes and mount them to a container path. OBS volumes are applicable to scenarios such as cloud workload, data analysis, content analysis, and hotspot objects.
- **SFS Turbo volumes:** Create SFS Turbo volumes and mount them to a container path. SFS Turbo volumes are fast, on-demand, and scalable, which makes them suitable for DevOps, containerized microservices, and enterprise office applications.

## 20.8.2 Can I Add a Node Without a Data Disk?

No. A data disk is mandatory.

A data disk dedicated for kubelet and the container engine will be attached to a new node. By default, CCE uses Logical Volume Manager (LVM) to manage data disks. With LVM, you can adjust the disk space ratio for different resources on a data disk.

If the data disk is uninstalled or damaged, the container engine will malfunction and the node becomes unavailable.

## 20.8.3 What Should I Do If the Host Cannot Be Found When Files Need to Be Uploaded to OBS During the Access to the CCE Service from a Public Network?

When a Service deployed on CCE attempts to upload files to OBS after receiving an access request from an offline machine, an error message is displayed, indicating that the host cannot be found. The following figure shows the error message:

Time	message
February 22nd 2020, 18:50:27.521	com.obs.services.exception.ObsException: OBS service <b>Error</b> Message. Request <b>Error</b> : java.net.UnknownHostException: obs.oss-cn-hangzhou.aliyuncs.com
February 22nd 2020, 18:50:27.521	18:50:27.520 [XNIO-1 task-16] <b>ERROR</b> c.h.f.c.provider.ExceptionProvider - OBS service <b>Error</b> Message. Request <b>Error</b> : java.net.UnknownHostException: obs.oss-cn-hangzhou.aliyuncs.com
February 22nd 2020, 18:50:27.298	18:50:27.298 [XNIO-1 task-9] <b>ERROR</b> c.h.f.c.provider.ExceptionProvider - OBS service <b>Error</b> Message. Request <b>Error</b> : java.net.UnknownHostException: obs.oss-cn-hangzhou.aliyuncs.com
February 22nd 2020, 18:50:27.298	com.obs.services.exception.ObsException: OBS service <b>Error</b> Message. Request <b>Error</b> : java.net.UnknownHostException: obs.oss-cn-hangzhou.aliyuncs.com
February 22nd 2020, 18:50:27.275	18:50:27.274 [XNIO-1 task-9] <b>WARN</b> c.o.s.internal.RestStorageService - com.obs.services.internal.ServiceException: Request <b>Error</b> : java.net.UnknownHostException: obs.oss-cn-hangzhou.aliyuncs.com HEAD 'https://obs.oss-cn-hangzhou.aliyuncs.com/obs-it-problem-management-media-test?apiversion' on Host 'obs.oss-cn-hangzhou.aliyuncs.com'
February 22nd 2020, 18:50:27.275	com.obs.services.internal.ServiceException: Request <b>Error</b> : java.net.UnknownHostException: obs.oss-cn-hangzhou.aliyuncs.com
February 22nd 2020, 18:50:27.275	2020-02-22 18:50:27 274 com.obs.services.internal.RestStorageService handleThrowable 205 com.obs.services.internal.ServiceException: Request <b>Error</b> : java.net.UnknownHostException:

## Fault Locating

After receiving the HTTP request, the Service transfers files to OBS through the proxy.

If too many files are transferred, a large number of resources are consumed. Currently, the proxy is assigned 128 MiB memory. According to pressure test results, resource consumption is large, resulting in request failure.

The test results show that all traffic passes through the proxy. Therefore, if the service volume is large, more resources need to be allocated.

## Solution

1. File transfer involves a large number of packet copies, which occupies a large amount of memory. You are advised to increase the proxy memory based on the actual scenario and then try to access the Service and upload files again.
2. In addition, you can remove the Service from the mesh because the proxy only forwards packets and does not perform any other operations. If requests pass through the ingress gateway, the grayscale release function of the Service is not affected.

## 20.8.4 How Can I Achieve Compatibility Between ExtendPathMode and Kubernetes client-go?

### Scenario

The Kubernetes pod structure does not contain **ExtendPathMode**. Therefore, when a user calls the API for creating a pod or deployment by using client-go, the created pod does not contain **ExtendPathMode**. CCE provides a solution to ensure compatibility with the Kubernetes client-go.

## Solution

### NOTICE

- When creating a pod, you need to add **kubernetes.io/extend-path-mode** to **annotation** of the pod.
- When creating a Deployment, you need to add **kubernetes.io/extend-path-mode** to **kubernetes.io/extend-path-mode** in the template.

The following is an example YAML of creating a pod. After the **kubernetes.io/extend-path-mode** keyword is added to **annotation**, the **containername**, **name**, and **mountpath** fields are matched, and the corresponding **extendpathmode** is added to **volumeMount**.

```
apiVersion: v1
kind: Pod
metadata:
 name: test-8b59d5884-96vdz
 generateName: test-8b59d5884-
 namespace: default
 selfLink: /api/v1/namespaces/default/pods/test-8b59d5884-96vdz
 labels:
 app: test
 pod-template-hash: 8b59d5884
 annotations:
 kubernetes.io/extend-path-mode:
 '[{"containername":"container-0","name":"vol-156738843032165499","mountpath":"/
 tmp","extendpathmode":"PodUID"}]'
 metrics.alpha.kubernetes.io/custom-endpoints: '[{"api":"","path":"","port":"","names":""}]'
ownerReferences:
 - apiVersion: apps/v1
 kind: ReplicaSet
 name: test-8b59d5884
 uid: 2633020b-cd23-11e9-8f83-fa163e592534
 controller: true
 blockOwnerDeletion: true
spec:
 volumes:
 - name: vol-156738843032165499
 hostPath:
 path: /tmp
 type: ""
 - name: default-token-4s959
 secret:
 secretName: default-token-4s959
 defaultMode: 420
 containers:
 - name: container-0
 image: 'nginx:latest'
 env:
 - name: PAAS_APP_NAME
 value: test
 - name: PAAS_NAMESPACE
 value: default
 - name: PAAS_PROJECT_ID
 value: b6315dd3d0ff4be5b31a963256794989
 resources:
 limits:
 cpu: 250m
 memory: 512Mi
 requests:
 cpu: 250m
 memory: 512Mi
```

```

volumeMounts:
 - name: vol-156738843032165499
 mountPath: /tmp
 extendPathMode: PodUID
 - name: default-token-4s959
 readOnly: true
 mountPath: /var/run/secrets/kubernetes.io/serviceaccount
terminationMessagePath: /dev/termination-log
terminationMessagePolicy: File
imagePullPolicy: Always
restartPolicy: Always
terminationGracePeriodSeconds: 30
dnsPolicy: ClusterFirst
serviceAccountName: default
serviceAccount: default
nodeName: 192.168.0.24
securityContext: {}
imagePullSecrets:
 - name: default-secret
 - name: default-secret
affinity: {}
schedulerName: default-scheduler
tolerations:
 - key: node.kubernetes.io/not-ready
 operator: Exists
 effect: NoExecute
 tolerationSeconds: 300
 - key: node.kubernetes.io/unreachable
 operator: Exists
 effect: NoExecute
 tolerationSeconds: 300
priority: 0
dnsConfig:
 options:
 - name: timeout
 value: ""
 - name: ndots
 value: '5'
 - name: single-request-reopen
enableServiceLinks: true

```

**Table 20-16** Descriptions of key parameters

Parameter	Type	Description
containername	String	Name of a container.
name	String	Name of a volume.
mountpath	String	Mount path.

Parameter	Type	Description
extendpathmode	String	<p>A third-level directory is added to the created volume directory/subdirectory to facilitate the obtaining of a single pod output file.</p> <p>The following types are supported.</p> <ul style="list-style-type: none"> <li>● <b>None:</b> The extended path is not 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>

## 20.8.5 Can CCE PVCs Detect Underlying Storage Faults?

CCE PersistentVolumeClaims (PVCs) are implemented as they are in Kubernetes. A PVC is defined as a storage declaration and is decoupled from underlying storage. It is not responsible for detecting underlying storage details. Therefore, CCE PVCs cannot detect underlying storage faults.

Cloud Eye allows users to view cloud service metrics. These metrics are built-in based on cloud service attributes. After users enable a cloud service on the cloud platform, Cloud Eye automatically associates its built-in metrics. Users can track the cloud service status by monitoring these metrics.

It is recommended that users who have storage fault detection requirements use Cloud Eye to monitor underlying storage and send alarm notifications.

## 20.9 Namespace

### 20.9.1 Why Cannot I Delete a Namespace Due to an APIService Object Access Failure?

#### Symptom

The namespace remains in the Deleting state. The error message "DiscoveryFailed" is displayed in **status** in the YAML file.

```

75 - kube:metrics
76 status:
77 phase: Terminating
78 conditions:
79 - type: NamespaceDeletionDiscoveryFailure
80 status: 'True'
81 lastTransitionTime: '2022-07-04T13:44:55Z'
82 reason: DiscoveryFailed
83 message: 'Discovery failed for some groups, 1 failing: unable to retrieve the complete list of server
84 APIs: metrics.k8s.io/v1beta1: the server is currently unable to handle the request'
85 - type: NamespaceDeletionGroupVersionParsingFailure
86 status: 'False'

```

In the preceding figure, the full error message is "Discovery failed for some groups, 1 failing: unable to retrieve the complete list of server APIs: metrics.k8s.io/v1beta1: the server is currently unable to handle the request".

This indicates that the namespace deletion is blocked when kube-apiserver accesses the APIService resource object of the metrics.k8s.io/v1beta1 API.

## Possible Causes

If an APIService object exists in the cluster, deleting the namespace will first access the APIService object. If the access fails, the namespace deletion will be blocked. In addition to the APIService objects created by users, add-ons like metrics-server and prometheus in the CCE cluster automatically create APIService objects.

### NOTE

For details, see <https://kubernetes.io/docs/concepts/extend-kubernetes/api-extension/apiserver-aggregation/>.

## Solution

Use either of the following methods:

- Rectify the APIService object in the error message. If the object is created by an add-on, ensure that the pod where the add-on locates is running properly.
- Delete the APIService object in the error message. If the object is created by an add-on, uninstall the add-on.

## 20.10 Chart and Add-on

### 20.10.1 Why Does Add-on Installation Fail and Prompt "The release name is already exist"?

#### Symptom

When an add-on fails to be installed, the error message "The release name is already exist" is returned.

#### Possible Cause

The add-on release record remains in the Kubernetes cluster. Generally, it is because the cluster etcd has backed up and restored the add-on, or the add-on fails to be installed or deleted.

#### Solution

Use kubectl to connect to the cluster and manually clear the Secret and Configmap corresponding to add-on release. The following uses autoscaler add-on release as an example.

- Step 1** Connect to the cluster using `kubectl`, and run the following command to view the Secret list of add-on releases:

**kubectl get secret -nkube-system |grep cceaddon**

```
[root@cce-123-vpc-node2 ~]# kubectl get secret -nkube-system |grep cceaddon
sh.helm.release.v1.cceaddon-autoscaler.v1 helm.sh/release.v1 1 61s
sh.helm.release.v1.cceaddon-autoscaler.v2 helm.sh/release.v1 1 47s
sh.helm.release.v1.cceaddon-coredns.v1 helm.sh/release.v1 1 6h2m
sh.helm.release.v1.cceaddon-everest.v1 helm.sh/release.v1 1 6h2m
[root@cce-123-vpc-node2 ~]#
```

The Secret name of an add-on release is in the format of **sh.helm.release.v1.cceaddon-*{add-on name}*.v\***. If there are multiple release versions, you can delete their Secrets at the same time.

- Step 2** Run the **release secret** command to delete the Secrets.

Example:

**kubectl delete secret sh.helm.release.v1.cceaddon-autoscaler.v1  
sh.helm.release.v1.cceaddon-autoscaler.v2 -nkube-system**

```
[root@cce-123-vpc-node2 ~]# kubectl delete secret sh.helm.release.v1.cceaddon-autoscaler.v1 sh.helm.release.v1.cceaddon-autoscaler.v2 -nkube-system
secret "sh.helm.release.v1.cceaddon-autoscaler.v1" deleted
secret "sh.helm.release.v1.cceaddon-autoscaler.v2" deleted
[root@cce-123-vpc-node2 ~]#
```

- Step 3** If the add-on is created when Helm v2 is used, CCE automatically bumps the v2 release in Configmaps to v3 release in Secrets when viewing the add-ons and their details. The v2 release in the original Configmap is not deleted. Run the following command to view the ConfigMap list of add-on releases:

**kubectl get configmap -nkube-system | grep cceaddon**

```
cluster-autoscaler-th-config 1 7d10h
[paas@192-168-0-64 ~]$ kubectl get configmap -nkube-system | grep cceaddon
cceaddon-autoscaler.v1 1 7d10h
cceaddon-autoscaler.v2 1 52m
cceaddon-coredns.v1 1 14d
cceaddon-everest.v1 1 14d
[paas@192-168-0-64 ~]$
```

The ConfigMap name of an add-on release is in the format of **cceaddon-*{add-on name}*.v\***. If there are multiple release versions, you can delete their ConfigMaps at the same time.

- Step 4** Run the **release configmap** command to delete the ConfigMaps.

Example:

**kubectl delete configmap cceaddon-autoscaler.v1 cceaddon-autoscaler.v2 -nkube-system**

```
[paas@192-168-0-64 ~]$ kubectl delete configmap cceaddon-autoscaler.v1 cceaddon-autoscaler.v2 -nkube-system
configmap "cceaddon-autoscaler.v1" deleted
configmap "cceaddon-autoscaler.v2" deleted
[paas@192-168-0-64 ~]$
```

**⚠ CAUTION**

Deleting resources in kube-system is a high-risk operation. Ensure that the command is correct before running it to prevent resources from being deleted by mistake.



- Step 5** On the CCE console, install add-on and then uninstall it. Ensure that the residual add-on resources are cleared. After the uninstall is complete, install the add-on again.

 **NOTE**

When installing the add-on for the first time, you may find it abnormal after the installation due to the residual resources of the previous add-on release, which is normal. In this case, you can uninstall the add-on on the console to ensure that the residual resources are cleared and the add-on can run properly after being installed again.

----End

## 20.11 API & kubectl FAQs

### 20.11.1 How Can I Access a Cluster API Server?

You can use either of the following methods to access the cluster API Server:

- (Recommended) Through the cluster API. This access mode uses certificate authentication. It is suitable for API calls on scale thanks to its direct connection to the API Server. This is a recommended option.
- API Gateway. This access mode uses token authentication. You need to obtain a token using your account. This access mode applies to small-scale API calls. API gateway flow control may be triggered when APIs are called on scale.

### 20.11.2 Can the Resources Created Using APIs or kubectl Be Displayed on the CCE Console?

The CCE console does not support the display of the following Kubernetes resources: DaemonSets, ReplicationControllers, ReplicaSets, and endpoints.

To query these resources, run the kubectl commands.

In addition, Deployments, StatefulSets, Services, and pods can be displayed on the console only when the following conditions are met:

- Deployments and StatefulSets: At least one label uses **app** as its key.
- Pods: Pods are displayed on the **Pods** tab page in the workload details only after a Deployment or StatefulSet has been created.
- Services: Services are displayed on the **Access Mode** tab page in the Deployment or StatefulSet details.

The Services displayed on this tab page are associated with the workload.

- a. At least one label of the workload uses **app** as its key.
- b. The label of a Service is the same as that of the workload.

### 20.11.3 How Do I Download kubeconfig for Connecting to a Cluster Using kubectl?

- Step 1** Log in to the CCE console. Click the target cluster to go to its details page.

**Step 2** In the **Connection Information** area, view the kubectl connection mode.

**Step 3** In the window that is displayed, download the kubectl configuration file (**kubeconfig.json**).

----End

## 20.11.4 How Do I Rectify the Error Reported When Running the kubectl top node Command?

### Symptom

The error message "Error from server (ServiceUnavailable): the server is currently unable to handle the request (get nodes.metrics.k8s.io)" is displayed after the **kubectl top node** command is executed.

### Possible Causes

"Error from server (ServiceUnavailable)" indicates that the cluster is not connected. In this case, you need to check whether the network between kubectl and the master node in the cluster is normal.

### Solution

- If the kubectl command is executed outside the cluster, check whether the cluster is bound to an EIP. If yes, download the **kubeconfig** file and run the kubectl command again.
- If the kubectl command is executed on a node in the cluster, check the security group of the node and check whether the TCP/UDP communication between the worker node and master node is allowed. For details about the security group, see [Configuring Cluster Security Group Rules](#).

## 20.11.5 Why Is "Error from server (Forbidden)" Displayed When I Use kubectl?

### Symptom

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"
```

### Possible Cause

This user has no permissions to operate Kubernetes resources.

### Solution

Assign permissions to the user.

- 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.

- Step 5** Click **OK**.

----End

## 20.12 DNS FAQs

### 20.12.1 What Should I Do If Domain Name Resolution Fails?

#### Check Item 1: Whether the coredns Add-on Has Been Installed

- Step 1** Log in to the CCE console and click the cluster.
- Step 2** In the navigation pane, choose **Add-ons**. In the **Add-ons Installed** area, check whether the coredns add-on has been installed.
- Step 3** If not, install the add-on. For details, see [Why Does a Container in a CCE Cluster Fail to Perform DNS Resolution?](#)

----End

#### Check Item 2: Whether the coredns Instance Reaches the Performance Limit

CoreDNS QPS is positively correlated with the CPU usage. If the QPS is high, adjust the the coredns instance specifications based on the QPS.

- Step 1** Log in to the CCE console and access the cluster console.
- Step 2** In the navigation pane on the left, choose **Add-ons**. In **Add-ons Installed**, find the coredns add-on corresponding to the cluster and ensure that the add-on status is **Running**.
- Step 3** Click the coredns add-on name to view the add-on list.
- Step 4** Click **Monitor** of the the coredns add-on to view the CPU and memory usage.

If the add-on performance reaches the bottleneck, adjust the coredns add-on specifications.

----End

### Check Item 3: Whether the External Domain Name Resolution Is Slow or Times Out

If the domain name resolution failure rate is lower than 1/10000, optimize parameters by referring to [How Do I Optimize the Configuration If the External Domain Name Resolution Is Slow or Times Out?](#) or add a retry policy in the service.

### Check Item 4: Whether UnknownHostException Occurs

When service requests in the cluster are sent to an external DNS server, a domain name resolution error occurs due to occasional UnknownHostException. UnknownHostException is a common exception. When this exception occurs, check whether there is any domain name-related error or whether you have entered a correct domain name.

To locate the fault, perform the following steps:

- Step 1** Check the host name carefully (spelling and extra spaces).
- Step 2** Check the DNS settings. Before running the application, run the **ping hostname** command to ensure that the DNS server has been started and running. If the host name is new, you need to wait for a period of time before the DNS server is accessed.
- Step 3** Check the CPU and memory usage of the coredns add-on to determine whether the performance bottleneck has been reached. For details, see [Check Item 2: Whether the coredns Instance Reaches the Performance Limit.](#)
- Step 4** Check whether traffic limiting is performed on the coredns add-on. If traffic limiting is triggered, the processing time of some requests may be prolonged. In this case, you need to adjust the coredns add-on specifications.

Log in to the node where the coredns add-on is installed and view the following content:

```
cat /sys/fs/cgroup/cpu/kubepods/pod<pod_uid>/<coredns container ID>/cpu.stat
```

- *<pod uid>* indicates the pod UID of the coredns add-on, which can be obtained by running the following command:

```
kubectl get po <pod name> -nkube-system -ojsonpath='{.metadata.uid}'
```

In the preceding command, *<pod name>* indicates the name of the coredns add-on running on the current node.

- *<coredns container ID>* must be a complete container ID, which can be obtained by running the following command:

```
docker ps --no-trunc | grep k8s_coredns | awk '{print $1}'
```

Example:

```
cat /sys/fs/cgroup/cpu/kubepods/
pod27f58662-3979-448e-8f57-09b62bd24ea6/6aa98c323f43d689ac47190bc84cf4fadd23bd8dd25307f773df2
5003ef0eef0/cpu.stat
```

Pay attention to the following metrics:

- **nr\_throttled**: number of times that traffic is limited.
- **throttled\_time**: total duration of traffic limiting, in nanoseconds.

----End

If the host name and DNS settings are correct, you can use the following optimization policies.

#### Optimization policies:

1. Change the `coredns` cache time.
2. Configure the stub domain.
3. Modify the value of `ndots`.

#### NOTE

- **Increasing the cache time of `coredns`** helps resolve the same domain name for the N time, reducing the number of cascading DNS requests.
- **Configuring the stub domain** can reduce the number of DNS request links.

#### How to modify:

1. Modifying the `coredns` cache time and configuring the stub domain:  
Restart the `coredns` add-on after you modify the configurations.
2. Modifying `ndots`:

#### [How Do I Optimize the Configuration If the External Domain Name Resolution Is Slow or Times Out?](#)

Example:

```
dnsConfig:
 options:
 - name: timeout
 value: '2'
 - name: ndots
 value: '5'
 - name: single-request-reopen
```

You are advised to change the value of `ndots` to 2.

## 20.12.2 Why Does a Container in a CCE Cluster Fail to Perform DNS Resolution?

### Symptom

A customer bound its domain name to the private domain names in the DNS service and also to a specific VPC. It is found that the ECSs in the VPC can properly resolve the private domain name but the containers in the VPC cannot.

### Application Scenario

Containers in a VPC cannot resolve domain names.

## Solution

According to the resolution rules of private domain names, the subnet DNS in the VPC must be set to the cloud DNS. You can find the details of the private network DNS service on its console.

The customer can perform domain name resolution on the ECSs in the VPC subnet, which indicates that the preceding configuration has been completed in the subnet.

```
bash-4.4# exit
exit
[root@global-skyworth1-vpn ~]# ping [REDACTED]
PING [REDACTED] (10.247.11.29) 56(84) bytes of data.

^C
--- ota.skyworth.web ping statistics ---
```

However, when the domain name resolution is performed in a container, the message "bad address" is displayed, indicating that the domain name cannot be resolved.

```
[root@global-skyworth1-vpn ~]#
[root@global-skyworth1-vpn ~]# docker exec -it 86cf062a5ba3 bash
bash-4.4# ping [REDACTED]
ping: bad address '[REDACTED]'
bash-4.4#
```

Log in to the CCE console and check the add-ons installed in the cluster.

If you find that the coredns add-on does not exist in **Add-ons Installed**, the coredns add-on may have been incorrectly uninstalled.

Install it and add the corresponding domain name and DNS service address to resolve the domain name.

### 20.12.3 How Do I Optimize the Configuration If the External Domain Name Resolution Is Slow or Times Out?

The following is an example `resolv.conf` file for a container in a workload:

```
root@test-5dffdddf95-vpt4m:/# cat /etc/resolv.conf
nameserver 10.247.3.10
search istio.svc.cluster.local svc.cluster.local cluster.local
options ndots:5 single-request-reopen timeout:2
```

In the preceding information:

- **nameserver**: IP address of the DNS. Set this parameter to the cluster IP address of CoreDNS.
- **search**: domain name search list, which is a common suffix of Kubernetes.
- **ndots**: If the number of dots (.) is less than the domain name, **search** is preferentially used for resolution.
- **timeout**: timeout interval.

- **single-request-reopen**: indicates that different source ports are used to send different types of requests.

By default, when you create a workload on the CCE console, the preceding parameters are configured as follows:

```
dnsConfig:
 options:
 - name: timeout
 value: '2'
 - name: ndots
 value: '5'
 - name: single-request-reopen
```

These parameters can be optimized or modified based on service requirements.

## Scenario 1: Slow External Domain Name Resolution

Optimization Solution

1. If the workload does not need to access the Kubernetes Service in the cluster, see [How Do I Configure a DNS Policy for a Container?](#).
2. If the number of dots (.) in the domain name used by the working Service to access other Kubernetes Services is less than 2, set **ndots** to 2.

## Scenario 2: External Domain Name Resolution Timeout

Optimization Solution

1. Generally, the timeout of a Service must be greater than the value of **timeout** multiplied by **attempts**.
2. If it takes more than 2s to resolve the domain name, you can set **timeout** to a larger value.

## 20.12.4 How Do I Configure a DNS Policy for a Container?

CCE uses **dnsPolicy** to identify different DNS policies for each pod. The value of **dnsPolicy** can be either of the following:

- **None**: No DNS policy is configured. In this mode, you can customize the DNS configuration, and **dnsPolicy** needs to be used together with **dnsConfig** to customize the DNS.
- **Default**: The pod inherits the name resolution configuration from the node where the pod is running. The container's DNS configuration file is the DNS configuration file that the kubelet's **--resolv-conf** flag points to. In this case, a cloud DNS is used for CCE clusters.
- **ClusterFirst**: In this mode, the DNS in the pod uses the DNS service configured in the cluster. That is, the kube-dns or CoreDNS service in the Kubernetes is used for domain name resolution. If the resolution fails, the DNS configuration of the host machine is used for resolution.

If the type of **dnsPolicy** is not specified, **ClusterFirst** is used by default.

- If the type of **dnsPolicy** is set to **Default**, the name resolution configuration is inherited from the worker node where the pod is running.
- If the type of **dnsPolicy** is set to **ClusterFirst**, DNS queries will be sent to the kube-dns service.

The kube-dns service responds to queries on the domains that use the configured cluster domain suffix as the root. All other queries (for example, [www.kubernetes.io](http://www.kubernetes.io)) are forwarded to the upstream name server inherited from the node. Before this feature was supported, stub domains were typically introduced by a custom resolver, instead of the upstream DNS. However, this causes the custom resolver itself to be the key path to DNS resolution, where scalability and availability issues can make the DNS functions unavailable to the cluster. This feature allows you to introduce custom resolvers without taking over the entire resolution path.

If a workload does not need to use CoreDNS in the cluster, you can use `kubectl` or call the APIs to set the `dnsPolicy` to `Default`.

## 20.13 Image Repository FAQs

### 20.13.1 How Do I Upload My Images to CCE?

SoftWare Repository for Container (SWR) manages images for CCE. It provides the following ways to upload images:

- [Uploading an Image Through the Client](#)
- [Uploading an Image Through the SWR Console](#)

## 20.14 Permissions

### 20.14.1 Can I Configure Only Namespace Permissions Without Cluster Management Permissions?

Namespace permissions and cluster management permissions are independent and complementary to each other.

- Namespace permissions: apply to clusters and are used to manage operations on cluster resources (such as creating workloads).
- Cluster management (IAM) permissions: apply to cloud services and used to manage CCE clusters and peripheral resources (such as VPC, ELB, and ECS).

Administrators of the IAM Admin user group can grant cluster management permissions (such as CCE Administrator and CCE FullAccess) to IAM users or grant namespace permissions on a cluster on the CCE console. However, the permissions you have on the CCE console are determined by the IAM system policy. If the cluster management permissions are not configured, you do not have the permissions for accessing the CCE console.

If you only run `kubectl` commands to work on cluster resources, you only need to obtain the `kubeconfig` file with the namespace permissions. For details, see [Can I Use kubectl If the Cluster Management Permissions Are Not Configured?](#). Note that information leakage may occur when you use the `kubeconfig` file.



## 20.14.2 Can I Use CCE APIs If the Cluster Management Permissions Are Not Configured?

CCE has cloud service APIs and cluster APIs.

- **Cloud service APIs:** You can perform operations on the infrastructure (such as creating nodes) and cluster resources (such as creating workloads).  
When using cloud service APIs, the cluster management (IAM) permissions must be configured.
- **Cluster APIs:** You can perform operations on cluster resources (such as creating workloads) through the Kubernetes native API server, but not on cloud infrastructure resources (such as creating nodes).  
When using cluster APIs, you only need to add the cluster certificate. Only the users with the cluster management (IAM) permissions can download the cluster certificate. Note that information leakage may occur during certificate transmission.

## 20.14.3 Can I Use kubectl If the Cluster Management Permissions Are Not Configured?

IAM authentication is not required for running kubectl commands. Therefore, you can run kubectl commands without configuring cluster management (IAM) permissions. However, you need to obtain the kubectl configuration file (kubeconfig) with the namespace permissions. In the following scenarios, information leakage may occur during file transmission.

- **Scenario 1**  
If an IAM user has been configured with the cluster management permissions and namespace permissions, downloads the kubeconfig authentication file and then deletes the cluster management permissions (reserving the namespace permissions), kubectl can still be used to perform operations on Kubernetes clusters. Therefore, if you want to permanently delete the permission of a user, you must also delete the cluster management permissions and namespace permissions of the user.
- **Scenario 2**  
An IAM user has certain cluster management and namespace permissions and downloads the kubeconfig authentication file. In this case, CCE determines which Kubernetes resources can be accessed by kubectl based on the user information. That is, the authentication information of a user is recorded in kubeconfig. Anyone can use kubeconfig to access the cluster.

## 20.15 Reference

## 20.15.1 How Do I Expand the Storage Capacity of a Container?

### Scenario

The default storage size of a container is 10 GB. If a large volume of data is generated in the container, expand the capacity using the method described in this topic.

### Solution

**Step 1** Log in to the CCE console and click the name of the target cluster in the cluster list.

**Step 2** Choose **Nodes** from the navigation pane.

**Step 3** Select the target node and choose **More > Reset Node** in the **Operation** column.

---

#### NOTICE

Resetting a node may make unavailable the node-specific resources (such as local storage and workloads scheduled to this node). Exercise caution when performing this operation to avoid impact on running services.

---

**Step 4** Click **Yes**.

**Step 5** Reconfigure node parameters.

If you need to adjust the container storage space, pay attention to the following configurations:

**Storage Settings:** Click **Expand** next to the data disk to set the following parameters:

- **Allocate Disk Space:** storage space used by the container engine to store the Docker/containerd working directory, container image data, and image metadata. Defaults to 90% of the data disk.
- **Allocate Pod Basesize:** CCE allows you to 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 be smaller than or equal to 80% of the container engine space.

 NOTE

- The capability of customizing pod basesize is related to the node OS and container storage roots.
  - When the rootfs uses Device Mapper, the node supports custom pod basesize. The default storage space of a single container is 10 GiB.
  - When the rootfs uses OverlayFS, most nodes do not support custom pod basesize. The storage space of a single container is not limited and defaults to the container engine space.  
Only EulerOS 2.9 nodes in clusters of 1.19.16, 1.21.3, 1.23.3, and later versions support custom pod basesize.
- In the case of using Docker on EulerOS 2.9 nodes, **basesize** will not take effect if **CAP\_SYS\_RESOURCE** or **privileged** is configured for a container.

**Step 6** After the node is reset, log in to the node and run the following command to access the container and check whether the container storage capacity has been expanded:

```
docker exec -it container_id /bin/sh or kubectl exec -it container_id /bin/sh
df -h
```

```
df -h
Filesystem Size Used Avail Use% Mounted on
/dev/mapper/docker-253:1-787293-631c1bde2cbe82e39f32253b216ba914cb183b168b54780b3e5b9a54ee40a8d1 15G 229M 15G 2% /
tmpfs 32G 0 32G 0% /dev
tmpfs 32G 0 32G 0% /sys/fs/cgroup
/dev/mapper/vgpaas-kubernetes 9.8G 37M 9.2G 1% /etc/hosts
/dev/vda1 48G 5.2G 43G 14% /etc/hostname
shm 64M 0 64M 0% /dev/shm
tmpfs 32G 16K 32G 1% /run/secrets/kubernetes.io/serviceaccount
tmpfs 32G 0 32G 0% /proc/acpi
tmpfs 32G 0 32G 0% /sys/firmware
tmpfs 32G 0 32G 0% /proc/scsi
tmpfs 32G 0 32G 0% /proc/kbox
tmpfs 32G 0 32G 0% /proc/oom_extend
```

----End

## 20.15.2 How Can Container IP Addresses Survive a Container Restart?

### If Containers Will Run in a Single-Node Cluster

Add **hostNetwork: true** to the **spec.spec**. in the YAML file of the workload to which the containers will belong.

### If Containers Will Run in a Multi-Node Cluster

Configure node affinity policies, in addition to perform the operations described in "If the Container Runs in a Single-Node Cluster". However, after the workload is created, the number of running pods cannot exceed the number of affinity nodes.

### Expected Result

After the previous settings are complete and the workload is running, the IP addresses of the workload's pods are the same as the node IP addresses. After the workload is restarted, these IP addresses will keep unchanged.

# 21 Best Practices

## 21.1 Checklist for Deploying Containerized Applications in the Cloud

### Overview

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

### Check Items

**Table 21-1** System availability

Category	Check Item	Type	Impact
Cluster	Before creating a cluster, properly plan the node network and container network based on service requirements to allow subsequent service expansion.	Network planning	If the subnet or container CIDR block where the cluster resides is small, the number of available nodes supported by the cluster may be less than required.
	Before creating a cluster, properly plan CIDR blocks for the related Direct Connect, peering connection, container network, service network, and subnet to avoid IP address conflicts.	Network planning	If CIDR blocks are not properly set and IP address conflicts occur, service access will be affected.

Category	Check Item	Type	Impact
	When a cluster is created, the default security group is automatically created and bound to the cluster. You can set custom security group rules based on service requirements.	Deployment	Security groups are key to security isolation. Improper security policy configuration may cause security risks and service connectivity problems.
	Enable the multi-master node mode, and set the number of master nodes to <b>3</b> when creating a cluster.	Reliability	After the multi-master node mode is enabled, three master nodes will be created. If a master node is faulty, the cluster can still be available without affecting service functions. In commercial scenarios, it is advised to enable the multi-master node mode.
	When creating a cluster, select a proper network model, such as container tunnel network or VPC network.	Deployment	After a cluster is created, the network model cannot be changed. Exercise caution when selecting a network model.
Workload	When creating a workload, set the CPU and memory limits to improve service robustness.	Deployment	When multiple applications are deployed on the same node, if the upper and lower resource limits are not set for an application, resource leakage occurs. As a result, resources cannot be allocated to other applications, and the application monitoring information will be inaccurate.
	When creating a workload, you can set probes for container health check, including <b>liveness probe</b> and <b>readiness probe</b> .	Reliability	If the health check function is not configured, a pod cannot detect service exceptions or automatically restart the service to restore it. This results in a situation where the pod status is normal but the service in the pod is abnormal.

Category	Check Item	Type	Impact
	When creating a workload, select a proper access mode (Service). Currently, the following types of Services are supported: ClusterIP, NodePort, and LoadBalancer.	Deployment	Improper Service configuration may cause logic confusion for internal and external access and resource waste.
	When creating a workload, do not set the number of replicas for a single pod. Set a proper node scheduling policy based on your service requirements.	Reliability	For example, if the number of replicas of a single pod is set, the service will be abnormal when the node or pod is abnormal. To ensure that your pods can be successfully scheduled, ensure that the node has idle resources for container scheduling after you set the scheduling rule.
	Properly set affinity and anti-affinity.	Reliability	If affinity and anti-affinity are both configured for an application that provides Services externally, Services may fail to be accessed after the application is upgraded or restarted.
	When creating a workload, set the pre-stop processing command ( <b>Lifecycle &gt; Pre-Stop</b> ) to ensure that the services running in the pods can be completed in advance in the case of application upgrade or pod deletion.	Reliability	If the pre-stop processing command is not configured, the pod will be directly killed and services will be interrupted during application upgrade.

**Table 21-2** Data reliability

Category	Check Item	Type	Impact
Container data persistence	Select a proper data volume type based on service requirements.	Reliability	When a node is faulty and cannot be recovered, data in the local disk cannot be recovered. Therefore, you are advised to use cloud storage volumes to ensure data reliability.

Category	Check Item	Type	Impact
Backup	Back up application data.	Reliability	Data cannot be restored after being lost.

**Table 21-3** O&M reliability

Category	Check Item	Type	Impact
Project	The quotas of ECS, VPC, subnet, EIP, and EVS resources must meet customer requirements.	Deployment	If the quota is insufficient, resources will fail to be created. Specifically, users who have configured auto scaling must have sufficient resource quotas.
	You are not advised to modify kernel parameters, system configurations, cluster core component versions, security groups, and ELB-related parameters on cluster nodes, or install software that has not been verified.	Deployment	Exceptions may occur on CCE clusters or Kubernetes components on the node, making the node unavailable for application deployment.
	Do not modify information about resources created by CCE, such as security groups and EVS disks. Resources created by CCE are labeled <b>cce</b> .	Deployment	CCE cluster functions may be abnormal.

Category	Check Item	Type	Impact
Proactive O&M	<p>CCE provides multi-dimensional monitoring and alarm reporting functions, allowing users to locate and rectify faults as soon as possible.</p> <ul style="list-style-type: none"> <li>• Application Operations Management (AOM): The default basic resource monitoring of CCE covers detailed container-related metrics and provides alarm reporting functions.</li> <li>• Open source Prometheus: A monitoring tool for cloud native applications. It integrates an independent alarm system to provide more flexible monitoring and alarm reporting functions.</li> </ul>	Monitoring	If the alarms are not configured, the standard of container cluster performance cannot be established. When an exception occurs, you cannot receive alarms and will need to manually locate the fault.

## 21.2 Containerization

### 21.2.1 Containerizing an Enterprise Application (ERP)

#### 21.2.1.1 Solution Overview

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

#### What Is a Container?

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

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

#### Why Is a Container Preferred?

- More efficient use of system resources



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

- **Faster startup**

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

- **Consistent runtime environment**

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

- **Easier application migration, maintenance, and scaling**

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

## Containerization Modes

The following modes are available for containerizing applications:

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

**Table 21-4** lists the advantages and disadvantages of the three modes.

**Table 21-4** Containerization modes

Containerization Mode	Advantage	Disadvantage
<p>Method 1: Containerize a single application as a whole.</p>	<ul style="list-style-type: none"> <li>• Zero modification on services: The application architecture and code require no change.</li> <li>• The deployment and upgrade efficiency is improved. Applications can be packed as container images to ensure application environment consistency and improve deployment efficiency.</li> <li>• Reduce resource costs: Containers use system resources more efficiently. Compared with a VM with the same configurations, a container can run more applications.</li> </ul>	<ul style="list-style-type: none"> <li>• Difficult to expand the entire architecture of an application. As the code size increases, code update and maintenance would be complicated.</li> <li>• Difficult to launch new functions, languages, frameworks, and technologies.</li> </ul>

Containerization Mode	Advantage	Disadvantage
<p>Method 2: Containerize first the application components that are frequently updated or have high requirements on auto scaling.</p>	<ul style="list-style-type: none"> <li>• Progressive transformation: Reconstructing the entire architecture involves a heavy workload. This mode containerizes only a part of components, which is easy to accept for customers.</li> <li>• Flexible scaling: Application components that have high requirements on auto scaling are containerized. When the application needs to be scaled, you only need to scale the containers, which is flexible and reduces the required system resources.</li> <li>• Faster rollout of new features: Application components that are frequently upgraded are containerized. In subsequent upgrades, only these containers need to be upgraded. This shortens the time to market (TTM) of new features.</li> </ul>	<p>Need to decouple some services.</p>

Containerization Mode	Advantage	Disadvantage
<p>Method 3: Transform an application to microservices and then containerize the microservices one by one.</p>	<ul style="list-style-type: none"> <li>• Independent scaling: After an application is split into microservices, you can independently increase or decrease the number of instances for each microservice.</li> <li>• Increased development speed: Microservices are decoupled from one another. Code development of a microservice does not affect other microservices.</li> <li>• Security assurance through isolation: For an overall application, if a security vulnerability exists, attackers can use this vulnerability to obtain the permission to all functions of the application. However, in a microservice architecture, if a service is attacked, attackers can only obtain the access permission to this service, but cannot intrude other services.</li> <li>• Breakdown isolation: If one microservice breaks down, other microservices can still run properly.</li> </ul>	<p>Need to transform the application to microservices, which involves a large number of changes.</p>

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

## 21.2.1.2 Procedure

### 21.2.1.2.1 Containerizing an Entire Application

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

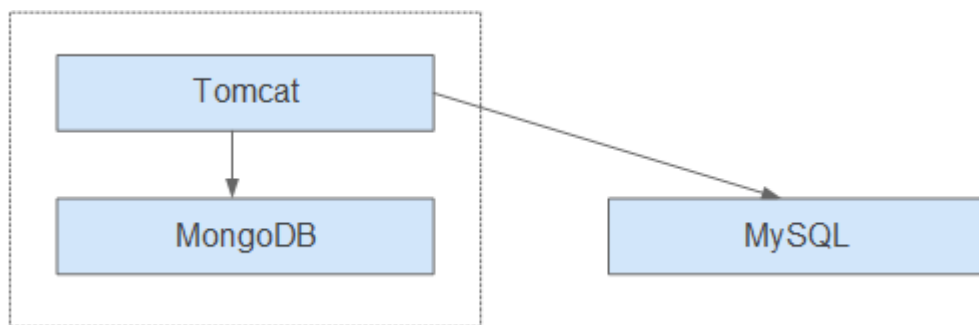
No recoding or re-architecting is required. You only need to pack the entire application into a container image and deploy the container image on CCE.

## Introduction

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

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

**Figure 21-1** Application architecture



As shown in [Figure 21-1](#), the application is a standard Tomcat application, and its backend interconnects with MongoDB and MySQL databases. For this type of applications, there is no need to split the architecture. The entire application is built as an image, and the MongoDB database is deployed in the same image as the Tomcat application. In this way, the application can be deployed or upgraded through the image.

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

## Benefits

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

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

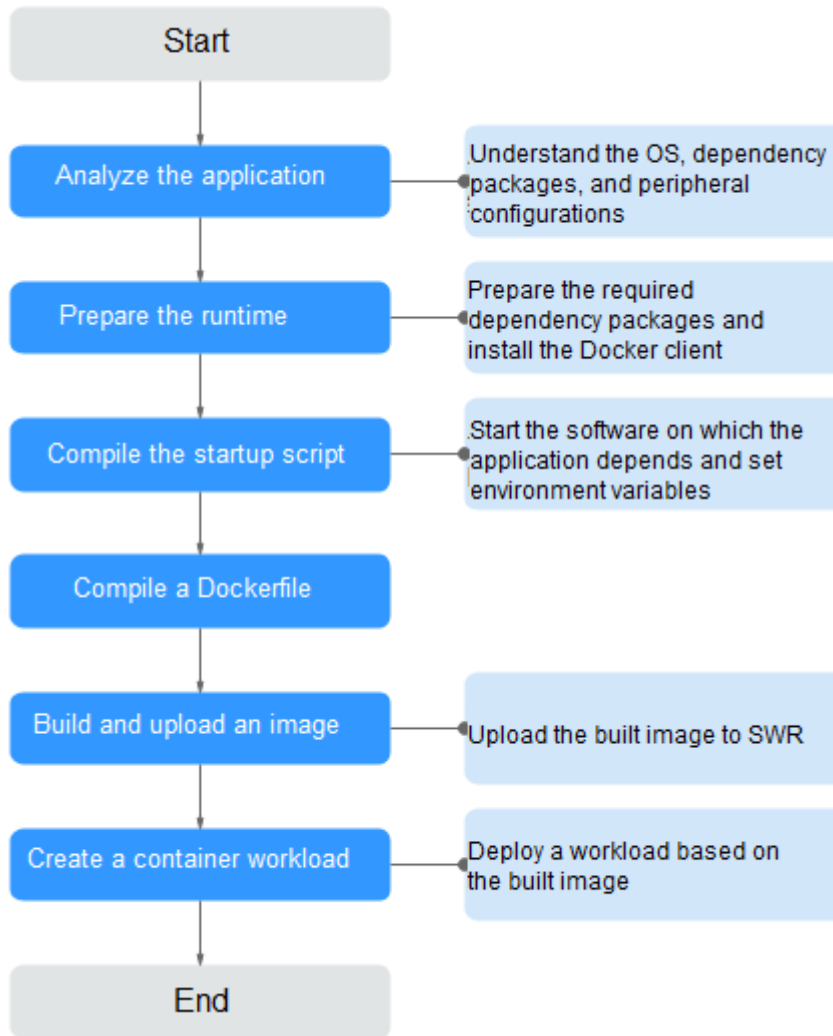
**Table 21-5** Comparison between the two deployment modes

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

### 21.2.1.2.2 Containerization Process

The following figure illustrates the process of containerizing an application.

**Figure 21-2** Process of containerizing an application



### 21.2.1.2.3 Analyzing the Application

Before containerizing an application, analyze the running environment and dependencies of the application, and get familiar with the application deployment mode. For details, see [Table 21-6](#).

**Table 21-6** Application environment

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

Category	Sub-category	Description
	Runtime environment	<p>The Java application requires Java Development Kit (JDK), the Go language requires GoLang, the web application requires Tomcat environment, and the corresponding version number needs to be confirmed.</p> <p>In this example, the web application of the Tomcat type is used. This application requires the runtime environment of Tomcat 7.0, and Tomcat requires JDK 1.8.</p>
	Dependency package	<p>Understand required dependency packages, such as OpenSSL and other system software, and their version numbers.</p> <p>In this example, no dependency package is required.</p>
Deployment mode	Peripheral configurations	<p>MongoDB database: In this example, the MongoDB database and Tomcat application are deployed on the same server. Therefore, their configurations can be fixed and there is no need to extract their configurations.</p>
		<p>External services with which the application needs to interconnect, such as databases and file systems.</p> <p>These configurations need to be manually configured each time you deploy an application on a VM. However, through containerized deployment, environment variables can be injected into a container, facilitating deployment.</p> <p>In this example, the application needs to interconnect with the MySQL database. Obtain the database configuration file. The server address, database name, database login username, and database login password are injected through environment variables.</p> <pre>url=jdbc:mysql://Server address/Database name #Database connection URL username=**** #Username for logging in to the database password=**** #Password for logging in to the database</pre>
	Application configurations	<p>Sort out the configuration parameters, such as configurations that need to be modified frequently and those remain unchanged during the running of the application.</p> <p>In this example, no application configurations need to be extracted.</p> <p><b>NOTE</b></p> <p>To avoid frequent image replacement, you are advised to classify configurations of the application.</p> <ul style="list-style-type: none"> <li>For the configurations (such as peripheral interconnection information and log levels) that are frequently changed, you are advised to configure them as environment variables.</li> <li>For the configurations that remain unchanged, directly write them into images.</li> </ul>



### 21.2.1.2.4 Preparing the Application Runtime

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

- **Installing Docker:** During application containerization, build a container image. To do so, you have to prepare a PC and install Docker on it.
- **Obtaining the base image tag:** Determine the base image based on the OS on which the application runs. In this example, the application runs on CentOS 7.1 and the base image can be obtained from an open-source image repository.
- **Obtaining the runtime:** Obtain the runtime of the application and the MongoDB database with which the application interconnects.

## Installing Docker

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

### NOTE

SWR uses Docker 1.11.2 or later to upload images.

You are advised to install Docker and build images as user **root**. Obtain the password of user **root** of the host where Docker is to be installed in advance.

**Step 1** Log in as user **root** to the device on which Docker is about to be installed.

**Step 2** Quickly install Docker on the device running Linux. You can also manually install Docker. For details, see [Docker Engine installation](#).

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

```
sh get-docker.sh
```

**Step 3** Run the following command to query the Docker version:

```
docker version
```

```
Client:
Version: 17.12.0-ce
API Version:1.35
...
```

**Version** indicates the version number.

----End

## Obtaining the Base Image Tag

Determine the base image based on the OS on which the application runs. In this example, the application runs on CentOS 7.1 and the base image can be obtained from an open-source image repository.

### NOTE

Search for the image tag based on the OS on which the application runs.

**Step 1** Visit the Docker website.

**Step 2** Search for CentOS. The image corresponding to CentOS 7.1 is **centos7.1.1503**. Use this image name when editing the Dockerfile.

**Figure 21-3** Obtaining the CentOS version



----End

## Obtaining the Runtime

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

### NOTE

Download the environment required by the application.

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

1. Download JDK 1.8.

Download address: <https://www.oracle.com/java/technologies/jdk8-downloads.html>.

2. Download Tomcat 7.0 from <http://archive.apache.org/dist/tomcat/tomcat-7/v7.0.82/bin/apache-tomcat-7.0.82.tar.gz>.
3. Download MongoDB 3.2 from [https://fastdl.mongodb.org/linux/mongodb-linux-x86\\_64-rhel70-3.2.9.tgz](https://fastdl.mongodb.org/linux/mongodb-linux-x86_64-rhel70-3.2.9.tgz).

**Step 2** Log in as user **root** to the device running Docker.

**Step 3** Run the following commands to create the directory where the application is to be stored: For example, set the directory to **apptest**.

```
mkdir apptest
```

```
cd apptest
```

**Step 4** Use Xshell to save the downloaded dependency files to the **apptest** directory.

**Step 5** Run the following commands to decompress the dependency files:

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

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

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

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

 **NOTE**

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

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

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

```
cd apache-tomcat-7.0.82/webapps/apptest
```

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

```
rm -rf apptest.war
```

```
----End
```

### 21.2.1.2.5 Compiling a Startup Script

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

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

 **NOTE**

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

## Procedure

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

```
mkdir apptest
```

```
cd apptest
```

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

### vi start\_tomcat\_and\_mongo.sh

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

----End

### 21.2.1.2.6 Compiling the Dockerfile

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

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

This section describes how to compile a Dockerfile file.

#### NOTE

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

## Procedure

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

### vi Dockerfile

The content is as follows:

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

Copy the files in the apache-tomcat-7.0.82 directory to the container path.
COPY ./apache-tomcat-7.0.82 /root/apache-tomcat-7.0.82
Copy the files in the jdk1.8.0_151 directory to the container path.
COPY ./jdk1.8.0_151 /root/jdk1.8.0_151
Copy the files in the mongodb-linux-x86_64-rhel70-3.2.9 directory to the container path.
COPY ./mongodb-linux-x86_64-rhel70-3.2.9/bin /usr/local/mongodb/bin
Copy start_tomcat_and_mongo.sh to the /root directory of the container.
COPY ./start_tomcat_and_mongo.sh /root/

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

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

In the preceding information:

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

----End

### 21.2.1.2.7 Building and Uploading an Image

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

#### NOTE

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

## Required Cloud Services

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

## Basic Concepts

- **Image:** A Docker image is a special file system that includes everything needed to run containers: programs, libraries, resources, settings, and so on. It

also includes corresponding configuration parameters (such as anonymous volumes, environment variables, and users) required within a container runtime. An image does not contain any dynamic data, and its content remains unchanged after being built.

- Container: Images become containers at runtime, that is, containers are created from images. A container can be created, started, stopped, deleted, or suspended.

## Procedure

**Step 1** Log in as the **root** user to the device running Docker.

**Step 2** Enter the **apptest** directory.

```
cd apptest
```

```
ll
```

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

```
root@ecs-aos:~/apptest# ll
total 264456
drwxr-xr-x 5 root root 4096 Jan 2 19:59 ./
drwx----- 6 root root 4096 Jan 2 19:59 ../
drwxr-xr-x 9 root root 4096 Jan 2 19:55 apache-tomcat-7.0.82/
-rw-r--r-- 1 root root 8997403 Jan 2 19:52 apache-tomcat-7.0.82.tar.gz
-rw-r--r-- 1 root root 599 Jan 2 19:59 Dockerfile
drwxr-xr-x 8 uucp 143 4096 Sep 6 10:32 jdk1.8.0_151/
-rw-r--r-- 1 root root 189736377 Jan 2 19:54 jdk-8u151-linux-x64.tar.gz
drwxr-xr-x 3 root root 4096 Jan 2 19:55 mongodb-linux-x86_64-rhel70-3.2.9/
-rw-r--r-- 1 root root 72035914 Jan 2 19:53 mongodb-linux-x86_64-rhel70-3.2.9.tgz
-rw-r--r-- 1 root root 597 Jan 2 19:58 start_tomcat_and_mongo.sh
```

**Step 3** Build an image.

```
docker build -t apptest .
```

**Step 4** Upload the image to SWR.

----End

### 21.2.1.2.8 Creating a Container Workload

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

#### NOTE

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

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

## Required Cloud Services

- Cloud Container Engine (CCE): a highly reliable and high-performance service that allows enterprises to manage containerized applications. With support for Kubernetes-native applications and tools, CCE makes it simple to set up an environment for running containers in the cloud.

- Elastic Cloud Server (ECS): a scalable and on-demand cloud server. It helps you to efficiently set up reliable, secure, and flexible application environments, ensuring stable service running and improving O&M efficiency.
- Virtual Private Cloud (VPC): an isolated and private virtual network environment that users apply for in the cloud. You can configure the IP address ranges, subnets, and security groups, as well as assign elastic IP addresses and allocate bandwidth in a VPC.

## Basic Concepts

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

## Procedure

**Step 1** Prepare the environment as described in [Table 21-7](#).

**Table 21-7** Preparing the environment

No.	Category	Procedure
1	Creating a VPC	<p>Create a VPC before you create a cluster. A VPC provides an isolated, configurable, and manageable virtual network environment for CCE clusters.</p> <p>If you have a VPC already, skip to the next task.</p> <ol style="list-style-type: none"> <li>1. Log in to the management console.</li> <li>2. In the service list, choose <b>Networking &gt; Virtual Private Cloud</b>.</li> <li>3. On the <b>Dashboard</b> page, click <b>Create VPC</b>.</li> <li>4. Follow the instructions to create a VPC. Retain default settings for parameters unless otherwise specified.</li> </ol>

No.	Category	Procedure
2	Creating a key pair	<p>Create a key pair before you create a containerized application. Key pairs are used for identity authentication during remote login to a node. If you have a key pair already, skip this task.</p> <ol style="list-style-type: none"> <li>1. Log in to the management console.</li> <li>2. In the service list, choose <b>Data Encryption Workshop</b> under <b>Security &amp; Compliance</b>.</li> <li>3. In the navigation pane, choose <b>Key Pair Service</b>. On the <b>Private Key Pairs</b> tab, click <b>Create Key Pair</b>.</li> <li>4. Enter a key pair name, select <b>I agree to have the private key managed on the cloud</b> and <b>I have read and agree to the Key Pair Service Disclaimer</b>, and click <b>OK</b>.</li> <li>5. In the dialog box displayed, click <b>OK</b>. View and save the key pair. For security purposes, a key pair can be downloaded only once. Keep it secure to ensure successful login.</li> </ol>

**Step 2** Create a cluster and a node.

1. Log in to the CCE console. Choose **Clusters**. On the displayed page, select the type of the cluster to be created and click **Create**.  
Configure cluster parameters and select the VPC created in [Step 1](#).
2. Buy a node and select the key pair created in [Step 1](#) as the login mode.

**Step 3** Deploy a workload on CCE.

1. Log in to the CCE console, click the created cluster, choose **Workloads** in the navigation pane, and click **Create Workload** in the upper right corner.
2. Configure the following parameters, and retain the default settings for other parameters:
  - **Workload Name:** Set it to **apptest**.
  - **Pods:** Set it to **1**.
3. In the **Container Settings** area, select the image uploaded in [Building and Uploading an Image](#).
4. In the **Container Settings** area, choose **Environment Variables** and add environment variables for interconnecting with the MySQL database. The environment variables are set in the [startup script](#).

 **NOTE**

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



**Table 21-8** Configuring environment variables

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

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

 **NOTE**

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

The mounted path must be the same as the MongoDB storage path in the Docker startup script. For details, see the [startup script](#). In this example, the path is `/usr/local/mongodb/data`.

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

 **NOTE**

In this example, the application will be accessible from public networks by using an elastic IP address.

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

- **Node Port:** Set it to **Auto**. The system automatically opens a real port on all nodes in the current cluster and then maps the port number to the container port.
  - 7. Click **Create Workload**.  
After the workload is created, you can view the running workload in the workload list.
- End

## Verifying a Workload

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

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

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

## 21.3 Disaster Recovery

### 21.3.1 Implementing High Availability for Applications in CCE

#### Basic Principles

To achieve high availability for your CCE containers, you can do as follows:

1. Deploy three master nodes for the cluster.
2. Create nodes in different AZs. When nodes are deployed across AZs, you can customize scheduling policies based on your requirements to maximize resource utilization.
3. Create multiple node pools in different AZs and use them for node scaling.
4. Set the number of pods to be greater than 2 when creating a workload.
5. Set pod affinity rules to distribute pods to different AZs and nodes.

#### Procedure

Assume that there are four nodes in a cluster distributed in different AZs.

```
$ kubectl get node -L topology.kubernetes.io/zone,kubernetes.io/hostname
NAME STATUS ROLES AGE VERSION ZONE HOSTNAME
192.168.5.112 Ready <none> 42m v1.21.7-r0-CCE21.11.1.B007 zone01 192.168.5.112
192.168.5.179 Ready <none> 42m v1.21.7-r0-CCE21.11.1.B007 zone01 192.168.5.179
192.168.5.252 Ready <none> 37m v1.21.7-r0-CCE21.11.1.B007 zone02 192.168.5.252
192.168.5.8 Ready <none> 33h v1.21.7-r0-CCE21.11.1.B007 zone03 192.168.5.8
```

Create workloads according to the following podAntiAffinity rules:

- Pod anti-affinity in an AZ. Configure the parameters as follows:
  - **weight:** A larger weight value indicates a higher priority of scheduling. In this example, set it to **50**.

- **topologyKey**: includes a default or custom key for the node label that the system uses to denote a topology domain. A topology key determines the scope where the pod should be scheduled to. In this example, set this parameter to **topology.kubernetes.io/zone**, which is the label for identifying the AZ where the node is located.
- **labelSelector**: Select the label of the workload to realize the anti-affinity between this container and the workload.
- The second one is the pod anti-affinity in the node hostname. Configure the parameters as follows:
  - **weight**: Set it to **50**.
  - **topologyKey**: Set it to **kubernetes.io/hostname**.
  - **labelSelector**: Select the label of the pod, which is anti-affinity with the pod.

```

kind: Deployment
apiVersion: apps/v1
metadata:
 name: nginx
 namespace: default
spec:
 replicas: 2
 selector:
 matchLabels:
 app: nginx
 template:
 metadata:
 labels:
 app: nginx
 spec:
 containers:
 - name: container-0
 image: nginx:alpine
 resources:
 limits:
 cpu: 250m
 memory: 512Mi
 requests:
 cpu: 250m
 memory: 512Mi
 affinity:
 podAntiAffinity:
 preferredDuringSchedulingIgnoredDuringExecution:
 - weight: 50
 podAffinityTerm:
 labelSelector: # Select the label of the workload to realize the anti-affinity
 between this container and the workload.
 matchExpressions:
 - key: app
 operator: In
 values:
 - nginx
 namespaces:
 - default
 topologyKey: topology.kubernetes.io/zone # It takes effect in the same AZ.
 - weight: 50
 podAffinityTerm:
 labelSelector: # Select the label of the workload to realize the anti-affinity
 between this container and the workload.
 matchExpressions:
 - key: app
 operator: In
 values:
 - nginx
 namespaces:

```

```
- default
 topologyKey: kubernetes.io/hostname # It takes effect on the node.
imagePullSecrets:
- name: default-secret
```

Create a workload and view the node where the pod is located.

```
$ kubectl get pod -owide
NAME READY STATUS RESTARTS AGE IP NODE
nginx-6fffd8d664-dpwbk 1/1 Running 0 17s 10.0.0.132 192.168.5.112
nginx-6fffd8d664-qhclc 1/1 Running 0 17s 10.0.1.133 192.168.5.252
```

Increase the number of pods to 3. The pod is scheduled to another node, and the three nodes are in three different AZs.

```
$ kubectl scale --replicas=3 deploy/nginx
deployment.apps/nginx scaled
$ kubectl get pod -owide
NAME READY STATUS RESTARTS AGE IP NODE
nginx-6fffd8d664-8t7rv 1/1 Running 0 3s 10.0.0.9 192.168.5.8
nginx-6fffd8d664-dpwbk 1/1 Running 0 2m45s 10.0.0.132 192.168.5.112
nginx-6fffd8d664-qhclc 1/1 Running 0 2m45s 10.0.1.133 192.168.5.252
```

Increase the number of pods to 4. The pod is scheduled to the last node. With podAntiAffinity rules, pods can be evenly distributed to AZs and nodes.

```
$ kubectl scale --replicas=4 deploy/nginx
deployment.apps/nginx scaled
$ kubectl get pod -owide
NAME READY STATUS RESTARTS AGE IP NODE
nginx-6fffd8d664-8t7rv 1/1 Running 0 2m30s 10.0.0.9 192.168.5.8
nginx-6fffd8d664-dpwbk 1/1 Running 0 5m12s 10.0.0.132 192.168.5.112
nginx-6fffd8d664-h796b 1/1 Running 0 78s 10.0.1.5 192.168.5.179
nginx-6fffd8d664-qhclc 1/1 Running 0 5m12s 10.0.1.133 192.168.5.252
```

## 21.4 Security

### 21.4.1 Cluster Security

For security purposes, you are advised to configure a cluster as follows.

#### Using the CCE Cluster of the Latest Version

Kubernetes releases a major version in about four months. CCE follows the same frequency as Kubernetes to release major versions. To be specific, a new CCE version is released about three months after a new Kubernetes version is released in the community. For example, Kubernetes v1.19 was released in September 2020 and CCE v1.19 was released in March 2021.

The latest cluster version has known vulnerabilities fixed or provides a more comprehensive security protection mechanism. You are advised to select the latest cluster version when creating a cluster. Before a cluster version is deprecated and removed, upgrade your cluster to a supported version.

#### Disabling the Automatic Token Mounting Function of the Default Service Account

By default, Kubernetes associates the default service account with every pod. That is, the token is mounted to a container. The container can use this token to pass

the authentication by the kube-apiserver and kubelet components. In a cluster with RBAC disabled, the service account who owns the token has the control permissions for the entire cluster. In a cluster with RBAC enabled, the permissions of the service account who owns the token depends on the roles associated by the administrator. The service account's token is generally used by workloads that need to access kube-apiserver, such as coredns, autoscaler, and prometheus. For workloads that do not need to access kube-apiserver, you are advised to disable the automatic association between the service account and token.

Two methods are available:

- Method 1: Set the **automountServiceAccountToken** field of the service account to **false**. After the configuration is complete, newly created workloads will not be associated with the default service account by default. Configure this field for each namespace as required.

```
apiVersion: v1
kind: ServiceAccount
metadata:
 name: default
automountServiceAccountToken: false
...
```

When a workload needs to be associated with a service account, explicitly set the **automountServiceAccountToken** field to **true** in the YAML file of the workload.

```
...
spec:
 template:
 spec:
 serviceAccountName: default
 automountServiceAccountToken: true
...
```

- Method 2: Explicitly disable the function of automatically associating with service accounts for workloads.

```
...
spec:
 template:
 spec:
 automountServiceAccountToken: false
...
```

## Configuring Proper Cluster Access Permissions for Users

CCE allows you to create multiple IAM users. Your account can create different user groups, assign different access permissions to different user groups, and add users to the user groups with corresponding permissions when creating IAM users. In this way, users can control permissions on different regions and assign read-only permissions. Your account can also assign namespace-level permissions for users or user groups. To ensure security, it is advised that minimum user access permissions are assigned.

If you need to create multiple IAM users, configure the permissions of the IAM users and namespaces properly.

## Configuring Resource Quotas for Cluster Namespaces

CCE provides resource quota management, which allows users to limit the total amount of resources that can be allocated to each namespace. These resources include CPU, memory, storage volumes, pods, Services, Deployments, and

StatefulSets. Proper configuration can prevent excessive resources created in a namespace from affecting the stability of the entire cluster.

## Configuring LimitRange for Containers in a Namespace

With resource quotas, cluster administrators can restrict the use and creation of resources by namespace. In a namespace, a pod or container can use the maximum CPU and memory resources defined by the resource quota of the namespace. In this case, a pod or container may monopolize all available resources in the namespace. You are advised to configure LimitRange to restrict resource allocation within the namespace. The LimitRange parameter has the following restrictions:

- Limits the minimum and maximum resource usage of each pod or container in a namespace.

For example, create the maximum and minimum CPU usage limits for a pod in a namespace as follows:

cpu-constraints.yaml

```
apiVersion: v1
kind: LimitRange
metadata:
 name: cpu-min-max-demo-lr
spec:
 limits:
 - max:
 cpu: "800m"
 min:
 cpu: "200m"
 type: Container
```

Then, run **kubectl -n <namespace> create -f cpu-constraints.yaml** to complete the creation. If the default CPU usage is not specified for the container, the platform automatically configures the default CPU usage. That is, the default configuration is automatically added after the container is created.

```
...
spec:
 limits:
 - default:
 cpu: 800m
 defaultRequest:
 cpu: 800m
 max:
 cpu: 800m
 min:
 cpu: 200m
 type: Container
```

- Limits the maximum and minimum storage space that each PersistentVolumeClaim can apply for in a namespace.

storagelimit.yaml

```
apiVersion: v1
kind: LimitRange
metadata:
 name: storagelimit
spec:
 limits:
 - type: PersistentVolumeClaim
 max:
 storage: 2Gi
```

```
min:
 storage: 1Gi
```

Then, run **kubectl -n <namespace> create -f storagelimit.yaml** to complete the creation.

## Configuring Network Isolation in a Cluster

- Container tunnel network  
If networks need to be isolated between namespaces in a cluster or between workloads in the same namespace, you can configure network policies to isolate the networks.
- VPC network  
Network isolation is not supported.

## Enabling the Webhook Authentication Mode with kubelet

---

### NOTICE

CCE clusters of v1.15.6-r1 or earlier are involved, whereas versions later than v1.15.6-r1 are not.

Upgrade the CCE cluster version to 1.13 or 1.15 and enable the RBAC capability for the cluster. If the version is 1.13 or later, no upgrade is required.

---

When creating a node, you can enable the kubelet authentication mode by injecting the **postinstall** file (by setting the kubelet startup parameter **--authorization-mode=Webhook**).

**Step 1** Run the following command to create clusterrolebinding:

```
kubectl create clusterrolebinding kube-apiserver-kubelet-admin --
clusterrole=system:kubelet-api-admin --user=system:kube-apiserver
```

**Step 2** For an existing node, log in to the node, change **authorization mode** in **/var/paas/kubernetes/kubelet/kubelet\_config.yaml** on the node to **Webhook**, and restart kubelet.

```
sed -i s/AlwaysAllow/Webhook/g /var/paas/kubernetes/kubelet/
kubelet_config.yaml; systemctl restart kubelet
```

**Step 3** For a new node, add the following command to the post-installation script to change the kubelet permission mode:

```
sed -i s/AlwaysAllow/Webhook/g /var/paas/kubernetes/kubelet/
kubelet_config.yaml; systemctl restart kubelet
```

----End

## Uninstalling web-terminal After Use

The web-terminal add-on can be used to manage CCE clusters. Keep the login password secure and uninstall the add-on when it is no longer needed.

## 21.4.2 Node Security

### Preventing Nodes from Being Exposed to Public Networks

- Do not bind an EIP to a node unless necessary to reduce the attack surface.
- If an EIP must be used, properly configure the firewall or security group rules to restrict access of unnecessary ports and IP addresses.

You may have configured the **kubeconfig.json** file on a node in your cluster. `kubectl` can use the certificate and private key in this file to control the entire cluster. You are advised to delete unnecessary files from the **/root/.kube** directory on the node to prevent malicious use.

```
rm -rf /root/.kube
```

### Hardening VPC Security Group Rules

CCE is a universal container platform. Its default security group rules apply to common scenarios. Based on security requirements, you can harden the security group rules set for CCE clusters on the **Security Groups** page of **Network Console**.

### Hardening Nodes on Demand

CCE cluster nodes use the default settings of open source OSs. After a node is created, you need to perform security hardening according to your service requirements.

In CCE, you can perform hardening as follows:

- Use the post-installation script after the node is created. For details, see the description about **Post-installation Script** in **Advanced Settings** when creating a node. This script is user-defined.

### Forbidding Containers to Obtain Host Machine Metadata

If a single CCE cluster is shared by multiple users to deploy containers, containers cannot access the management address (169.254.169.254) of OpenStack, preventing containers from obtaining metadata of host machines.

For details about how to restore the metadata, see the "Notes" section in "Obtaining Metadata".

---

** WARNING**

This solution may affect the password change on the ECS console. Therefore, you must verify the solution before rectifying the fault.

---

**Step 1** Obtain the network model and container CIDR of the cluster.

On the **Cluster Information** page, check the network model and container CIDR block of the cluster in the **Networking Configuration** area.



**Step 2** Prevent the container from obtaining host metadata.

- VPC network
  - a. Log in to each node in the CCE cluster as user **root** and run the following command:

```
iptables -I OUTPUT -s {container_cidr} -d 169.254.169.254 -j REJECT
```

*{container\_cidr}* indicates the container CIDR of the cluster, for example, 10.0.0.0/16.

To ensure configuration persistence, you are advised to write the command to the **/etc/rc.local** script.
  - b. Run the following commands in the container to access the **userdata** and **metadata** interfaces of OpenStack and check whether the request is intercepted:

```
curl 169.254.169.254/openstack/latest/meta_data.json
curl 169.254.169.254/openstack/latest/user_data
```
- Container tunnel network
  - a. Log in to each node in the CCE cluster as user **root** and run the following command:

```
iptables -I FORWARD -s {container_cidr} -d 169.254.169.254 -j REJECT
```

*{container\_cidr}* indicates the container CIDR of the cluster, for example, 10.0.0.0/16.

To ensure configuration persistence, you are advised to write the command to the **/etc/rc.local** script.
  - b. Run the following commands in the container to access the **userdata** and **metadata** interfaces of OpenStack and check whether the request is intercepted:

```
curl 169.254.169.254/openstack/latest/meta_data.json
curl 169.254.169.254/openstack/latest/user_data
```

----End

## 21.4.3 Container Security

### Controlling the Pod Scheduling Scope

The nodeSelector or nodeAffinity is used to limit the range of nodes to which applications can be scheduled, preventing the entire cluster from being threatened due to the exceptions of a single application.

### Suggestions on Container Security Configuration

- Set the computing resource limits (**request** and **limit**) of a container. This prevents the container from occupying too many resources and affecting the stability of the host and other containers on the same node.
- Unless necessary, do not mount sensitive host directories to containers, such as **/**, **/boot**, **/dev**, **/etc**, **/lib**, **/proc**, **/sys**, and **/usr**.
- Do not run the sshd process in containers unless necessary.
- Unless necessary, it is not recommended that containers and hosts share the network namespace.
- Unless necessary, it is not recommended that containers and hosts share the process namespace.

- Unless necessary, it is not recommended that containers and hosts share the IPC namespace.
- Unless necessary, it is not recommended that containers and hosts share the UTS namespace.
- Unless necessary, do not mount the sock file of Docker to any container.

## Container Permission Access Control

When using a containerized application, comply with the minimum privilege principle and properly set securityContext of Deployments or StatefulSets.

- Configure runAsUser to specify a non-root user to run a container.
- Configure privileged to prevent containers being used in scenarios where privilege is not required.
- Configure capabilities to accurately control the privileged access permission of containers.
- Configure allowPrivilegeEscalation to disable privilege escape in scenarios where privilege escalation is not required for container processes.
- Configure seccomp to restrict the container syscalls. For details, see [Restrict a Container's Syscalls with seccomp](#) in the official Kubernetes documentation.
- Configure ReadOnlyRootFilesystem to protect the root file system of a container.

Example YAML for a Deployment:

```
apiVersion: apps/v1
kind: Deployment
metadata:
 name: security-context-example
 namespace: security-example
spec:
 replicas: 1
 selector:
 matchLabels:
 app: security-context-example
 label: security-context-example
 strategy:
 rollingUpdate:
 maxSurge: 25%
 maxUnavailable: 25%
 type: RollingUpdate
 template:
 metadata:
 annotations:
 seccomp.security.alpha.kubernetes.io/pod: runtime/default
 labels:
 app: security-context-example
 label: security-context-example
 spec:
 containers:
 - image: ...
 imagePullPolicy: Always
 name: security-context-example
 securityContext:
 allowPrivilegeEscalation: false
 readOnlyRootFilesystem: true
 runAsUser: 1000
 capabilities:
 add:
 - NET_BIND_SERVICE
 drop:
```

```

- all
volumeMounts:
- mountPath: /etc/localtime
 name: localtime
 readOnly: true
- mountPath: /opt/write-file-dir
 name: tmpfs-example-001
securityContext:
seccompProfile:
 type: RuntimeDefault
volumes:
- hostPath:
 path: /etc/localtime
 type: ""
 name: localtime
- emptyDir: {}
 name: tmpfs-example-001

```

## Restricting the Access of Containers to the Management Plane

If application containers on a node do not need to access Kubernetes, you can perform the following operations to disable containers from accessing kube-apiserver:

**Step 1** Query the container CIDR block and private API server address.

On the **Clusters** page of the CCE console, click the name of the cluster to find the information on the details page.

**Step 2** Log in to each node in the CCE cluster as user **root** and run the following command:

- VPC network:  

```
iptables -I OUTPUT -s {container_cidr} -d {Private API server IP} -j REJECT
```
- Container tunnel network:  

```
iptables -I FORWARD -s {container_cidr} -d {Private API server IP} -j REJECT
```

*{container\_cidr}* indicates the container CIDR of the cluster, for example, 10.0.0.0/16.

To ensure configuration persistence, you are advised to write the command to the **/etc/rc.local** script.

**Step 3** Run the following command in the container to access kube-apiserver and check whether the request is intercepted:

```
curl -k https://{Private API server IP}:5443
```

----End

### 21.4.4 Secret Security

Currently, CCE has configured static encryption for secret resources. The secrets created by users will be encrypted and stored in etcd of the CCE cluster. Secrets can be used in two modes: environment variable and file mounting. No matter which mode is used, CCE still transfers the configured data to users. Therefore, it is recommended that:

1. Do not record sensitive information in logs.
2. For the secret that uses the file mounting mode, the default file permission mapped in the container is 0644. Configure stricter permissions for the file. For example:

```

apiversion: v1
kind: Pod
metadata:
 name: mypod
spec:
 containers:
 - name: mypod
 image: redis
 volumeMounts:
 - name: foo
 mountPath: "/etc/foo"
 volumes:
 - name: foo
 secret:
 secretName: mysecret
 defaultMode: 256

```

In **defaultMode: 256**, **256** is a decimal number, which corresponds to the octal number **0400**.

3. When the file mounting mode is used, configure the secret file name to hide the file in the container.

```

apiVersion: v1
kind: Secret
metadata:
 name: dotfile-secret
data:
 .secret-file: dmFsdWUtMg0KDQo=

apiVersion: v1
kind: Pod
metadata:
 name: secret-dotfiles-pod
spec:
 volumes:
 - name: secret-volume
 secret:
 secretName: dotfile-secret
 containers:
 - name: dotfile-test-container
 image: k8s.gcr.io/busybox
 command:
 - ls
 - "-l"
 - "/etc/secret-volume"
 volumeMounts:
 - name: secret-volume
 readOnly: true
 mountPath: "/etc/secret-volume"

```

In this way, **.secret-file** cannot be viewed by running the **ls -l** command in the **/etc/secret-volume/** directory, but can be viewed by running the **ls -al** command.

4. Encrypt sensitive information before creating a secret and decrypt the information when using it.

## Using a Bound ServiceAccount Token to Access a Cluster

The secret-based ServiceAccount token does not support expiration time or auto update. In addition, after the mounting pod is deleted, the token is still stored in the secret. Token leakage may incur security risks. A bound ServiceAccount token is recommended for CCE clusters of version 1.23 or later. In this mode, the expiration time can be set and is the same as the pod lifecycle, reducing token leakage risks. Example:

```

apiVersion: apps/v1
kind: Deployment
metadata:
 name: security-token-example
 namespace: security-example
spec:
 replicas: 1
 selector:
 matchLabels:
 app: security-token-example
 label: security-token-example
 template:
 metadata:
 annotations:
 seccomp.security.alpha.kubernetes.io/pod: runtime/default
 labels:
 app: security-token-example
 label: security-token-example
 spec:
 serviceAccountName: test-sa
 containers:
 - image: ...
 imagePullPolicy: Always
 name: security-token-example
 volumes:
 - name: test-projected
 projected:
 defaultMode: 420
 sources:
 - serviceAccountToken:
 expirationSeconds: 1800
 path: token
 - configMap:
 items:
 - key: ca.crt
 path: ca.crt
 name: kube-root-ca.crt
 - downwardAPI:
 items:
 - fieldRef:
 apiVersion: v1
 fieldPath: metadata.namespace
 path: namespace

```

For details, visit <https://kubernetes.io/docs/reference/access-authn-authz/service-accounts-admin/>.

## 21.5 Auto Scaling

### 21.5.1 Using HPA and CA for Auto Scaling of Workloads and Nodes

#### Application Scenarios

The best way to handle surging traffic is to automatically adjust the number of machines based on the traffic volume or resource usage, which is called scaling.

When pods or containers are used for deploying applications, the upper limit of available resources is typically required to set for pods or containers to prevent unlimited usage of node resources during peak hours. However, after the upper limit is reached, an application error may occur. To resolve this issue, scale in the

number of pods to share workloads. If the node resource usage increases to a certain extent that newly added pods cannot be scheduled, scale in the number of nodes based on the node resource usage.

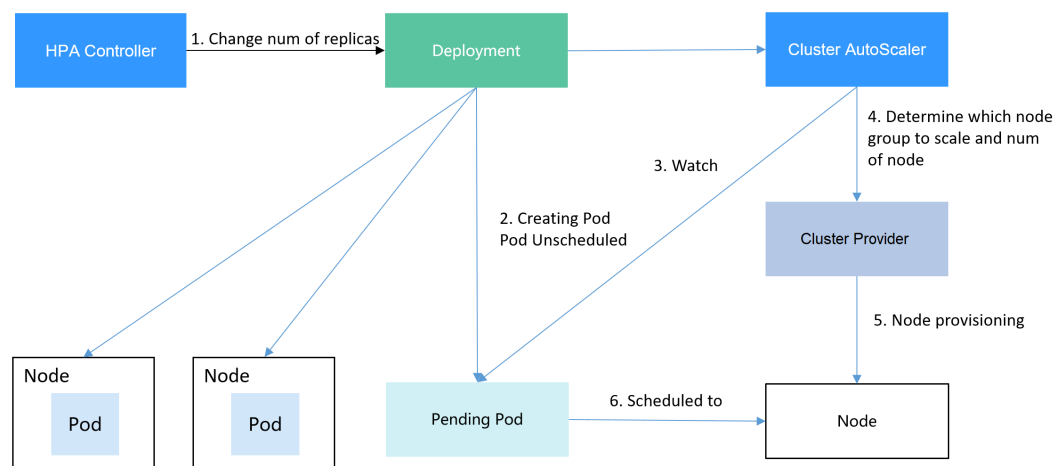
## Solution

Two major auto scaling policies are HPA (Horizontal Pod Autoscaling) and CA (Cluster AutoScaling). HPA is for workload auto scaling and CA is for node auto scaling.

HPA and CA work with each other. HPA requires sufficient cluster resources for successful scaling. When the cluster resources are insufficient, CA is needed to add nodes. If HPA reduces workloads, the cluster will have a large number of idle resources. In this case, CA needs to release nodes to avoid resource waste.

As shown in [Figure 21-4](#), 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 21-4** HPA and CA working flows



Using HPA and CA can easily implement auto scaling in most scenarios. In addition, the scaling process of nodes and pods can be easily observed.

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 to create an organization.

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 node pool parameters, add a node with 2 vCPUs and 4 GiB memory, and enable auto scaling.
- **Nodes:** Set it to **1**, indicating that one node is created by default when a node pool is created.
  - **Auto Scaling:** Enable the option, meaning that nodes will be automatically created or deleted in the node pool based on the cluster loads.
  - **Max. Nodes:** Set it to **5**, indicating the maximum number of nodes in a node pool.
  - **Specifications:** 2 vCPUs | 4 GiB

Retain the defaults for other parameters.

- Step 3** Click **Add-ons** on the left of the cluster console, click **Edit** under the autoscaler add-on, modify the add-on configuration, enable **Auto node scale-in**, and configure scale-in parameters. For example, trigger scale-in when the node resource utilization is less than 50%.

After the preceding configurations, scale-out is performed based on the pending status of the pod and scale-in is triggered when the node resource utilization decreases.

- Step 4** Click **Node Scaling** on the left of the cluster console and click **Create Node Scaling Policy** in the upper right corner. Node scaling policies added here trigger scale-out based on the CPU/memory allocation rate or periodically.

As shown in the following figure, when the cluster CPU allocation rate is greater than 70%, one node will be added. 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.

----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 Pending state. During this period, nodes are added.

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 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

Using HPA and CA can easily implement auto scaling in most scenarios. In addition, the scaling process of nodes and pods can be easily observed.

## 21.6 Monitoring

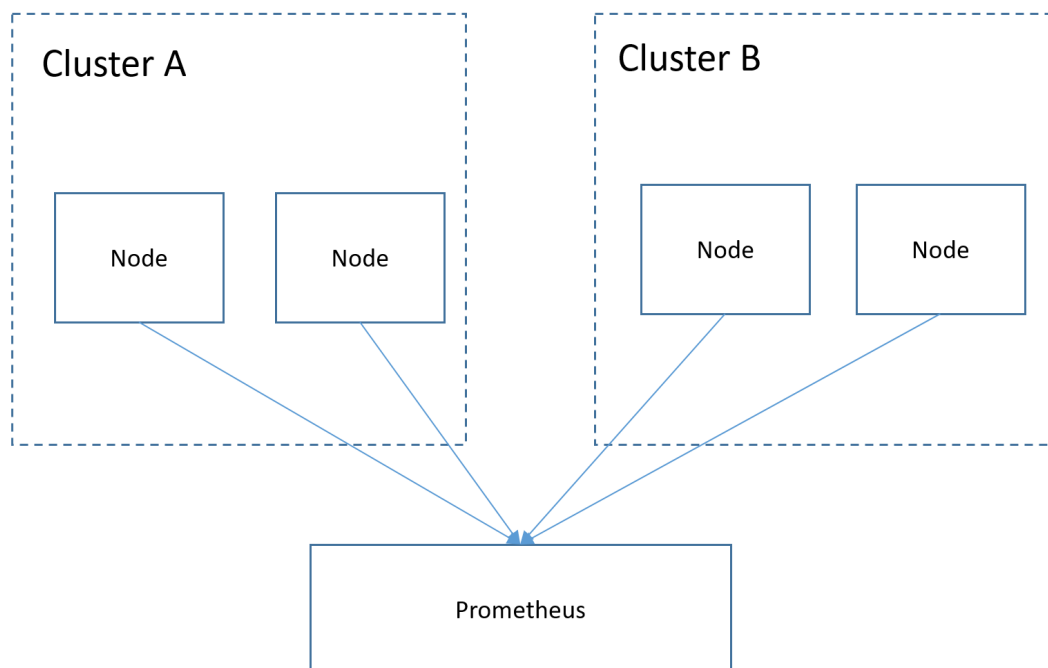
## 21.6.1 Using Prometheus for Multi-cluster Monitoring

### Application Scenarios

Generally, a user has different clusters for different purposes, such as production, testing, and development. To monitor, collect, and view metrics of these clusters, you can deploy a set of Prometheus.

### Solution Architecture

Multiple clusters are connected to the same Prometheus monitoring system, as shown in the following figure. This reduces maintenance and resource costs and facilitates monitoring information aggregation.



### Prerequisites

- The target cluster has been created.
- Prometheus has been properly connected to the target cluster.
- Prometheus has been installed on a Linux host using a binary file. For details, see [Installation](#).

### Procedure

**Step 1** Obtain the **bearer\_token** information of the target cluster.

1. Create the RBAC permission in the target cluster.

Log in to the background node of the target cluster and create the **prometheus\_rbac.yaml** file.

```
apiVersion: v1
kind: ServiceAccount
metadata:
 name: prometheus-test
```

```

namespace: kube-system

apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
 name: prometheus-test
rules:
- apiGroups:
 - ""
 resources:
 - nodes
 - services
 - endpoints
 - pods
 - nodes/proxy
 verbs:
 - get
 - list
 - watch
- apiGroups:
 - "extensions"
 resources:
 - ingresses
 verbs:
 - get
 - list
 - watch
- apiGroups:
 - ""
 resources:
 - configmaps
 - nodes/metrics
 verbs:
 - get
- nonResourceURLs:
 - /metrics
 verbs:
 - get

apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
 name: prometheus-test
roleRef:
 apiGroup: rbac.authorization.k8s.io
 kind: ClusterRole
 name: prometheus-test
subjects:
- kind: ServiceAccount
 name: prometheus-test
 namespace: kube-system

```

Run the following command to create the RBAC permission:

**kubectl apply -f prometheus\_rbac.yaml**

2. Obtain the **bearer\_token** information of the target cluster.



```

- job_name: k8s_cAdvisor
 scheme: https
 bearer_token_file: k8s_token # Token file in the previous step.
 tls_config:
 insecure_skip_verify: true
 kubernetes_sd_configs: # kubernetes automatic discovery configuration
- role: node # Automatic discovery of the node type
 bearer_token_file: k8s_token # Token file in the previous step
 api_server: https://192.168.0.153:5443 # API server address of the Kubernetes cluster
 tls_config:
 insecure_skip_verify: true # Skip the authentication on the server.
 relabel_configs: ## Modify the existing label of the target cluster before capturing metrics.
- target_label: __address__
 replacement: 192.168.0.153:5443
 action: replace
 ## Convert metrics_path to /api/v1/nodes/${1}/proxy/metrics/cadvisor.
 # Obtain data from kubelet using the API server proxy.
- source_labels: [__meta_kubernetes_node_name] # Specifies the source label to be processed.
 regex: (.+) # Matched value of the source label. (.+) indicates that any value of the source label can
be matched.
 target_label: __metrics_path__ # Specifies the label to be replaced.
 replacement: /api/v1/nodes/${1}/proxy/metrics/cadvisor # Indicates the new label, that is, the value of
__metrics_path__. ${1} indicates the value that matches the regular expression, that is, node name.
- target_label: cluster
 replacement: xxxxx ## (Optional) Enter the cluster information.

The following job monitors another cluster.
- job_name: k8s02_cAdvisor
 scheme: https
 bearer_token_file: k8s02_token # Token file in the previous step
 tls_config:
 insecure_skip_verify: true
 kubernetes_sd_configs:
- role: node
 bearer_token_file: k8s02_token # Token file in the previous step
 api_server: https://192.168.0.147:5443 # API server address of the Kubernetes cluster
 tls_config:
 insecure_skip_verify: true # Skip the authentication on the server.
 relabel_configs: ## Modify the existing label of the target cluster before capturing metrics.
- target_label: __address__
 replacement: 192.168.0.147:5443
 action: replace

- source_labels: [__meta_kubernetes_node_name]
 regex: (.+)
 target_label: __metrics_path__
 replacement: /api/v1/nodes/${1}/proxy/metrics/cadvisor

- target_label: cluster
 replacement: xxxx ## (Optional) Enter the cluster information.

```

#### Step 4 Enable Prometheus.

After the configuration, enable Prometheus.

**./prometheus --config.file=prometheus.yml**

#### Step 5 Log in to Prometheus and view the monitoring information.



### Targets

All Unhealthy

k8s02\_cAdvisor (2/2 up) [show less](#)

Endpoint	State	Labels	Last Scrape	Scrape Duration	Error
https://192.168.0.223:5443/api/v1/nodes/192.168.0.110:10250/proxy/metrics/cadvisor	UP	cluster="h... instance="192.168.0.110" job="k8s02_cAdvisor"	1.689s	47.677ms	
https://192.168.0.223:5443/api/v1/nodes/192.168.0.162:10250/proxy/metrics/cadvisor	UP	cluster="h... instance="192.168.0.162" job="k8s02_cAdvisor"	7.279s	65.193ms	

k8s\_cAdvisor (4/4 up) [show less](#)

Endpoint	State	Labels	Last Scrape	Scrape Duration	Error
https://192.168.0.153:5443/api/v1/nodes/192.168.0.65:10250/proxy/metrics/cadvisor	UP	cluster="T...-testspa" instance="192.168.0.65" job="k8s_cAdvisor"	12.365s	37.925ms	
https://192.168.0.153:5443/api/v1/nodes/192.168.0.250:10250/proxy/metrics/cadvisor	UP	cluster="T...-testspa" instance="192.168.0.250" job="k8s_cAdvisor"	2.390s	29.235ms	
https://192.168.0.153:5443/api/v1/nodes/192.168.0.109:10250/proxy/metrics/cadvisor	UP	cluster="T...-testspa" instance="192.168.0.109" job="k8s_cAdvisor"	1.578s	102.146ms	
https://192.168.0.153:5443/api/v1/nodes/192.168.0.228:10250/proxy/metrics/cadvisor	UP	cluster="T...-testspa" instance="192.168.0.228" job="k8s_cAdvisor"	416.000ms	21.256ms	

Prometheus Alerts Graph Status Help Classic UI

Enable query history Use local time Enable autocomplete

container\_cpu\_load\_average\_10s(namespaces="default")

Table	Graph
container_cpu_load_average_10s{cluster="h...", container="POD", id="kubepods/pod946a0fc-aa4d-40e4-aba8-1f76ba03d2bf/0008b1880a1187214c1f8494d3614d3a1b3d1944ed3e27070a5d3f386387e", image="cncf/pause:3.1", instance="192.168.0.162", job="k8s02_cAdvisor", name="k8s_POD_test-70554b08fc-vp8hc-defaul..."} 0	
container_cpu_load_average_10s{cluster="h...", container="POD", id="kubepods/pod946a0fc-aa4d-40e4-aba8-1f76ba03d2bf/0008b1880a1187214c1f8494d3614d3a1b3d1944ed3e27070a5d3f386387e", image="cncf/pause:3.1", instance="192.168.0.162", job="k8s02_cAdvisor", name="k8s_POD_test-70554b08fc-vp8hc-defaul..."} 0	
container_cpu_load_average_10s{cluster="T...-testspa", container="POD", id="kubepods/pod946a0fc-aa4d-40e4-aba8-1f76ba03d2bf/0008b1880a1187214c1f8494d3614d3a1b3d1944ed3e27070a5d3f386387e", image="cncf/pause:3.1", instance="192.168.0.162", job="k8s02_cAdvisor", name="k8s_POD_test-70554b08fc-vp8hc-defaul..."} 0	
container_cpu_load_average_10s{cluster="T...-testspa", container="POD", id="kubepods/pod946a0fc-aa4d-40e4-aba8-1f76ba03d2bf/0008b1880a1187214c1f8494d3614d3a1b3d1944ed3e27070a5d3f386387e", image="cncf/pause:3.1", instance="192.168.0.162", job="k8s02_cAdvisor", name="k8s_POD_test-70554b08fc-vp8hc-defaul..."} 0	
container_cpu_load_average_10s{cluster="T...-testspa", container="POD", id="kubepods/pod946a0fc-aa4d-40e4-aba8-1f76ba03d2bf/0008b1880a1187214c1f8494d3614d3a1b3d1944ed3e27070a5d3f386387e", image="cncf/pause:3.1", instance="192.168.0.162", job="k8s02_cAdvisor", name="k8s_POD_test-70554b08fc-vp8hc-defaul..."} 0	
container_cpu_load_average_10s{cluster="T...-testspa", container="POD", id="kubepods/pod946a0fc-aa4d-40e4-aba8-1f76ba03d2bf/0008b1880a1187214c1f8494d3614d3a1b3d1944ed3e27070a5d3f386387e", image="cncf/pause:3.1", instance="192.168.0.162", job="k8s02_cAdvisor", name="k8s_POD_test-70554b08fc-vp8hc-defaul..."} 0	
container_cpu_load_average_10s{cluster="T...-testspa", container="POD", id="kubepods/pod946a0fc-aa4d-40e4-aba8-1f76ba03d2bf/0008b1880a1187214c1f8494d3614d3a1b3d1944ed3e27070a5d3f386387e", image="cncf/pause:3.1", instance="192.168.0.162", job="k8s02_cAdvisor", name="k8s_POD_test-70554b08fc-vp8hc-defaul..."} 0	
container_cpu_load_average_10s{cluster="T...-testspa", container="POD", id="kubepods/pod946a0fc-aa4d-40e4-aba8-1f76ba03d2bf/0008b1880a1187214c1f8494d3614d3a1b3d1944ed3e27070a5d3f386387e", image="cncf/pause:3.1", instance="192.168.0.162", job="k8s02_cAdvisor", name="k8s_POD_test-70554b08fc-vp8hc-defaul..."} 0	
container_cpu_load_average_10s{cluster="T...-testspa", container="POD", id="kubepods/pod946a0fc-aa4d-40e4-aba8-1f76ba03d2bf/0008b1880a1187214c1f8494d3614d3a1b3d1944ed3e27070a5d3f386387e", image="cncf/pause:3.1", instance="192.168.0.162", job="k8s02_cAdvisor", name="k8s_POD_test-70554b08fc-vp8hc-defaul..."} 0	
container_cpu_load_average_10s{cluster="T...-testspa", container="POD", id="kubepods/pod946a0fc-aa4d-40e4-aba8-1f76ba03d2bf/0008b1880a1187214c1f8494d3614d3a1b3d1944ed3e27070a5d3f386387e", image="cncf/pause:3.1", instance="192.168.0.162", job="k8s02_cAdvisor", name="k8s_POD_test-70554b08fc-vp8hc-defaul..."} 0	
container_cpu_load_average_10s{cluster="T...-testspa", container="POD", id="kubepods/pod946a0fc-aa4d-40e4-aba8-1f76ba03d2bf/0008b1880a1187214c1f8494d3614d3a1b3d1944ed3e27070a5d3f386387e", image="cncf/pause:3.1", instance="192.168.0.162", job="k8s02_cAdvisor", name="k8s_POD_test-70554b08fc-vp8hc-defaul..."} 0	

----End

## 21.7 Cluster

### 21.7.1 Connecting to Multiple Clusters Using kubectl

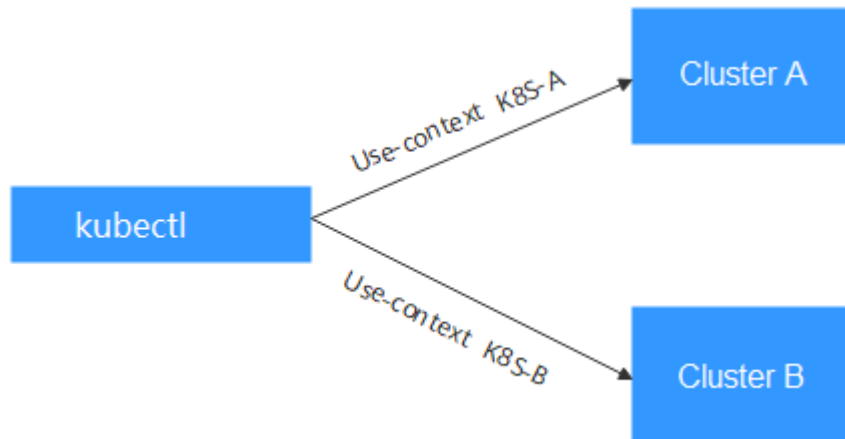
#### Background

When you have multiple CCE clusters, you may find it difficult to efficiently connect to all of them.

#### Solution

This section describes how to configure access to multiple clusters by modifying **kubeconfig.json**. The file describes multiple clusters, users, and contexts. To access different clusters, run the **kubectl config use-context** command to switch between contexts.

**Figure 21-5** Using kubectl to connect to multiple clusters



## Prerequisites

kubectl can access multiple clusters.

## Introduction to kubeconfig.json

kubeconfig.json is the configuration file of kubectl. You can download it on the cluster details page.

The content of kubeconfig.json is as follows:

```
{
 "kind": "Config",
 "apiVersion": "v1",
 "preferences": {},
 "clusters": [
 {
 "name": "internalCluster",
 "cluster": {
 "server": "https://192.168.0.85:5443",
 "certificate-authority-data": "LS0tLS1CRUULIE..."
 }
 },
 {
 "name": "externalCluster",
 "cluster": {
 "server": "https://xxx.xxx.xxx.xxx:5443",
 "insecure-skip-tls-verify": true
 }
 }
],
 "users": [
 {
 "name": "user",
 "user": {
 "client-certificate-data": "LS0tLS1CRUdJTiBDRVJ...",
 "client-key-data": "LS0tLS1CRUdJTiBS..."
 }
 }
],
 "contexts": [
 {
 "name": "internal",
 "context": {
 "cluster": "internalCluster",
 "user": "user"
 }
 },
 {
 "name": "external",
 "context": {
```

```

 "cluster": "externalCluster",
 "user": "user"
 }
 },
 "current-context": "external"
}

```

It mainly consists of three sections.

- **clusters:** describes the cluster information, mainly the access address of the cluster.
- **users:** describes information about the users who access the cluster. It includes the **client-certificate-data** and **client-key-data** certificate files.
- **contexts:** describes the configuration contexts. You switch between contexts to access different clusters. A context is associated with **user** and **cluster**, that is, it defines which user accesses which cluster.

The preceding kubeconfig.json defines the private network address and public network address of the cluster as two clusters with two different contexts. You can switch the context to use different addresses to access the cluster.

## Configuring Access to Multiple Clusters

The following steps walk you through the procedure of configuring access to two clusters by modifying kubeconfig.json.

This example configures only the public network access to the clusters. If you want to access multiple clusters over private networks, retain the **clusters** field and ensure that the clusters can be accessed over private networks. Its configuration is similar to that described in this example.

**Step 1** Download kubeconfig.json of the two clusters and delete the lines related to private network access, as shown in the following figure.

- Cluster A:

```

{
 "kind": "Config",
 "apiVersion": "v1",
 "preferences": {},
 "clusters": [{
 "name": "externalCluster",
 "cluster": {
 "server": "https://119.xxx.xxx.xxx:5443",
 "insecure-skip-tls-verify": true
 }
 }
],
 "users": [{
 "name": "user",
 "user": {
 "client-certificate-data": "LS0tLS1CRUdJTzM...",
 "client-key-data": "LS0tLS1CRUdJTjB..."
 }
 }
],
 "contexts": [{
 "name": "external",
 "context": {
 "cluster": "externalCluster",
 "user": "user"
 }
 }
],
 "current-context": "external"
}

```

- Cluster B:

```
{
 "kind": "Config",
 "apiVersion": "v1",
 "preferences": {},
 "clusters": [{
 "name": "externalCluster",
 "cluster": {
 "server": "https://124.xxx.xxx.xxx:5443",
 "insecure-skip-tls-verify": true
 }
 }
],
 "users": [{
 "name": "user",
 "user": {
 "client-certificate-data": "LS0tLS1CRUdJTxB...",
 "client-key-data": "LS0rTUideUdJTjB...."
 }
 }
],
 "contexts": [{
 "name": "external",
 "context": {
 "cluster": "externalCluster",
 "user": "user"
 }
 }
],
 "current-context": "external"
}
```

The preceding files have the same structure except that the **client-certificate-data** and **client-key-data** fields of **user** and the **clusters.cluster.server** field are different.

**Step 2** Modify the **name** field as follows:

- Cluster A:

```
{
 "kind": "Config",
 "apiVersion": "v1",
 "preferences": {},
 "clusters": [{
 "name": "Cluster-A",
 "cluster": {
 "server": "https://119.xxx.xxx.xxx:5443",
 "insecure-skip-tls-verify": true
 }
 }
],
 "users": [{
 "name": "Cluster-A-user",
 "user": {
 "client-certificate-data": "LS0tLS1CRUdJTxB...",
 "client-key-data": "LS0tLS1CRUdJTjB...."
 }
 }
],
 "contexts": [{
 "name": "Cluster-A-Context",
 "context": {
 "cluster": "Cluster-A",
 "user": "Cluster-A-user"
 }
 }
],
 "current-context": "Cluster-A-Context"
}
```

- Cluster B:

```
{
 "kind": "Config",
 "apiVersion": "v1",
 "preferences": {},
 "clusters": [{
```

```

 "name": "Cluster-B",
 "cluster": {
 "server": "https://124.xxx.xxx.xxx:5443",
 "insecure-skip-tls-verify": true
 }
 },
 "users": [{
 "name": "Cluster-B-user",
 "user": {
 "client-certificate-data": "LS0tLS1CRUdJTzM...",
 "client-key-data": "LS0rTUideUdJTjB...."
 }
 }],
 "contexts": [{
 "name": "Cluster-B-Context",
 "context": {
 "cluster": "Cluster-B",
 "user": "Cluster-B-user"
 }
 }],
 "current-context": "Cluster-B-Context"
}

```

**Step 3** Combine these two files.

The file structure remains unchanged. Combine the contents of **clusters**, **users**, and **contexts** as follows:

```

{
 "kind": "Config",
 "apiVersion": "v1",
 "preferences": {},
 "clusters": [{
 "name": "Cluster-A",
 "cluster": {
 "server": "https://119.xxx.xxx.xxx:5443",
 "insecure-skip-tls-verify": true
 }
 },
 {
 "name": "Cluster-B",
 "cluster": {
 "server": "https://124.xxx.xxx.xxx:5443",
 "insecure-skip-tls-verify": true
 }
 }
],
 "users": [{
 "name": "Cluster-A-user",
 "user": {
 "client-certificate-data": "LS0tLS1CRUdJTzM...",
 "client-key-data": "LS0tLS1CRUdJTjB...."
 }
 },
 {
 "name": "Cluster-B-user",
 "user": {
 "client-certificate-data": "LS0tLS1CRUdJTzM...",
 "client-key-data": "LS0rTUideUdJTjB...."
 }
 }
],
 "contexts": [{
 "name": "Cluster-A-Context",
 "context": {
 "cluster": "Cluster-A",
 "user": "Cluster-A-user"
 }
 },
 {
 "name": "Cluster-B-Context",
 "context": {

```

```
"cluster": "Cluster-B",
 "user": "Cluster-B-user"
 }
}],
 "current-context": "Cluster-A-Context"
}
```

----End

## Verification

Run the following commands to copy the file to the kubectl configuration path:

```
mkdir -p $HOME/.kube
```

```
mv -f kubeconfig.json $HOME/.kube/config
```

Run the kubectl commands to check whether the two clusters can be connected.

```
kubectl config use-context Cluster-A-Context
Switched to context "Cluster-A-Context".
kubectl cluster-info
Kubernetes control plane is running at https://119.xxx.xxx.xxx:5443
CoreDNS is running at https://119.xxx.xxx.xxx:5443/api/v1/namespaces/kube-system/services/coresdns/dns/
proxy
```

To further debug and diagnose cluster problems, use 'kubectl cluster-info dump'.

```
kubectl config use-context Cluster-B-Context
Switched to context "Cluster-B-Context".
kubectl cluster-info
Kubernetes control plane is running at https://124.xxx.xxx.xxx:5443
CoreDNS is running at https://124.xxx.xxx.xxx:5443/api/v1/namespaces/kube-system/services/coresdns/dns/
proxy
```

To further debug and diagnose cluster problems, use 'kubectl cluster-info dump'.

## 21.7.2 Selecting a Data Disk for the Node

When a node is created, a data disk is attached by default for a container runtime and kubelet. The data disk used by the container runtime and kubelet cannot be detached, and the default capacity is 100 GiB. To cut costs, you can reduce the disk capacity attached to a node to the minimum of 10 GiB.

**NOTICE**

Adjusting the size of the data disk used by the container runtime and kubelet may incur risks. You are advised to evaluate the capacity adjustment and then perform the operations described in this section.

- If the disk capacity is too small, the image pull may fail. If different images need to be frequently pulled on the node, you are not advised to reduce the data disk capacity.
- Before a cluster upgrade, the system checks whether the data disk usage exceeds 95%. If the usage is high, the cluster upgrade may be affected.
- If Device Mapper is used, the disk capacity may be insufficient. You are advised to use the OverlayFS or select a large-capacity data disk.
- For dumping logs, application logs must be stored in a separate disk to prevent insufficient storage capacity of the dockersys volume from affecting service running.
- After reducing the data disk capacity, you are advised to install the npd add-on in the cluster to detect disk usage. If the disk usage of a node is high, resolve this problem by referring to [What If the Data Disk Capacity Is Insufficient?](#)

## Constraints

- Only clusters of v1.19 or later allow reducing the capacity of the data disk used by container runtimes and kubelet.
- Only the EVS disk capacity can be adjusted. (Local disks are available only when the node specification is **disk-intensive** or **Ultra-high I/O**.)

## Selecting a Data Disk

When selecting a data disk, consider the following factors:

- During image pull, the system downloads the image package (the .tar package) from the image repository, and decompresses the package. Then it deletes the package but retain the image file. During the decompression of the .tar package, the package and the decompressed image file coexist. Reserve the capacity for the decompressed files.
- Mandatory add-ons (such as everest and coredns) may be deployed on nodes during cluster creation. When calculating the data disk size, reserve about 2 GiB storage capacity for them.
- Logs are generated during application running. To ensure stable application running, reserve about 1 GiB storage capacity for each pod.

For details about the calculation formulas, see [OverlayFS](#) and [Device Mapper](#).

## OverlayFS

By default, the container engine and container image storage capacity of a node using the OverlayFS storage driver occupies 90% of the data disk capacity (you are advised to retain this value). All the 90% storage capacity is used for dockersys partitioning. The calculation methods are as follows:

- Capacity for storing container engines and container images requires 90% of the data disk capacity by default.
  - Capacity for dockersys volume (in the `/var/lib/docker` directory) requires 90% of the data disk capacity. The entire container engine and container image capacity (need 90% of the data disk capacity by default) are in the `/var/lib/docker` directory.
- Capacity for storing temporary kubelet and emptyDir requires 10% of the data disk capacity.

On a node using the OverlayFS, when an image is pulled, the .tar package is decompressed after being downloaded. During this process, the .tar package and the decompressed image file are stored in the dockersys volume, occupying about twice the actual image storage capacity. After the decompression is complete, the .tar package is deleted. Therefore, during image pull, after deducting the storage capacity occupied by the system add-on images, ensure that the remaining capacity of the dockersys volume is greater than twice the actual image storage capacity. To ensure that the containers can run stably, reserve certain capacity in the dockersys volume for container logs and other related files.

When selecting a data disk, consider the following formula:

**Capacity of dockersys volume > Actual total image storage capacity x 2 + Total system add-on image storage capacity (about 2 GiB) + Number of containers x Available storage capacity for a single container (about 1 GiB log storage capacity for each container)**

#### NOTE

If container logs are output in the `json.log` format, they will occupy some capacity in the dockersys volume. If container logs are stored on persistent storage, they will not occupy capacity in the dockersys volume. Estimate the capacity of every container as required.

Example:

Assume that the node uses the OverlayFS and the data disk attached to this node is 20 GiB. According to [the preceding methods](#), the capacity for storing container engines and images occupies 90% of the data disk capacity, and the capacity for the dockersys volume is 18 GiB (20 GiB x 90%). Additionally, mandatory add-ons may occupy about 2 GiB storage capacity during cluster creation. If you deploy a .tar package of 10 GiB, the package decompression takes 20 GiB of the dockersys volume's storage capacity. This, coupled with the storage capacity occupied by mandatory add-ons, exceeds the remaining capacity of the dockersys volume. As a result, the image pull may fail.

## Device Mapper

By default, the capacity for storing container engines and container images of a node using the Device Mapper storage driver occupies 90% of the data disk capacity (you are advised to retain this value). The occupied capacity includes the dockersys volume and thinpool volume. The calculation methods are as follows:

- Capacity for storing container engines and container images requires 90% of the data disk capacity by default.
  - Capacity for the dockersys volume (in the `/var/lib/docker` directory) requires 20% of the capacity for storing container engines and container images.



- Capacity for the thinpool volume requires 80% of the container engine and container image storage capacity.
- Capacity for storing temporary kubelet and emptyDir requires 10% of the data disk capacity.

On a node using the Device Mapper storage driver, when an image is pulled, the .tar package is temporarily stored in the dockersys volume. After the .tar package is decompressed, the image file is stored in the thinpool volume, and the package in the dockersys volume will be deleted. Therefore, during image pull, ensure that the storage capacity of the dockersys volume and thinpool volume are sufficient, and note that the former is smaller than the latter. To ensure that the containers can run stably, reserve certain capacity in the dockersys volume for container logs and other related files.

When selecting a data disk, consider the following formulas:

- **Capacity for dockersys volume > Temporary storage capacity of the .tar package (approximately equal to the actual total image storage capacity) + Number of containers x Storage capacity of a single container (about 1 GiB log storage capacity must be reserved for each container)**
- **Capacity for thinpool volume > Actual total image storage capacity + Total add-on image storage capacity (about 2 GiB)**

#### NOTE

If container logs are output in the `json.log` format, they will occupy some capacity in the dockersys volume. If container logs are stored on persistent storage, they will not occupy capacity in the dockersys volume. Estimate the capacity of every container as required.

Example:

Assume that the node uses the Device Mapper and the data disk attached to this node is 20 GiB. According to [the preceding methods](#), the container engine and image storage capacity occupies 90% of the data disk capacity, and the disk usage of the dockersys volume is 3.6 GiB. Additionally, the storage capacity of the mandatory add-ons may occupy about 2 GiB of the dockersys volume during cluster creation. The remaining storage capacity is about 1.6 GiB. If you deploy a .tar image package larger than 1.6 GiB, the storage capacity of the dockersys volume is insufficient for the package to be decompressed. As a result, the image pull may fail.

## What If the Data Disk Capacity Is Insufficient?

### Solution 1: Clearing images

Perform the following operations to clear unused images:

- Nodes that use containerd
  - a. Obtain local images on the node.

```
crictl images -v
```
  - b. Delete the images that are not required by image ID.

```
crictl rmi Image ID
```
- Nodes that use Docker
  - a. Obtain local images on the node.

```
docker images
```

- b. Delete the images that are not required by image ID.  
`docker rmi Image ID`

 **NOTE**

Do not delete system images such as the cce-pause image. Otherwise, pods may fail to be created.

**Solution 2: Expanding the disk capacity**

- Step 1** Expand the capacity of the data disk on the EVS console.
- Step 2** Log in to the CCE console and click the cluster. In the navigation pane, choose **Nodes**. Click **More > Sync Server Data** in the row containing the target node.
- Step 3** Log in to the target node.
- Step 4** Run the **lsblk** command to check the block device information of the node.

A data disk is divided depending on the container storage **Rootfs**:

- **Overlayfs**: No independent thin pool is allocated. Image data is stored in the **dockersys** disk.

```
lsblk
NAME MAJ:MIN RM SIZE RO TYPE MOUNTPOINT
sda 8:0 0 50G 0 disk
├─sda1 8:1 0 50G 0 part /
sdb 8:16 0 200G 0 disk
├─vgpaas-dockersys 253:0 0 90G 0 lvm /var/lib/docker # Space used by the container
engine
└─vgpaas-kubernetes 253:1 0 10G 0 lvm /mnt/paas/kubernetes/kubelet # Space used by
Kubernetes
```

Run the following commands on the node to add the new disk capacity to the **dockersys** disk:

```
pvresize /dev/sdb
lvextend -l+100%FREE -n vgpaas/dockersys
resize2fs /dev/vgpaas/dockersys
```

- **Devicemapper**: A thin pool is allocated to store image data.

```
lsblk
NAME MAJ:MIN RM SIZE RO TYPE MOUNTPOINT
sda 8:0 0 50G 0 disk
├─sda1 8:1 0 50G 0 part /
sdb 8:16 0 200G 0 disk
├─vgpaas-dockersys 253:0 0 18G 0 lvm /var/lib/docker
├─vgpaas-thinpool_tmeta 253:1 0 3G 0 lvm
├─vgpaas-thinpool 253:3 0 67G 0 lvm # Thin pool space.
...
├─vgpaas-thinpool_tdata 253:2 0 67G 0 lvm
├─vgpaas-thinpool 253:3 0 67G 0 lvm
...
└─vgpaas-kubernetes 253:4 0 10G 0 lvm /mnt/paas/kubernetes/kubelet
```

- Run the following commands on the node to add the new disk capacity to the **thinpool** disk:

```
pvresize /dev/sdb
lvextend -l+100%FREE -n vgpaas/thinpool
```

- Run the following commands on the node to add the new disk capacity to the **dockersys** disk:

```
pvresize /dev/sdb
lvextend -l+100%FREE -n vgpaas/dockersys
resize2fs /dev/vgpaas/dockersys
```

----End

## 21.8 Networking

### 21.8.1 Planning CIDR Blocks for a Cluster

Before creating a cluster on CCE, determine the number of VPCs, number of subnets, container CIDR blocks, and Services for access based on service requirements.

This topic describes the addresses in a CCE cluster in a VPC and how to plan CIDR blocks.

#### Constraints

To access a CCE cluster through a VPN, ensure that the VPN does not conflict with the VPC CIDR block where the cluster resides and the container CIDR block.

#### Basic Concepts

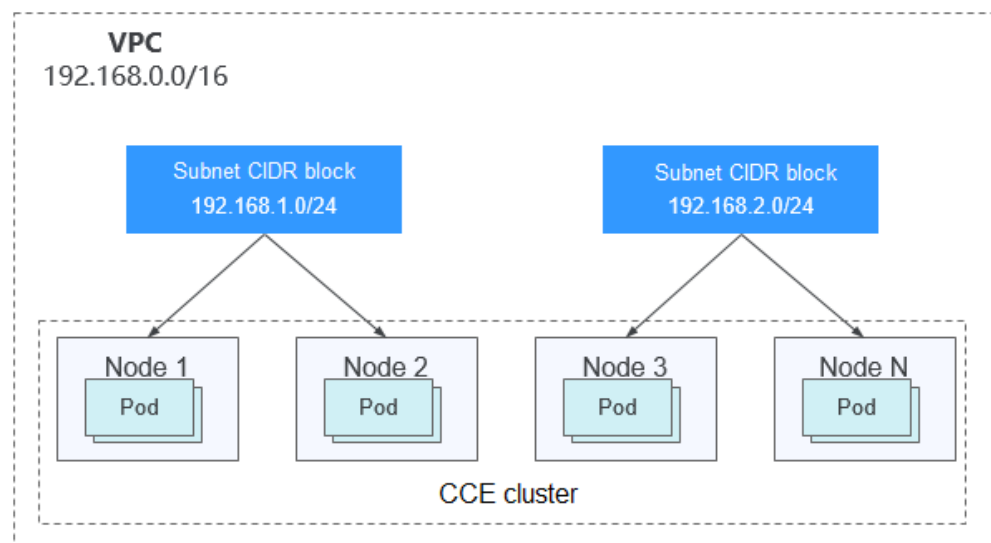
- **VPC CIDR Block**

Virtual Private Cloud (VPC) enables you to provision logically isolated, configurable, and manageable virtual networks for cloud servers, cloud containers, and cloud databases. You have complete control over your virtual network, including selecting your own CIDR block, creating subnets, and configuring security groups. You can also assign EIPs and allocate bandwidth in your VPC for secure and easy access to your business system.

- **Subnet CIDR Block**

A subnet is a network that manages ECS network planes. It supports IP address management and DNS. The IP addresses of all ECSs in a subnet belong to the subnet.

Figure 21-6 VPC CIDR block architecture



By default, ECSs in all subnets of the same VPC can communicate with one another, while ECSs in different VPCs cannot communicate with each other.

You can create a peering connection on VPC to enable ECSs in different VPCs to communicate with each other.

- **Container (Pod) CIDR Block**

Pod is a Kubernetes concept. Each pod has an IP address.

When creating a cluster on CCE, you can specify the pod (container) CIDR block, which cannot overlap with the subnet CIDR block. For example, if the subnet CIDR block is 192.168.0.0/16, the container CIDR block cannot be 192.168.0.0/18 or 192.168.1.0/18, because these addresses are included in 192.168.0.0/16.

- **Container Subnet (Only for CCE Turbo Clusters)**

In a CCE Turbo cluster, a container is assigned an IP address from the CIDR block of a VPC. The container subnet can overlap with the subnet CIDR block. Note that the subnet you select determines the maximum number of pods in the cluster. After a cluster is created, you can only add container subnets but cannot delete them.

- **Service CIDR Block**

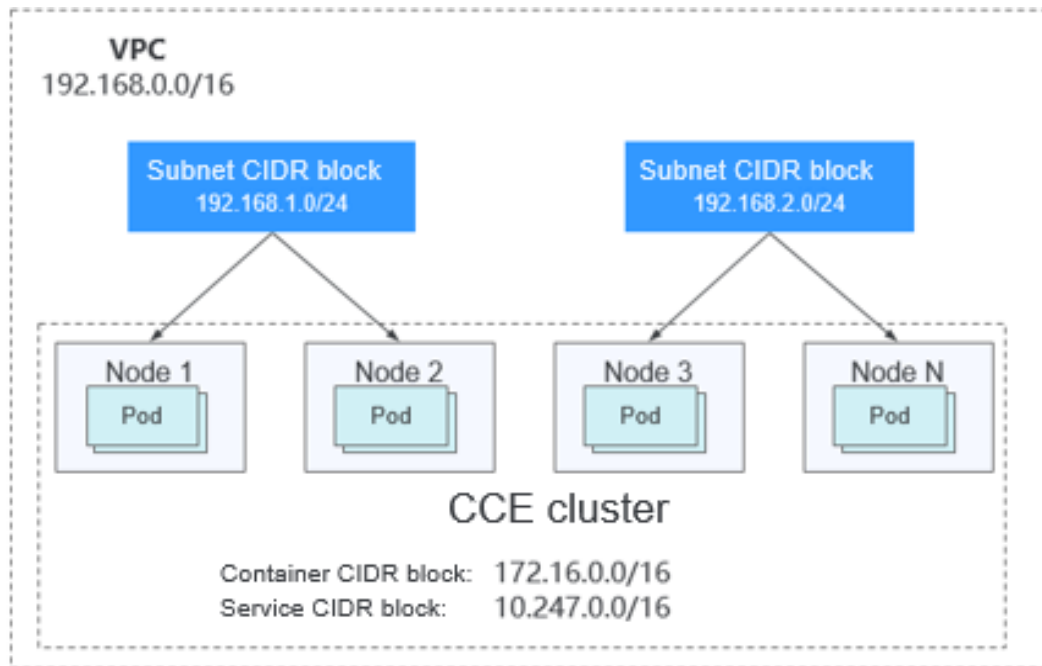
Service is also a Kubernetes concept. Each Service has an address. When creating a cluster on CCE, you can specify the Service CIDR block. Similarly, the Service CIDR block cannot overlap with the subnet CIDR block or the container CIDR block. The Service CIDR block can be used only within a cluster.

## Single-VPC Single-Cluster Scenarios

**CCE Clusters:** include clusters in VPC network model and container tunnel network model. [Figure 21-7](#) shows the CIDR block planning of a cluster.

- VPC CIDR Block: specifies the VPC CIDR block where the cluster resides. The size of this CIDR block affects the maximum number of nodes that can be created in the cluster.
- Subnet CIDR Block: specifies the subnet CIDR block where the node in the cluster resides. The subnet CIDR block is included in the VPC CIDR block. Different nodes in the same cluster can be allocated to different subnet CIDR blocks.
- Container CIDR Block: cannot overlap with the subnet CIDR block.
- Service CIDR Block: cannot overlap with the subnet CIDR block or the container CIDR block.

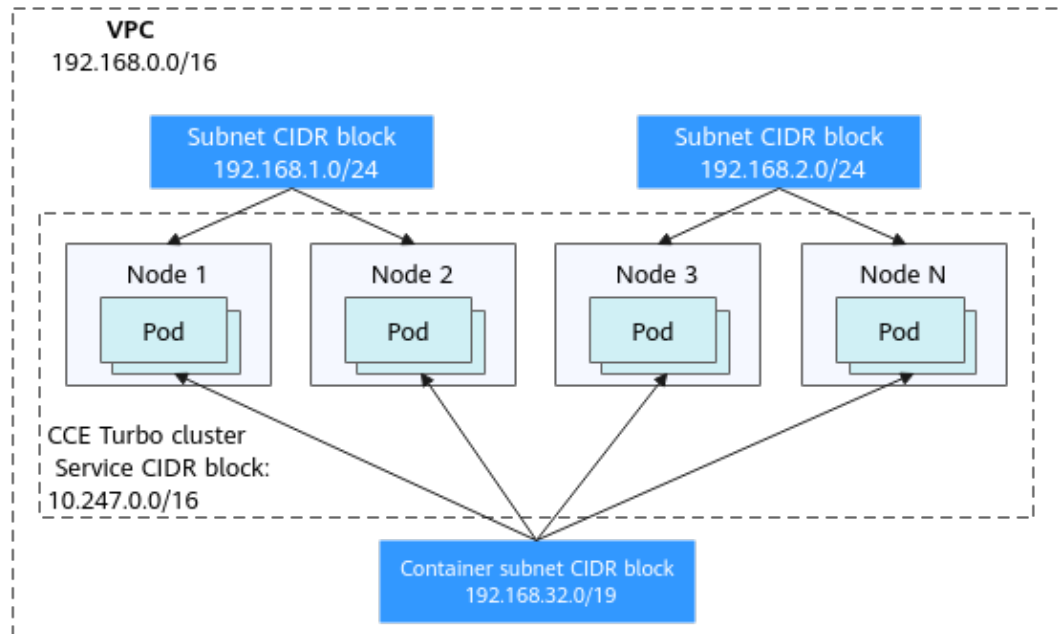
**Figure 21-7** Network CIDR block planning in single-VPC single-cluster scenarios (CCE cluster)



**Figure 21-8** shows the CIDR block planning for a **CCE Turbo cluster** (Cloud Native Network 2.0).

- VPC CIDR Block: specifies the VPC CIDR block where the cluster resides. The size of this CIDR block affects the maximum number of nodes that can be created in the cluster.
- Subnet CIDR Block: specifies the subnet CIDR block where the node in the cluster resides. The subnet CIDR block is included in the VPC CIDR block. Different nodes in the same cluster can be allocated to different subnet CIDR blocks.
- Container Subnet CIDR Block: The container subnet is included in the VPC CIDR block and can overlap with the subnet CIDR block or even be the same as the subnet CIDR block. Note that the container subnet size determines the maximum number of containers in the cluster because IP addresses in the VPC are directly allocated to containers. After a cluster is created, you can only add container subnets but cannot delete them. Set a larger IP address segment for the container subnet to prevent insufficient container IP addresses.
- Service CIDR Block: cannot overlap with the subnet CIDR block or the container CIDR block.

**Figure 21-8** Network CIDR block planning in single-VPC single-cluster scenarios (CCE Turbo Clusters)



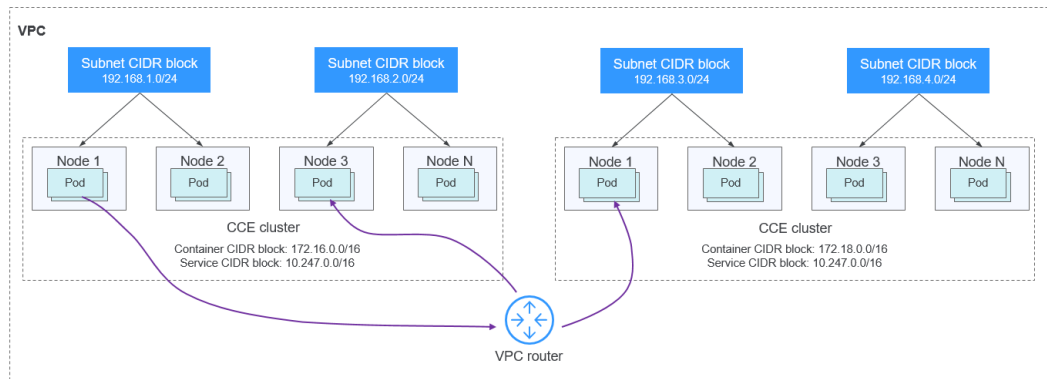
## Single-VPC Multi-Cluster Scenarios

### VPC network model

Pod packets are forwarded through VPC routes. CCE automatically configures a routing table on the VPC routes to each container CIDR block. The network scale is limited by the VPC route table. [Figure 21-9](#) shows the CIDR block planning of the cluster.

- VPC CIDR Block: specifies the VPC CIDR block where the cluster resides. The size of this CIDR block affects the maximum number of nodes that can be created in the cluster.
- Subnet CIDR Block: The subnet CIDR block in each cluster cannot overlap with the container CIDR block.
- Container CIDR Block: If multiple VPC network model clusters exist in a single VPC, the container CIDR blocks of all clusters cannot overlap because the clusters use the same routing table. In this case, CCE clusters are partially interconnected. A pod of a cluster can directly access the pods of another cluster, but cannot access the Services of the cluster.
- Service CIDR Block: can be used only in clusters. Therefore, the Service CIDR blocks of different clusters can overlap, but cannot overlap with the subnet CIDR block and container CIDR block of the cluster.

**Figure 21-9 VPC network - multi-cluster scenario**

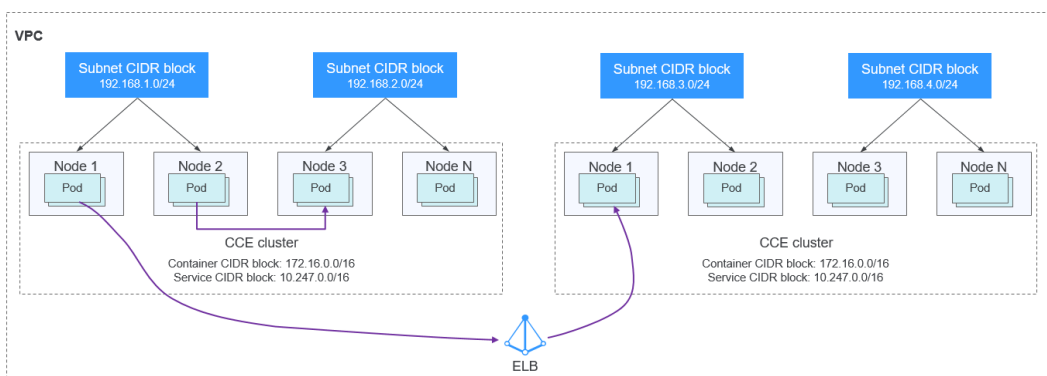


**Tunnel network model**

Though at some cost of performance, the tunnel encapsulation enables higher interoperability and compatibility with advanced features (such as network policy-based isolation), meeting the requirements of most applications. [Figure 21-10](#) shows the CIDR block planning of the cluster.

- VPC CIDR Block: specifies the VPC CIDR block where the cluster resides. The size of this CIDR block affects the maximum number of nodes that can be created in the cluster.
- Subnet CIDR Block: The subnet CIDR block in each cluster cannot overlap with the container CIDR block.
- Container CIDR Block: The container CIDR blocks of all clusters can overlap. In this case, pods in different clusters cannot be directly accessed using IP addresses. It is recommended that ELB be used for the cross-cluster access between containers.
- Service CIDR Block: can be used only in clusters. Therefore, the Service CIDR blocks of different clusters can overlap, but cannot overlap with the subnet CIDR block and container CIDR block of the cluster.

**Figure 21-10 Tunnel network - multi-cluster scenario**

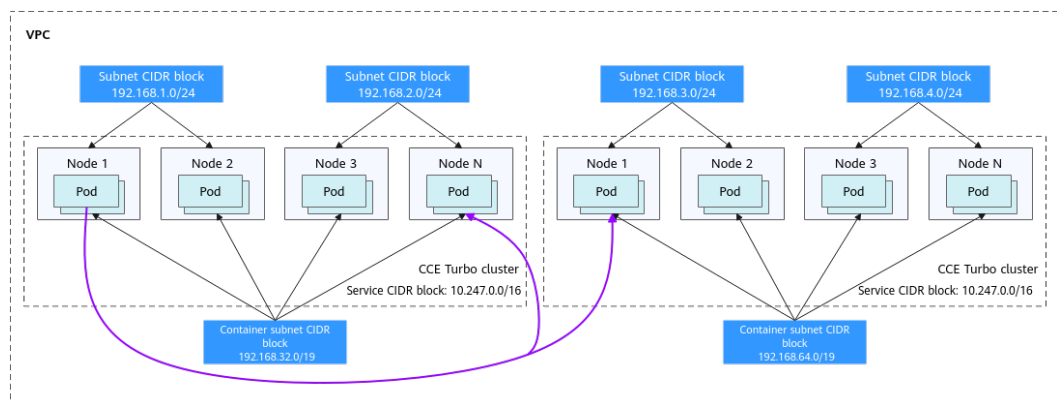


**Cloud Native Network 2.0 Model (CCE Turbo Clusters)**

In this mode, container IP addresses are allocated from the VPC CIDR block. ELB passthrough networking is supported to direct access requests to containers. Security groups and multiple types of VPC networks can be bound to deliver high performance.

- **VPC CIDR Block:** specifies the VPC CIDR block where the cluster resides. In a CCE Turbo cluster, the CIDR block size affects the total number of nodes and containers that can be created in the cluster.
- **Subnet CIDR Block:** There is no special restriction on the subnet CIDR blocks in CCE Turbo clusters.
- **Container Subnet:** The CIDR block of the container subnet is included in the VPC CIDR block. Container subnets in different clusters can overlap with each other or overlap with the subnet CIDR block. However, you are advised to stagger the container CIDR blocks of different clusters and ensure that the container subnet CIDR blocks have sufficient IP addresses. In this case, pods in different clusters can directly access each other through IP addresses.
- **Service CIDR Block:** can be used only in clusters. Therefore, the Service CIDR blocks of different clusters can overlap, but cannot overlap with the subnet CIDR block and container subnet CIDR block of the cluster.

**Figure 21-11** Cloud Native Network 2.0 - multi-cluster scenario



### Coexistence of Clusters in Multi-Network

When a VPC contains clusters created with different network models, comply with the following rules when creating a cluster:

- **VPC CIDR Block:** In this scenario, all clusters are located in the same VPC CIDR block. Ensure that there are sufficient available IP addresses in the VPC.
- **Subnet CIDR Block:** Ensure that the subnet CIDR block does not overlap with the container CIDR block. Even in some scenarios (for example, coexistence with CCE Turbo clusters), the subnet CIDR block can overlap with the container (subnet) CIDR block. However, this is not recommended.
- **Container CIDR Block:** Ensure that the container CIDR blocks of clusters in **VPC network model** do not overlap.
- **Service CIDR Block:** The Service CIDR blocks of all clusters can overlap, but cannot overlap with the subnet CIDR block and container CIDR block of the cluster.

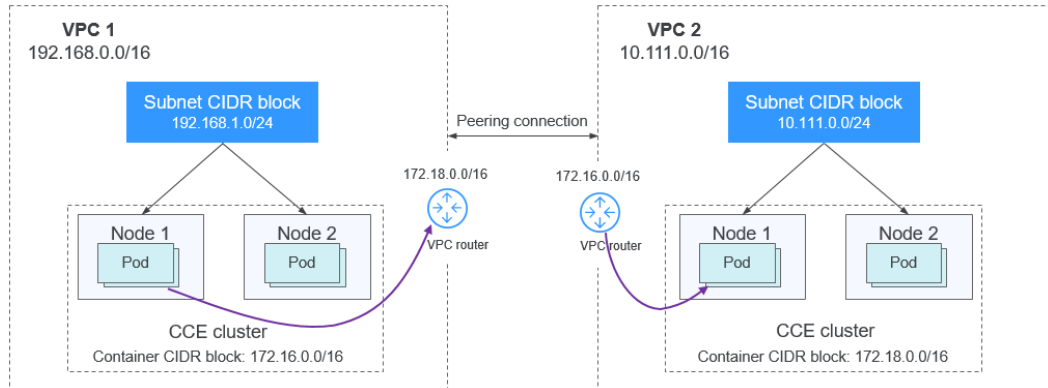
### Cross-VPC Cluster Interconnection

When two VPC networks are interconnected, you can configure the packets to be sent to the peer VPC in the route table.



In the VPC network model, after creating a peering connection, add routes for the peering connection to enable communication between the two VPCs.

**Figure 21-12** VPC network - VPC interconnection scenario

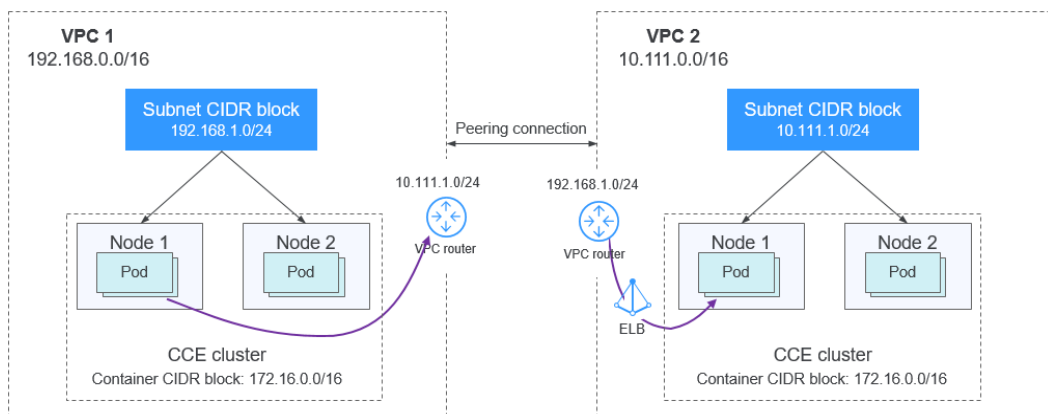


When creating a VPC peering connection between containers across VPCs, pay attention to the following points:

- The VPC to which the clusters belong must not overlap. In each cluster, the subnet CIDR block cannot overlap with the container CIDR block.
- The container CIDR blocks of clusters at both ends cannot overlap, but the Service CIDR blocks can.
- Add not only the peer VPC CIDR block but also the peer container CIDR block to the VPC routing tables at both ends. Note that this operation must be performed in the VPC route tables of the clusters.

In the Tunnel network model, after creating a peering connection, add routes for the peering connection to enable communication between the two VPCs.

**Figure 21-13** Tunnel network - VPC interconnection scenario



Pay attention to the following:

- The VPCs of the clusters must not overlap.
- The container CIDR blocks of all clusters can overlap, so do the Service CIDR blocks.

- Add the peer subnet CIDR block to the route table of the VPC peering connection.

In **Cloud Native Network 2.0** mode, after creating a VPC peering connection, you only need to add routes for the VPC peering connection to enable communication between the two VPCs. Ensure that the VPCs of the clusters do not overlap.

## VPC-IDC Scenarios

Similar to the VPC interconnection scenario, some CIDR blocks in the VPC are routed to the IDC. The pod IP addresses of CCE clusters cannot overlap with the addresses within these CIDR blocks. To access the pod IP addresses in the cluster in the IDC, configure the route table to the private line VBR on the IDC.

## 21.8.2 Selecting a Network Model

CCE uses proprietary, high-performance container networking add-ons to support the tunnel network, Cloud Native Network 2.0, and VPC network models.

---

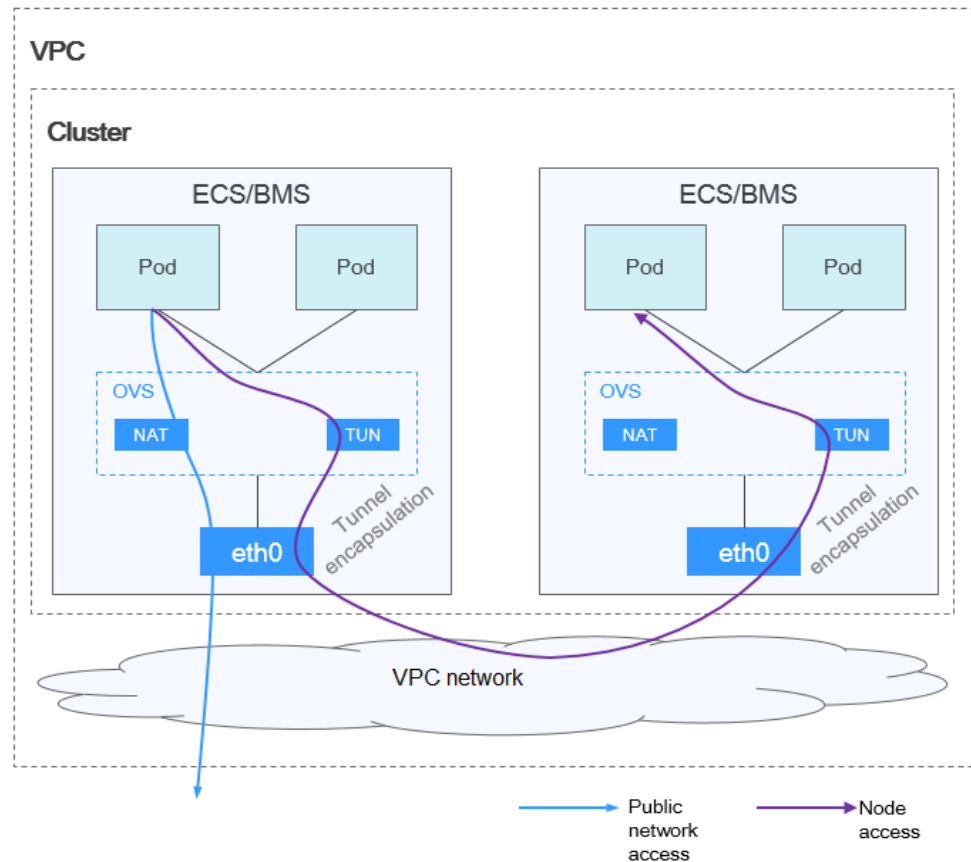
 **CAUTION**

After a cluster is created, the network model cannot be changed. Exercise caution when selecting a network model.

---

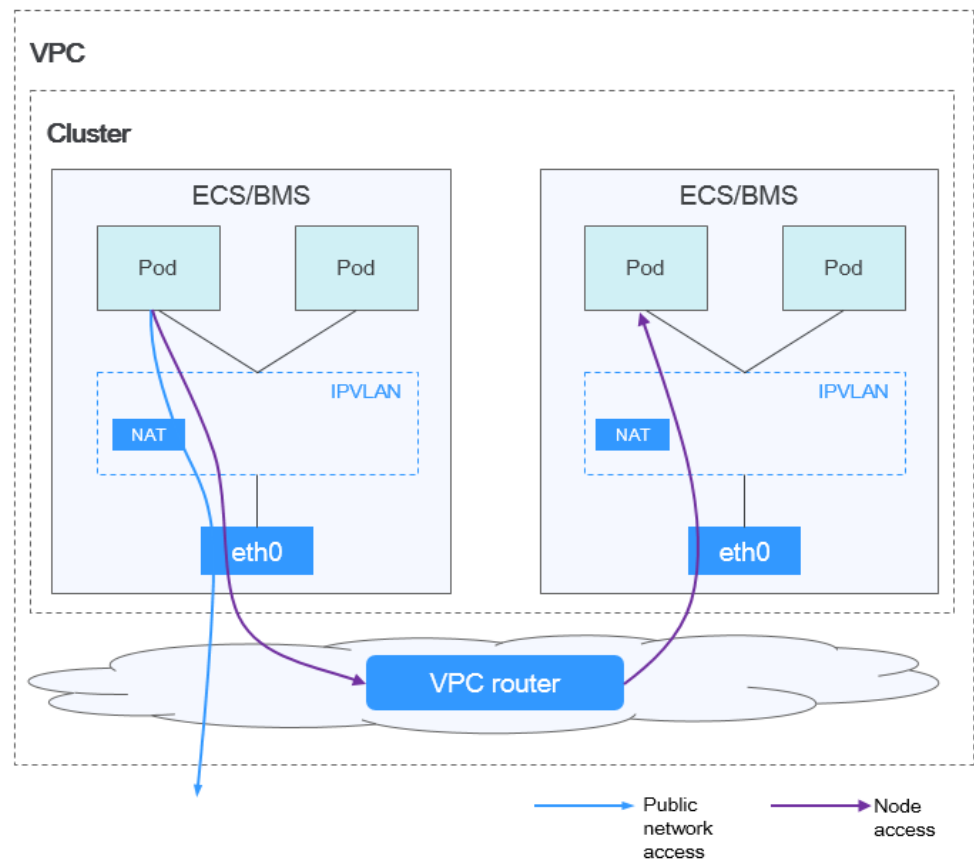
- **Tunnel network:** The container network is an overlay tunnel network on top of a VPC network and uses the VXLAN technology. This network model is applicable when there is no high requirements on performance. VXLAN encapsulates Ethernet packets as UDP packets for tunnel transmission. Though at some cost of performance, the tunnel encapsulation enables higher interoperability and compatibility with advanced features (such as network policy-based isolation), meeting the requirements of most applications.

Figure 21-14 Container tunnel network



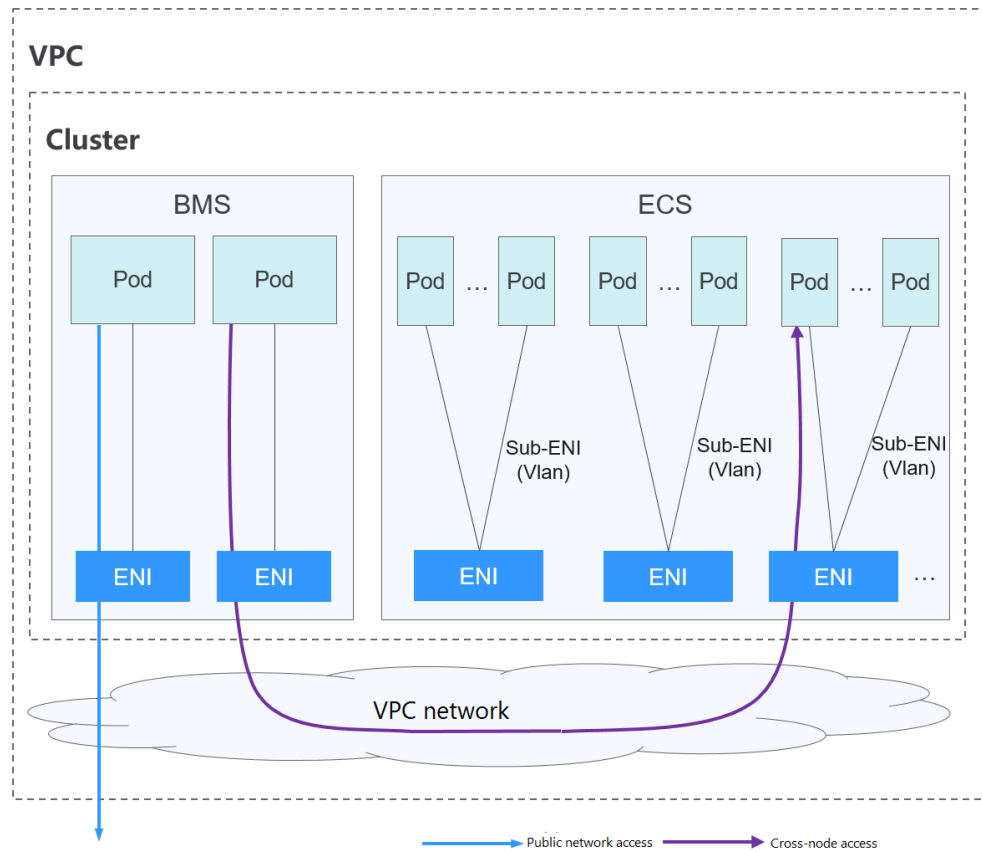
- **VPC network:** The container network uses VPC routing to integrate with the underlying network. This network model is applicable to performance-intensive scenarios. The maximum number of nodes allowed in a cluster depends on the route quota in a VPC network. Each node is assigned a CIDR block of a fixed size. VPC networks are free from tunnel encapsulation overhead and outperform container tunnel networks. In addition, as VPC routing includes routes to node IP addresses and container network segment, container pods in the cluster can be directly accessed from outside the cluster.

Figure 21-15 VPC network



- **Cloud Native Network 2.0:** The container network deeply integrates the elastic network interface (ENI) capability of VPC, uses the VPC CIDR block to allocate container addresses, and supports passthrough networking to containers through a load balancer.

**Figure 21-16** Cloud Native Network 2.0



The following table lists the differences between the network models.

**Table 21-9** Networking model comparison

Dimension	Tunnel Network	VPC Network	Cloud Native Network 2.0
Core technology	OVS	IPvlan and VPC route	VPC ENI/sub-ENI
Applicable Clusters	CCE cluster	CCE cluster	CCE Turbo cluster
Network isolation	Kubernetes native NetworkPolicy for pods	No	Pods support security group isolation.

Dimension	Tunnel Network	VPC Network	Cloud Native Network 2.0
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>	The container CIDR block is divided from the VPC subnet and does not need to be allocated separately.
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.	The container network is integrated with the VPC network, eliminating performance loss.
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.	A maximum of 2,000 nodes are supported.

Dimension	Tunnel Network	VPC Network	Cloud Native Network 2.0
Application Scenarios	<ul style="list-style-type: none"> <li>• Common container services</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 communicate with VMs using a microservice registration framework, such as Dubbo and CSE.</li> </ul>	<ul style="list-style-type: none"> <li>• Scenarios that have high requirements on network latency, bandwidth, and performance</li> <li>• Containers communicate with VMs using a microservice registration framework, such as Dubbo and CSE.</li> </ul>

**NOTICE**

1. The scale of a cluster that uses the VPC network model is limited by the custom routes of the VPC. Therefore, you need to estimate the number of required nodes before creating a cluster.
2. By default, VPC routing network supports direct communication between containers and hosts in the same VPC. If a peering connection policy is configured between the VPC and another VPC, the containers can directly communicate with hosts on the peer VPC. In addition, in hybrid networking scenarios such as Direct Connect and VPN, communication between containers and hosts on the peer end can also be achieved with proper planning.

### 21.8.3 Implementing Sticky Session Through Load Balancing

#### Concepts

Session persistence is one of the most common while complex problems in load balancing.

Session persistence is also called sticky sessions. After the sticky session function is enabled, requests from the same client are distributed to the same backend ECS by the load balancer for better continuity.

In load balancing and sticky session, connection and session are two key concepts. When only load balancing is concerned, session and connection refer to the same thing.

Simply put, if a user needs to log in, it can be regarded as a session; otherwise, a connection.

The sticky session mechanism fundamentally conflicts with the basic functions of load balancing. A load balancer forwards requests from clients to multiple

backend servers to avoid overload on a single server. However, sticky session requires that some requests be forwarded to the same server for processing. Therefore, select a proper sticky session mechanism based on the application environment.

## Layer-4 Load Balancing (Service)

In layer-4 load balancing, source IP address-based sticky session (Hash routing based on the client IP address) can be enabled. To enable source IP address-based sticky session on Services, the following conditions must be met:

### CCE clusters

1. **Service Affinity** of the Service is set to **Node level** (that is, the value of the **externalTrafficPolicy** field of the Service is **Local**).
2. Enable the source IP address-based sticky session in the load balancing configuration of the Service.

```
apiVersion: v1
kind: Service
metadata:
 name: svc-example
 namespace: default
 annotations:
 kubernetes.io/elb.class: union
 kubernetes.io/elb.id: 56dcc1b4-8810-480c-940a-a44f7736f0dc
 kubernetes.io/elb.lb-algorithm: ROUND_ROBIN
 kubernetes.io/elb.session-affinity-mode: SOURCE_IP
spec:
 selector:
 app: nginx
 externalTrafficPolicy: Local
 ports:
 - name: cce-service-0
 targetPort: 80
 nodePort: 32633
 port: 80
 protocol: TCP
 type: LoadBalancer
```

3. Anti-affinity is enabled for the backend application corresponding to the Service.

## Layer-7 Load Balancing (Ingress)

In layer-7 load balancing, sticky session based on HTTP cookies and app cookies can be enabled. To enable such sticky session, the following conditions must be met:

1. The application (workload) corresponding to the ingress is enabled with workload anti-affinity.
2. Node affinity is enabled for the Service corresponding to the ingress.

### Procedure

#### Step 1 Create an Nginx workload.

Set the number of pods to 3 and set the podAntiAffinity.

```
kind: Deployment
apiVersion: apps/v1
metadata:
 name: nginx
 namespace: default
```



```
spec:
 replicas: 3
 selector:
 matchLabels:
 app: nginx
 template:
 metadata:
 labels:
 app: nginx
 spec:
 containers:
 - name: container-0
 image: 'nginx:perl'
 resources:
 limits:
 cpu: 250m
 memory: 512Mi
 requests:
 cpu: 250m
 memory: 512Mi
 imagePullSecrets:
 - name: default-secret
 affinity:
 podAntiAffinity:
 # Pod anti-affinity.
 requiredDuringSchedulingIgnoredDuringExecution:
 - labelSelector:
 matchExpressions:
 - key: app
 operator: In
 values:
 - nginx
 topologyKey: kubernetes.io/hostname
```

## Step 2 Creating a NodePort Service

Configure the sticky session in a Service. An ingress can connect to multiple Services, and each Service can have different sticky sessions.

```
apiVersion: v1
kind: Service
metadata:
 name: nginx
 namespace: default
 annotations:
 kubernetes.io/elb.lb-algorithm: ROUND_ROBIN
 kubernetes.io/elb.session-affinity-mode: HTTP_COOKIE # HTTP cookie type.
 kubernetes.io/elb.session-affinity-option: '{"persistence_timeout":"1440"}' # Session stickiness duration,
in minutes. The value ranges from 1 to 1440.
spec:
 selector:
 app: nginx
 ports:
 - name: cce-service-0
 protocol: TCP
 port: 80
 targetPort: 80
 nodePort: 32633 # Node port number.
 type: NodePort
 externalTrafficPolicy: Local # Node-level forwarding.
```

You can also select **APP\_COOKIE**.

```
apiVersion: v1
kind: Service
metadata:
 name: nginx
 namespace: default
 annotations:
 kubernetes.io/elb.lb-algorithm: ROUND_ROBIN
 kubernetes.io/elb.session-affinity-mode: APP_COOKIE # Select APP_COOKIE.
```

```
kubernetes.io/elb.session-affinity-option: '{"app_cookie_name":"test"}' # Application cookie name.
```

### Step 3 Create an ingress and associate it with a Service.

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
 name: ingress-test
 namespace: default
 annotations:
 kubernetes.io/elb.class: union
 kubernetes.io/elb.port: '80'
 kubernetes.io/elb.autocreate:
 '{
 "type":"public",
 "bandwidth_name":"cce-bandwidth-test",
 "bandwidth_chargemode":"traffic",
 "bandwidth_size":1,
 "bandwidth_sharetype":"PER",
 "eip_type":"5_bgp"
 }'
spec:
 rules:
 - host: 'www.example.com'
 http:
 paths:
 - path: '/'
 backend:
 service:
 name: nginx #Service name
 port:
 number: 80
 property:
 ingress.beta.kubernetes.io/url-match-mode: STARTS_WITH
 pathType: ImplementationSpecific
 ingressClassName: cce
```

### Step 4 Log in to the ELB console, access the load balancer details page, and check whether the sticky session feature is enabled.

----End

## 21.8.4 Obtaining the Client Source IP Address for a Container

### Background

There may be different types of proxy servers between a client and a container server. How can a container obtain the real source IP address of the client? This section describes several scenarios you may encounter.

### Principles

#### Layer-7 forwarding:

Ingress: If this access mode is used, the client source IP address is saved in the **X-Forwarded-For** HTTP header field by default. No other configuration is required.

- ELB ingress: A self-developed ingress to implement layer-7 network access between the internet and intranet (in the same VPC) based on ELB. If the backend Service type is **NodePort**, set **Service Affinity** to **Node level**.
- Nginx ingress: An ingress that implements layer-7 networking based on the nginx-ingress add-on. The backend Service type can be either **ClusterIP** or

**NodePort.** If the backend Service type is **NodePort**, set **Service Affinity** to **Node level**.

**Layer-4 forwarding:**

- **NodePort:** In this access mode, the container port is mapped to the node port. If cluster-level affinity is configured, access requests will be forwarded through the node and the client source IP address cannot be obtained. If node-level affinity is configured, access requests are not forwarded and the client source IP address can be obtained.

## Ingress

Configure the application server and obtain the IP address of a client from the HTTP header.

The real IP address is placed in the **X-Forwarded-For** HTTP header field by the load balancer in the following format:

*X-Forwarded-For: IP address of the client,Proxy server 1-IP address,Proxy server 2-IP address,...*

If you use this method, the first IP address obtained is the IP address of the client.

 **NOTE**

- When adding an ingress, if the backend service is of the NodePort type, set **Service Affinity** to **Node level**, that is, set **spec.externalTrafficPolicy** to **Local**. For details, see **NodePort**.

**Step 1** Take the Nginx workload as an example. Before configuring the source IP address, obtain the access logs. **nginx-c99fd67bb-ghv4q** indicates the pod name.


```
kubectl logs nginx-c99fd67bb-ghv4q
```

Information similar to the following is displayed:

```
...
10.0.0.7 - - [17/Aug/2023:01:30:11 +0000] "GET / HTTP/1.1" 200 19 "http://114.114.114.114:9421/"
"Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/115.0.0.0
Safari/537.36 Edg/115.0.1901.203" "100.125.**.*"
```

**100.125.\*\*.\*** specifies the CIDR block of the load balancer, indicating that the traffic is forwarded through the load balancer.

**Step 2** Go to the ELB console and enable the function of obtaining the client IP address of the listener corresponding to the load balancer. **Transparent transmission of source IP addresses is enabled for dedicated load balancers by default. You do not need to manually enable this function.**

1. Log in to the ELB console.
2. Click  in the upper left corner of the management console and select a region and a project.
3. Click **Service List**. Under **Networking**, click **Elastic Load Balance**.
4. On the **Load Balancers** page, click the name of the load balancer.
5. Switch to the **Listeners** tab and click **Modify** on the right of the target listener. If modification protection exists, disable the protection on the basic information page of the listener and try again.
6. Enable **Obtain Client IP Address**.

**Step 3** (Perform this step only for nginx-ingress.) Edit the nginx-ingress add-on. In the nginx configuration parameter area, configure the configuration fields and information. (For details about the parameter range, see [community document](#).) After the configuration is complete, update the add-on.

```
{
 "enable-real-ip": "true",
 "forwarded-for-header": "X-Forwarded-For",
 "proxy-real-ip-cidr": "100.125.0.0/16",
 "keep-alive-requests": "100"
}
```

 **NOTE**

The **proxy-real-ip-cidr** parameter indicates the CIDR block of the proxy server.

- For shared load balancers, add CIDR block 100.125.0.0/16 (reserved only for load balancers and therefore, there is no risk) and the high-defense CIDR block.
- For dedicated load balancers, add the CIDR block of the VPC subnet where the ELB resides.

**Step 4** Access the workload again and view the new access log.

```
...
10.0.0.7 - - [17/Aug/2023:02:43:11 +0000] "GET / HTTP/1.1" 304 0 "http://114.114.114.114:9421/"
"Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/115.0.0.0
Safari/537.36 Edg/115.0.1901.203" "124.**.**"
```

The source IP address of the client is obtained.

----End

## NodePort

Set the service affinity of a NodePort Service to **Node level** instead of **Cluster level**. That is, set **spec.externalTrafficPolicy** of the Service to **Local**.

## 21.9 Storage

### 21.9.1 Expanding the Storage Space

The storage classes that can be expanded for CCE nodes are as follows:

**Table 21-10** Capacity expansion methods

Type	Name	Purpose	Capacity Expansion Method
Node disk	System disk	A disk attached to a node for installing the operating system	<a href="#">Expanding System Disk Capacity</a>

Type	Name	Purpose	Capacity Expansion Method
	Data disk	A disk that must be attached to a node for the container engine and kubelet	<ul style="list-style-type: none"> <li>Expanding the Capacity of a Data Disk Used by Container Engines</li> <li>Expanding the Capacity of a Data Disk Used by kubelet</li> </ul>
Container storage	Pod container space	The base size of a container, which is, the upper limit of the disk space occupied by each pod (including the storage space occupied by container images)	Expanding the Capacity of a Data Disk Used by Pod (basesize)
	PVC	Storage resources mounted to the containers	Expanding a PVC

## Expanding System Disk Capacity

EulerOS 2.9 is used as the sample OS. There is only one partition (`/dev/vda1`) with a capacity of 50 GiB in the system disk `/dev/vda`, and then 50 GiB is added to the system disk. In this example, the additional 50 GiB is allocated to the existing `/dev/vda1` partition.

**Step 1** Expand the capacity of the system disk on the EVS console.

**Step 2** Log in to the node and run the `growpart` command to check whether `growpart` has been installed.

If the tool operation guide is displayed, the `growpart` has been installed. Otherwise, run the following command to install `growpart`:

```
yum install cloud-utils-growpart
```

**Step 3** Run the following command to view the total capacity of the system disk `/dev/vda`:

```
fdisk -l
```

If the following information is displayed, the total capacity of `/dev/vda` is 100 GiB.

```
[root@test-48162 ~]# fdisk -l
Disk /dev/vda: 100 GiB, 107374182400 bytes, 209715200 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disklabel type: dos
Disk identifier: 0x78d88f0b

Device Boot Start End Sectors Size Id Type
/dev/vda1 * 2048 104857566 104855519 50G 83 Linux

Disk /dev/vdb: 100 GiB, 107374182400 bytes, 209715200 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
```

I/O size (minimum/optimal): 512 bytes / 512 bytes

Disk /dev/mapper/vgpaas-dockersys: 90 GiB, 96632569856 bytes, 188735488 sectors  
Units: sectors of 1 \* 512 = 512 bytes  
Sector size (logical/physical): 512 bytes / 512 bytes  
I/O size (minimum/optimal): 512 bytes / 512 bytes

Disk /dev/mapper/vgpaas-kubernetes: 10 GiB, 10733223936 bytes, 20963328 sectors  
Units: sectors of 1 \* 512 = 512 bytes  
Sector size (logical/physical): 512 bytes / 512 bytes  
I/O size (minimum/optimal): 512 bytes / 512 bytes

**Step 4** Run the following command to check the capacity of the system disk partition **/dev/vda1**:

```
df -TH
```

Information similar to the following is displayed:

```
[root@test-48162 ~]# df -TH
Filesystem Type Size Used Avail Use% Mounted on
devtmpfs devtmpfs 1.8G 0 1.8G 0% /dev
tmpfs tmpfs 1.8G 0 1.8G 0% /dev/shm
tmpfs tmpfs 1.8G 13M 1.8G 1% /run
tmpfs tmpfs 1.8G 0 1.8G 0% /sys/fs/cgroup
/dev/vda1 ext4 53G 3.3G 47G 7% /
tmpfs tmpfs 1.8G 75M 1.8G 5% /tmp
/dev/mapper/vgpaas-dockersys ext4 95G 1.3G 89G 2% /var/lib/docker
/dev/mapper/vgpaas-kubernetes ext4 11G 39M 10G 1% /mnt/paas/kubernetes/kubelet
...
```

**Step 5** Run the following command to extend the partition using growpart:

```
growpart System disk Partition number
```

The partition number is **1** because there is only one **/dev/vda1** partition in the system disk, as shown in the following command:

```
growpart /dev/vda 1
```

Information similar to the following is displayed:

```
CHANGED: partition=1 start=2048 old: size=104855519 end=104857567 new: size=209713119 end=209715167
```

**Step 6** Run the following command to extend the file system:

```
resize2fs Disk partition
```

An example command is as follows:

```
resize2fs /dev/vda1
```

Information similar to the following is displayed:

```
resize2fs 1.45.6 (20-Mar-2020)
Filesystem at /dev/vda1 is mounted on /; on-line resizing required
old_desc_blocks = 7, new_desc_blocks = 13
The filesystem on /dev/vda1 is now 26214139 (4k) blocks long.
```

**Step 7** Run the following command to view the new capacity of the **/dev/vda1** partition:

```
df -TH
```

Information similar to the following is displayed:

```
[root@test-48162 ~]# df -TH
Filesystem Type Size Used Avail Use% Mounted on
devtmpfs devtmpfs 1.8G 0 1.8G 0% /dev
tmpfs tmpfs 1.8G 0 1.8G 0% /dev/shm
tmpfs tmpfs 1.8G 13M 1.8G 1% /run
tmpfs tmpfs 1.8G 0 1.8G 0% /sys/fs/cgroup
```

```

/dev/vda1 ext4 106G 3.3G 98G 4% /
tmpfs tmpfs 1.8G 75M 1.8G 5% /tmp
/dev/mapper/vgpaas-dockersys ext4 95G 1.3G 89G 2% /var/lib/docker
/dev/mapper/vgpaas-kubernetes ext4 11G 39M 10G 1% /mnt/paas/kubernetes/kubelet
...

```

**Step 8** Log in to the CCE console and click the cluster. In the navigation pane, choose **Nodes**. Click **More > Sync Server Data** in the row containing the target node.

----End

## Expanding the Capacity of a Data Disk Used by Container Engines

CCE divides the data disk space for two parts by default. One part is used to store the Docker/containerd working directories, container images, 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. This section uses Docker as an example to describe how to expand the container engine capacity.

**Step 1** Expand the capacity of the data disk on the EVS console.

**Step 2** Log in to the CCE console and click the cluster. In the navigation pane, choose **Nodes**. Click **More > Sync Server Data** in the row containing the target node.

**Step 3** Log in to the target node.

**Step 4** Run the **lsblk** command to check the block device information of the node.

A data disk is divided depending on the container storage **Rootfs**:

- **Overlays:** No independent thin pool is allocated. Image data is stored in the **dockersys** disk.

```

lsblk
NAME MAJ:MIN RM SIZE RO TYPE MOUNTPOINT
sda 8:0 0 50G 0 disk
├─sda1 8:1 0 50G 0 part /
└─sdb 8:16 0 200G 0 disk
 └─vgpaas-dockersys 253:0 0 90G 0 lvm /var/lib/docker # Space used by the container
 engine
 └─vgpaas-kubernetes 253:1 0 10G 0 lvm /mnt/paas/kubernetes/kubelet # Space used by
 Kubernetes

```

Run the following commands on the node to add the new disk capacity to the **dockersys** disk:

```

pvresize /dev/sdb
lvextend -l+100%FREE -n vgpaas/dockersys
resize2fs /dev/vgpaas/dockersys

```

- **Devicemapper:** A thin pool is allocated to store image data.

```

lsblk
NAME MAJ:MIN RM SIZE RO TYPE MOUNTPOINT
sda 8:0 0 50G 0 disk
├─sda1 8:1 0 50G 0 part /
└─sdb 8:16 0 200G 0 disk
 └─vgpaas-dockersys 253:0 0 18G 0 lvm /var/lib/docker
 └─vgpaas-thinpool_tmeta 253:1 0 3G 0 lvm
 └─vgpaas-thinpool 253:3 0 67G 0 lvm # Thin pool space.
 ...
 └─vgpaas-thinpool_tdata 253:2 0 67G 0 lvm
 └─vgpaas-thinpool 253:3 0 67G 0 lvm
 ...
 └─vgpaas-kubernetes 253:4 0 10G 0 lvm /mnt/paas/kubernetes/kubelet

```

- Run the following commands on the node to add the new disk capacity to the **thinpool** disk:

```
pvresize /dev/sdb
lvextend -l+100%FREE -n vgpaas/thinpool
```

- Run the following commands on the node to add the new disk capacity to the **dockersys** disk:

```
pvresize /dev/sdb
lvextend -l+100%FREE -n vgpaas/dockersys
resize2fs /dev/vgpaas/dockersys
```

----End

## Expanding the Capacity of a Data Disk Used by kubelet

CCE divides the data disk space for two parts by default. One part is used to store the Docker/containerd working directories, container images, and image metadata. The other is reserved for kubelet and emptyDir volumes. To expand the kubelet space, perform the following steps:

- Step 1** Expand the capacity of the data disk on the EVS console.
- Step 2** Log in to the CCE console and click the cluster. In the navigation pane, choose **Nodes**. Click **More > Sync Server Data** in the row containing the target node.
- Step 3** Log in to the target node.
- Step 4** Run the following commands on the node to add the new disk capacity to the Kubernetes disk:

```
pvresize /dev/sdb
lvextend -l+100%FREE -n vgpaas/kubernetes
resize2fs /dev/vgpaas/kubernetes
```

----End

## Expanding the Capacity of a Data Disk Used by Pod (basesize)

- Step 1** Log in to the CCE console and click the name of the target cluster in the cluster list.
- Step 2** Choose **Nodes** from the navigation pane.
- Step 3** Select the target node and choose **More > Reset Node** in the **Operation** column.

### NOTICE

Resetting a node may make the node-specific resources (such as local storage and workloads scheduled to this node) unavailable. Exercise caution when performing this operation to avoid impact on running services.

- Step 4** Reconfigure node parameters.

If you need to adjust the container storage space, pay attention to the following configurations:

**Storage Settings:** Click **Expand** next to the data disk to set the following parameters:



- **Allocate Disk Space:** storage space used by the container engine to store the Docker/containerd working directory, container image data, and image metadata. Defaults to 90% of the data disk.
- **Allocate Pod Basesize:** CCE allows you to 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 be less than or equal to 80% of the container engine space.

 **NOTE**

- The capability of customizing pod basesize is related to the node OS and container storage rootfs.
  - When the rootfs uses Device Mapper, the node supports custom pod basesize. The default storage space of a single container is 10 GiB.
  - When the rootfs uses OverlayFS, most nodes do not support custom pod basesize. The storage space of a single container is not limited and defaults to the container engine space.  
Only nodes running EulerOS 2.9 in clusters of 1.19.16, 1.21.3, 1.23.3, and later versions support custom pod basesize.
- In the case of using Docker on nodes running EulerOS 2.9, **basesize** will not take effect if **CAP\_SYS\_RESOURCE** or **privileged** is configured for a container.

**Step 5** After the node is reset, log in to the node and run the following command to access the container and check whether the container storage capacity has been expanded.

**docker exec -it *container\_id* /bin/sh** or **kubectl exec -it *container\_id* /bin/sh**

**df -h**

```
df -h
Filesystem Size Used Avail Use% Mounted on
/dev/mapper/docker-253:1-787293-631c1bde2cbe82e39f32253b216ba914cb183b168b54700b3e5b9a54ee40abd1 15G 229M 15G 2% /
tmpfs 32G 0 32G 0% /dev
tmpfs 32G 0 32G 0% /sys/fs/cgroup
/dev/mapper/vgpaas-kubernetes 9.8G 37M 9.2G 1% /etc/hosts
/dev/vda1 40G 5.2G 33G 14% /etc/hostname
shm 64M 0 64M 0% /dev/shm
tmpfs 32G 16K 32G 1% /run/secrets/kubernetes.io/serviceaccount
tmpfs 32G 0 32G 0% /proc/acpi
tmpfs 32G 0 32G 0% /sys/firmware
tmpfs 32G 0 32G 0% /proc/scsi
tmpfs 32G 0 32G 0% /proc/kbox
tmpfs 32G 0 32G 0% /proc/oom_extend
```

----End

## Expanding a PVC

Cloud storage:

- OBS and SFS: There is no storage restriction and capacity expansion is not required.
- EVS:
  - You can expand the capacity of automatically created pay-per-use volumes on the console. The procedure is as follows:
    - Choose **Storage** in the navigation pane and click the **PersistentVolumeClaims (PVCs)** tab. Locate the row containing the target PVC and choose **More > Scale-out** in the **Operation** column.
    - Enter the capacity to be added and click **OK**.

- For SFS Turbo, expand the capacity on the SFS console and then change the capacity in the PVC.

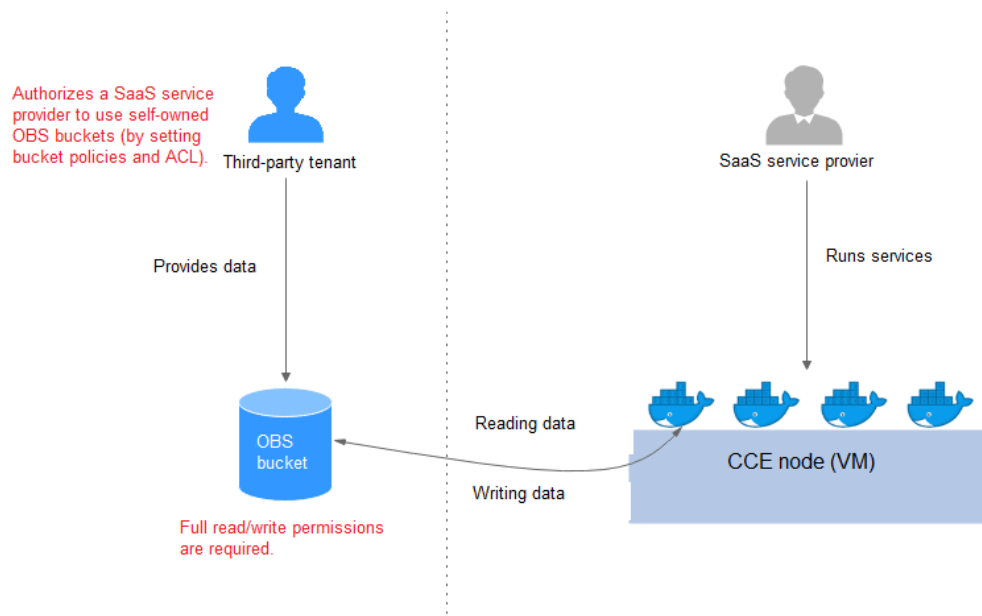
## 21.9.2 Mounting an Object Storage Bucket of a Third-Party Tenant

This section describes how to mount OBS buckets and OBS parallel file systems (preferred) of third-party tenants.

### Application Scenarios

The CCE cluster of a SaaS service provider needs to be mounted with the OBS bucket of a third-party tenant, as shown in [Figure 21-17](#).

**Figure 21-17** Mounting an OBS bucket of a third-party tenant



1. **The third-party tenant authorizes the SaaS service provider to access the OBS buckets or parallel file systems** by setting the bucket policy and bucket ACL.
2. **The SaaS service provider statically imports the OBS buckets and parallel file systems of the third-party tenant.**
3. The SaaS service provider processes the service and writes the processing result (result file or result data) back to the OBS bucket of the third-party tenant.

### Precautions

- Only parallel file systems and OBS buckets of third-party tenants in the same region can be mounted.
- Only clusters where the everest add-on of v1.1.11 or later has been installed (the cluster version must be v1.15 or later) can be mounted with OBS buckets of third-party tenants.

- The service platform of the SaaS service provider needs to manage the lifecycle of the third-party bucket PVs. When a PVC is deleted separately, the PV is not deleted. Instead, it will be retained. To do so, call the native Kubernetes APIs to create and delete static PVs.

## Authorizing the SaaS Service Provider to Access the OBS Buckets

The following uses an OBS bucket as an example to describe how to set a bucket policy and bucket ACL to authorize the SaaS service provider. The configuration for an OBS parallel file system is the same.

**Step 1** Log in to the OBS console.

**Step 2** In the bucket list, click a bucket name to access the **Overview** page.

**Step 3** In the navigation pane, choose **Permissions**. Click the **Bucket Policies** tab and click **Create Bucket Policy** under **Custom Bucket Policies**.

- **Policy Mode:** Select **Customized**.
- **Effect:** Select **Allow**.
- **Principal:** Select **Include**, select **Cloud service user**, and enter the account ID and user ID. The bucket policy is applied to the specified user.
- **Resources:** Select the resources that can be operated.
- **Actions:** Select the actions that can be operated.

**Step 4** In the navigation pane, choose **Permissions > Bucket ACLs**. In the right pane, click **Add**. Enter the account ID or account name of the authorized user, select **Read** and **Write** for **Access to Bucket**, select **Read** and **Write** for **Access to ACL**, and click **OK**.

----End

## Statically Importing OBS Buckets and Parallel File Systems

- **Static PV of an OBS bucket:**

```

apiVersion: v1
kind: PersistentVolume
metadata:
 name: objbucket #Replace the name with the actual PV name of the bucket.
 annotations:
 pv.kubernetes.io/provisioned-by: everest-csi-provisioner
spec:
 accessModes:
 - ReadWriteMany
 capacity:
 storage: 1Gi
 mountOptions:
 - default_acl=bucket-owner-full-control #New OBS mounting parameters
 csi:
 driver: obs.csi.everest.io
 fsType: s3fs
 volumeAttributes:
 everest.io/obs-volume-type: STANDARD
 everest.io/region: #Set it to the ID of the current region.
 storage.kubernetes.io/csiProvisionerIdentity: everest-csi-provisioner
 volumeHandle: objbucket #Replace the name with the actual bucket name of the third-party tenant.
 persistentVolumeReclaimPolicy: Retain #This parameter must be set to Retain to ensure that the bucket will not be deleted when a PV is deleted.
 storageClassName: csi-obs-mountoption #You can associate a new custom OBS storage class or the built-in csi-obs of the cluster.

```

- **mountOptions:** This field contains the new OBS mounting parameters that allow the bucket owner to have full access to the data in the bucket. This field solves the problem that the bucket owner cannot read the data written into a mounted third-party bucket. If the object storage of a third-party tenant is mounted, **default\_acl** must be set to **bucket-owner-full-control**.
- **persistentVolumeReclaimPolicy:** When the object storage of a third-party tenant is mounted, this field must be set to **Retain**. In this way, the OBS bucket will not be deleted when a PV is deleted. The service platform of the SaaS service provider needs to manage the lifecycle of the third-party bucket PVs. When a PVC is deleted separately, the PV is not deleted. Instead, it will be retained. To do so, call the native Kubernetes APIs to create and delete static PVs.
- **storageClassName:** You can associate a new custom OBS storage class ([click here](#)) or the built-in `csi-obs` of the cluster.

**PVC of a bound OBS bucket:**

```

apiVersion: v1
kind: PersistentVolumeClaim
metadata:
 annotations:
 csi.storage.k8s.io/fstype: obsfs
 everest.io/obs-volume-type: STANDARD
 volume.beta.kubernetes.io/storage-provisioner: everest-csi-provisioner
 name: objbucketpvc #Replace the name with the actual PVC name of the bucket.
 namespace: default
spec:
 accessModes:
 - ReadWriteMany
 resources:
 requests:
 storage: 1Gi
 storageClassName: csi-obs-mountoption #The value must be the same as the storage class
 associated with the bound PV.
 volumeName: objbucket #Replace the name with the actual PV name of the bucket to be bound.

```

- **Static PV of an OBS parallel file system:**

```

apiVersion: v1
kind: PersistentVolume
metadata:
 name: obsfscheck #Replace the name with the actual PV name of the parallel file system.
 annotations:
 pv.kubernetes.io/provisioned-by: everest-csi-provisioner
spec:
 accessModes:
 - ReadWriteMany
 capacity:
 storage: 1Gi
 mountOptions:
 - default_acl=bucket-owner-full-control #New OBS mounting parameters
 csi:
 driver: obs.csi.everest.io
 fsType: obsfs
 volumeAttributes:
 everest.io/obs-volume-type: STANDARD
 everest.io/region:
 storage.kubernetes.io/csiProvisionerIdentity: everest-csi-provisioner
 volumeHandle: obsfscheck #Replace the name with the actual name of the parallel file
 system of the third-party tenant.
 persistentVolumeReclaimPolicy: Retain #This parameter must be set to Retain to ensure that
 the bucket will not be deleted when a PV is deleted.
 storageClassName: csi-obs-mountoption #You can associate a new custom OBS storage class
 or the built-in csi-obs of the cluster.

```

- **mountOptions:** This field contains the new OBS mounting parameters that allow the bucket owner to have full access to the data in the bucket. This field solves the problem that the bucket owner cannot read the data written into a mounted third-party bucket. If the object storage of a third-party tenant is mounted, **default\_acl** must be set to **bucket-owner-full-control**.
- **persistentVolumeReclaimPolicy:** When the object storage of a third-party tenant is mounted, this field must be set to **Retain**. In this way, the OBS bucket will not be deleted when a PV is deleted. The service platform of the SaaS service provider needs to manage the lifecycle of the third-party bucket PVs. When a PVC is deleted separately, the PV is not deleted. Instead, it will be retained. To do so, call the native Kubernetes APIs to create and delete static PVs.
- **storageClassName:** You can associate a new custom OBS storage class ([click here](#)) or the built-in `csi-obs` of the cluster.

PVC of a bound OBS parallel file system:

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
 annotations:
 csi.storage.k8s.io/fstype: obsfs
 everest.io/obs-volume-type: STANDARD
 volume.beta.kubernetes.io/storage-provisioner: everest-csi-provisioner
 name: obsfscheckpvc #Replace the name with the actual PVC name of the parallel file system.
 namespace: default
spec:
 accessModes:
 - ReadWriteMany
 resources:
 requests:
 storage: 1Gi
 storageClassName: csi-obs-mountoption #The value must be the same as the storage class
 associated with the bound PV.
 volumeName: obsfscheck #Replace the name with the actual PV name of the parallel file system.
```

- **(Optional) Creating a custom OBS storage class to associate with a static PV:**

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
 name: csi-obs-mountoption
mountOptions:
 - default_acl=bucket-owner-full-control
parameters:
 csi.storage.k8s.io/csi-driver-name: obs.csi.everest.io
 csi.storage.k8s.io/fstype: obsfs
 everest.io/obs-volume-type: STANDARD
provisioner: everest-csi-provisioner
reclaimPolicy: Retain
volumeBindingMode: Immediate
```

- **csi.storage.k8s.io/fstype:** File type. The value can be **obsfs** or **s3fs**. If the value is **s3fs**, an OBS bucket is created and mounted using `s3fs`. If the value is **obsfs**, an OBS parallel file system is created and mounted using `obsfs`.
- **reclaimPolicy:** Reclaim policy of a PV. The value will be set in **PV.spec.persistentVolumeReclaimPolicy** dynamically created based on the new PVC associated with the storage class. If the value is **Delete**, the external OBS bucket and the PV will be deleted when the PVC is deleted. If the value is **Retain**, the PV and external storage are retained when the PVC is deleted. In this case, clear the PV separately. In the scenario where

an imported third-party bucket is associated, the storage class is used only for associating static PVs (with this field set to **Retain**). Dynamic creation is not involved.

## 21.9.3 Dynamically Creating and Mounting Subdirectories of an SFS Turbo File System

### 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.
- 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.
- Nodes running Ubuntu OS do not support this function if they use the Docker container engine.

### Creating an SFS Turbo Volume of the subpath Type

---

**CAUTION**

Do not expand, disassociate, or delete a **subpath** volume.

---

**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.
    - More options are available in everest 1.2.8 or a later version. **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 the subpath mode, it is useless to specify this parameter. The storage capacity is limited by the total capacity of the SFS Turbo file system. If the total capacity of the SFS Turbo file system is insufficient, expand the capacity on the SFS Turbo page in a timely manner.

Step 5 Run `kubectl create -f sfs-turbo-test.yaml`.

----End

 **NOTE**

It is meaningless to conduct capacity expansion on an SFS Turbo volume created in the subpath mode. This operation does not expand the capacity of the SFS Turbo file system. Ensure that the total capacity of the SFS Turbo file system is not used up.

Creating a Deployment and Mounting an Existing Volume to the Deployment

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 Deployment.
- **image**: specifies the image used by the Deployment.
- **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 StatefulSet.
- **image**: specifies the image used by the StatefulSet.
- **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 are mapped to each other.
- **storageClassName**: indicates the name of the StorageClass.

Step 2 Create the StatefulSet.

```
kubectl create -f statefulset-test.yaml
```

```
----End
```

21.9.4 How Do I Change the Storage Class Used by a Cluster of v1.15 from FlexVolume to CSI Everest?

In clusters later than v1.15.11-r1, CSI (the everest add-on) has taken over all functions of fuxi FlexVolume (the storage-driver add-on) for managing container storage. You are advised to use CSI Everest.

To migrate your storage volumes, create a static PV to associate with the original underlying storage, and then create a PVC to associate with this static PV. When you upgrade your application, mount the new PVC to the original mounting path to migrate the storage volumes.

WARNING

Services will be interrupted during the migration. Therefore, properly plan the migration and back up data.

Procedure

Step 1 (Optional) Back up data to prevent data loss in case of exceptions.

Step 2 Configure a YAML file of the PV in the CSI format according to the PV in the FlexVolume format and associate the PV with the existing storage.

To be specific, run the following commands to configure the `pv-example.yaml` file, which is used to create a PV.

```
touch pv-example.yaml
```

```
vi pv-example.yaml
```

Configuration example of a PV for an EVS volume:

```
apiVersion: v1
kind: PersistentVolume
metadata:
  labels:
    failure-domain.beta.kubernetes.io/region: eu-west-0
    failure-domain.beta.kubernetes.io/zone: <zone name>
  annotations:
    pv.kubernetes.io/provisioned-by: everest-csi-provisioner
    name: pv-evs-example
spec:
  accessModes:
    - ReadWriteOnce
  capacity:
    storage: 10Gi
  csi:
    driver: disk.csi.everest.io
    fsType: ext4
    volumeAttributes:
      everest.io/disk-mode: SCSI
      everest.io/disk-volume-type: SAS
      storage.kubernetes.io/csiProvisionerIdentity: everest-csi-provisioner
      volumeHandle: 0992dbda-6340-470e-a74e-4f0db288ed82
    persistentVolumeReclaimPolicy: Delete
  storageClassName: csi-disk
```

Pay attention to the fields in bold and red. The parameters are described as follows:

Table 21-11 EVS volume configuration parameters

| Parameter | Description |
|---|---|
| <code>failure-domain.beta.kubernetes.io/region</code> | Region where the EVS disk is located. Use the same value as that of the FlexVolume PV. |
| <code>failure-domain.beta.kubernetes.io/zone</code> | AZ where the EVS disk is located. Use the same value as that of the FlexVolume PV. |
| <code>name</code> | Name of the PV, which must be unique in the cluster. |
| <code>storage</code> | EVS volume capacity in the unit of Gi. Use the value of <code>spec.capacity.storage</code> of the FlexVolume PV. |

| Parameter | Description |
|-----------------------------|--|
| driver | Storage driver used to attach the volume. Set the driver to disk.csi.everest.io for the EVS volume. |
| volumeHandle | Volume ID of the EVS disk. Use the value of spec.flexVolume.options.volumeID of the FlexVolume PV. |
| everest.io/disk-mode | EVS disk mode. Use the value of spec.flexVolume.options.disk-mode of the FlexVolume PV. |
| everest.io/disk-volume-type | EVS disk type. Use the value of kubernetes.io/volumetype in the storage class corresponding to spec.storageClassName of the FlexVolume PV. |
| storageClassName | Name of the Kubernetes storage class associated with the storage volume. Set this field to csi-disk for EVS disks. |

Configuration example of a PV for an SFS volume:

```
apiVersion: v1
kind: PersistentVolume
metadata:
  name: pv-sfs-example
  annotations:
    pv.kubernetes.io/provisioned-by: everest-csi-provisioner
spec:
  accessModes:
    - ReadWriteMany
  capacity:
    storage: 10Gi
  csi:
    driver: nas.csi.everest.io
    fsType: nfs
    volumeAttributes:
      everest.io/share-export-location: # Shared path of the file storage
      storage.kubernetes.io/csiProvisionerIdentity: everest-csi-provisioner
      volumeHandle: 682f00bb-ace0-41d8-9b3e-913c9aa6b695
  persistentVolumeReclaimPolicy: Delete
  storageClassName: csi-nas
```

Pay attention to the fields in bold and red. The parameters are described as follows:

Table 21-12 SFS volume configuration parameters

| Parameter | Description |
|-----------|--|
| name | Name of the PV, which must be unique in the cluster. |
| storage | File storage size in the unit of Gi. Use the value of spec.capacity.storage of the FlexVolume PV. |
| driver | Storage driver used to attach the volume. Set the driver to nas.csi.everest.io for the file system. |

| Parameter | Description |
|----------------------------------|---|
| everest.io/share-export-location | Shared path of the file system. Use the value of spec.flexVolume.options.deviceMountPath of the FlexVolume PV. |
| volumeHandle | File system ID. Use the value of spec.flexVolume.options.volumeID of the FlexVolume PV. |
| storageClassName | Name of the Kubernetes storage class. Set this field to csi-nas . |

Configuration example of a PV for an OBS volume:

```

apiVersion: v1
kind: PersistentVolume
metadata:
  name: pv-obs-example
  annotations:
    pv.kubernetes.io/provisioned-by: everest-csi-provisioner
spec:
  accessModes:
    - ReadWriteMany
  capacity:
    storage: 1Gi
  csi:
    driver: obs.csi.everest.io
    fsType: s3fs
    volumeAttributes:
      everest.io/obs-volume-type: STANDARD
      everest.io/region: eu-west-0
      storage.kubernetes.io/csiProvisionerIdentity: everest-csi-provisioner
    volumeHandle: obs-normal-static-pv
  persistentVolumeReclaimPolicy: Delete
  storageClassName: csi-obs
  
```

Pay attention to the fields in bold and red. The parameters are described as follows:

Table 21-13 OBS volume configuration parameters

| Parameter | Description |
|-----------|---|
| name | Name of the PV, which must be unique in the cluster. |
| storage | Storage capacity, in the unit of Gi. Set this parameter to the fixed value 1Gi . |
| driver | Storage driver used to attach the volume. Set the driver to obs.csi.everest.io for the OBS volume. |

| Parameter | Description |
|----------------------------|--|
| fsType | File type. Value options are obsfs or s3fs . If the value is s3fs , an OBS bucket is created and mounted using s3fs. If the value is obsfs , an OBS parallel file system is created and mounted using obsfs. Set this parameter according to the value of spec.flexVolume.options.posix of the FlexVolume PV. If the value of spec.flexVolume.options.posix is true , set this parameter to obsfs . If the value is false , set this parameter to s3fs . |
| everest.io/obs-volume-type | Storage class, including STANDARD (standard bucket) and WARM (infrequent access bucket). Set this parameter according to the value of spec.flexVolume.options.storage_class of the FlexVolume PV. If the value of spec.flexVolume.options.storage_class is standard , set this parameter to STANDARD . If the value is standard_ia , set this parameter to WARM . |
| everest.io/region | Region where the OBS bucket is located. Use the value of spec.flexVolume.options.region of the FlexVolume PV. |
| volumeHandle | OBS bucket name. Use the value of spec.flexVolume.options.volumeID of the FlexVolume PV. |
| storageClassName | Name of the Kubernetes storage class. Set this field to csi-obs . |

Configuration example of a PV for an SFS Turbo volume:

```

apiVersion: v1
kind: PersistentVolume
metadata:
  name: pv-efs-example
  annotations:
    pv.kubernetes.io/provisioned-by: everest-csi-provisioner
spec:
  accessModes:
    - ReadWriteMany
  capacity:
    storage: 10Gi
  csi:
    driver: sfsturbo.csi.everest.io
    fsType: nfs
    volumeAttributes:
      everest.io/share-export-location: 192.168.0.169/
      storage.kubernetes.io/csiProvisionerIdentity: everest-csi-provisioner
    volumeHandle: 8962a2a2-a583-4b7f-bb74-fe76712d8414
  persistentVolumeReclaimPolicy: Delete
  storageClassName: csi-sfsturbo
  
```

Pay attention to the fields in bold and red. The parameters are described as follows:

Table 21-14 SFS Turbo volume configuration parameters

| Parameter | Description |
|----------------------------------|--|
| name | Name of the PV, which must be unique in the cluster. |
| storage | File system size. Use the value of spec.capacity.storage of the FlexVolume PV. |
| driver | Storage driver used to attach the volume. Set it to sfsturbo.csi.everest.io . |
| everest.io/share-export-location | Shared path of the SFS Turbo volume. Use the value of spec.flexVolume.options.deviceMountPath of the FlexVolume PV. |
| volumeHandle | SFS Turbo volume ID. Use the value of spec.flexVolume.options.volumeID of the FlexVolume PV. |
| storageClassName | Name of the Kubernetes storage class. Set this field to csi-sfsturbo for SFS Turbo volumes. |

Step 3 Configure a YAML file of the PVC in the CSI format according to the PVC in the FlexVolume format and associate the PVC with the PV created in [Step 2](#).

To be specific, run the following commands to configure the `pvc-example.yaml` file, which is used to create a PVC.

touch pvc-example.yaml

vi pvc-example.yaml

Configuration example of a PVC for an EVS volume:

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  labels:
    failure-domain.beta.kubernetes.io/region: eu-west-0
    failure-domain.beta.kubernetes.io/zone: <zone name>
  annotations:
    everest.io/disk-volume-type: SAS
    volume.beta.kubernetes.io/storage-provisioner: everest-csi-provisioner
  name: pvc-evs-example
  namespace: default
spec:
  accessModes:
  - ReadWriteOnce
  resources:
    requests:
      storage: 10Gi
  volumeName: pv-evs-example
  storageClassName: csi-disk
```

Pay attention to the fields in bold and red. The parameters are described as follows:

Table 21-15 PVC configuration parameters for an EVS volume

| Parameter | Description |
|--|--|
| failure-domain.beta.kubernetes.io/region | Region where the cluster is located. Use the same value as that of the FlexVolume PVC. |
| failure-domain.beta.kubernetes.io/zone | AZ where the EVS disk is deployed. Use the same value as that of the FlexVolume PVC. |
| everest.io/disk-volume-type | Storage class of the EVS disk. The value can be SAS or SSD . Set this parameter to the same value as that of the PV created in Step 2 . |
| name | PVC name, which must be unique in the namespace. The value must be unique in the namespace. (If the PVC is dynamically created by a stateful application, the value of this parameter must be the same as the name of the FlexVolume PVC.) |
| namespace | Namespace to which the PVC belongs. Use the same value as that of the FlexVolume PVC. |
| storage | Requested capacity of the PVC, which must be the same as the storage size of the existing PV. |
| volumeName | Name of the PV. Set this parameter to the name of the static PV in Step 2 . |
| storageClassName | Name of the Kubernetes storage class. Set this field to csi-disk for EVS disks. |

Configuration example of a PVC for an SFS volume:

```

apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  annotations:
    volume.beta.kubernetes.io/storage-provisioner: everest-csi-provisioner
  name: pvc-sfs-example
  namespace: default
spec:
  accessModes:
    - ReadWriteMany
  resources:
    requests:
      storage: 10Gi
  storageClassName: csi-nas
  volumeName: pv-sfs-example

```

Pay attention to the fields in bold and red. The parameters are described as follows:

Table 21-16 PVC configuration parameters for an SFS volume

| Parameter | Description |
|------------------|--|
| name | PVC name, which must be unique in the namespace. The value must be unique in the namespace. (If the PVC is dynamically created by a stateful application, the value of this parameter must be the same as the name of the FlexVolume PVC.) |
| namespace | Namespace to which the PVC belongs. Use the same value as that of the FlexVolume PVC. |
| storage | Storage capacity, in the unit of Gi. The value must be the same as the storage size of the existing PV. |
| storageClassName | Set this field to csi-nas . |
| volumeName | Name of the PV. Set this parameter to the name of the static PV in Step 2 . |

Configuration example of a PVC for an OBS volume:

```

apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  annotations:
    volume.beta.kubernetes.io/storage-provisioner: everest-csi-provisioner
    everest.io/obs-volume-type: STANDARD
    csi.storage.k8s.io/fstype: s3fs
    name: pvc-obs-example
    namespace: default
spec:
  accessModes:
    - ReadWriteMany
  resources:
    requests:
      storage: 1Gi
  storageClassName: csi-obs
  volumeName: pv-obs-example

```

Pay attention to the fields in bold and red. The parameters are described as follows:

Table 21-17 PVC configuration parameters for an OBS volume

| Parameter | Description |
|----------------------------|--|
| everest.io/obs-volume-type | OBS volume type, which can be STANDARD (standard bucket) and WARM (infrequent access bucket). Set this parameter to the same value as that of the PV created in Step 2 . |
| csi.storage.k8s.io/fstype | File type, which can be obsfs or s3fs . The value must be the same as that of fsType of the static OBS volume PV. |

| Parameter | Description |
|------------------|--|
| name | PVC name, which must be unique in the namespace. The value must be unique in the namespace. (If the PVC is dynamically created by a stateful application, the value of this parameter must be the same as the name of the FlexVolume PVC.) |
| namespace | Namespace to which the PVC belongs. Use the same value as that of the FlexVolume PVC. |
| storage | Storage capacity, in the unit of Gi. Set this parameter to the fixed value 1Gi . |
| storageClassName | Name of the Kubernetes storage class. Set this field to csi-obs . |
| volumeName | Name of the PV. Set this parameter to the name of the static PV created in Step 2 . |

Configuration example of a PVC for an SFS Turbo volume:

```

apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  annotations:
    volume.beta.kubernetes.io/storage-provisioner: everest-csi-provisioner
  name: pvc-efs-example
  namespace: default
spec:
  accessModes:
    - ReadWriteMany
  resources:
    requests:
      storage: 10Gi
  storageClassName: csi-sfsturbo
  volumeName: pv-efs-example

```

Pay attention to the fields in bold and red. The parameters are described as follows:

Table 21-18 PVC configuration parameters for an SFS Turbo volume

| Parameter | Description |
|------------------|--|
| name | PVC name, which must be unique in the namespace. The value must be unique in the namespace. (If the PVC is dynamically created by a stateful application, the value of this parameter must be the same as the name of the FlexVolume PVC.) |
| namespace | Namespace to which the PVC belongs. Use the same value as that of the FlexVolume PVC. |
| storageClassName | Name of the Kubernetes storage class. Set this field to csi-sfsturbo . |

| Parameter | Description |
|------------|---|
| storage | Storage capacity, in the unit of Gi. The value must be the same as the storage size of the existing PV. |
| volumeName | Name of the PV. Set this parameter to the name of the static PV created in Step 2 . |

Step 4 Upgrade the workload to use a new PVC.

For Deployments

1. Run the **kubectl create -f** commands to create a PV and PVC.

```
kubectl create -f pv-example.yaml
kubectl create -f pvc-example.yaml
```

NOTE

Replace the example file name **pvc-example.yaml** in the preceding commands with the names of the YAML files configured in [Step 2](#) and [Step 3](#).

2. Go to the CCE console. On the workload upgrade page, click **Upgrade > Advanced Settings > Data Storage > Cloud Storage**.
3. Uninstall the old storage and add the PVC in the CSI format. Retain the original mounting path in the container.
4. Click **Submit**.
5. Wait until the pods are running.

For StatefulSets that use existing storage

1. Run the **kubectl create -f** commands to create a PV and PVC.

```
kubectl create -f pv-example.yaml
kubectl create -f pvc-example.yaml
```

NOTE

Replace the example file name **pvc-example.yaml** in the preceding commands with the names of the YAML files configured in [Step 2](#) and [Step 3](#).

2. Run the **kubectl edit** command to edit the StatefulSet and use the newly created PVC.

```
kubectl edit sts sts-example -n xxx
```

```
30 pod.alpha.kubernetes.io/initialized: true
31 spec:
32   volumes:
33     - name: cce-efs-import-kjxmtzqn-z65j
34       persistentVolumeClaim:
35         claimName: pvc-csi-sfsturbo-f2ed93a7-408c-49c3-9a8b-9ded5c6e1533-1
36   containers:
```

NOTE

Replace **sts-example** in the preceding command with the actual name of the StatefulSet to upgrade. **xxx** indicates the namespace to which the StatefulSet belongs.

3. Wait until the pods are running.

 NOTE

The current console does not support the operation of adding new cloud storage for StatefulSets. Use the `kubectl` commands to replace the storage with the newly created PVC.

For StatefulSets that use dynamically allocated storage

1. Back up the PV and PVC in the flexVolume format used by the StatefulSet.
`kubectl get pvc xxx -n {namespaces} -oyaml > pvc-backup.yaml`
`kubectl get pv xxx -n {namespaces} -oyaml > pv-backup.yaml`
2. Change the number of pods to **0**.
3. On the storage page, disassociate the flexVolume PVC used by the StatefulSet.
4. Run the `kubectl create -f` commands to create a PV and PVC.
`kubectl create -f pv-example.yaml`
`kubectl create -f pvc-example.yaml`

 NOTE

Replace the example file name `pvc-example.yaml` in the preceding commands with the names of the YAML files configured in [Step 2](#) and [Step 3](#).

5. Change the number of pods back to the original value and wait until the pods are running.

 NOTE

The dynamic allocation of storage for StatefulSets is achieved by using **volumeClaimTemplates**. This field cannot be modified by Kubernetes. Therefore, data cannot be migrated by using a new PVC.

The PVC naming rule of the **volumeClaimTemplates** is fixed. When a PVC that meets the naming rule exists, this PVC is used.

Therefore, disassociate the original PVC first, and then create a PVC with the same name in the CSI format.

6. (Optional) Recreate the stateful application to ensure that a CSI PVC is used when the application is scaled out. Otherwise, FlexVolume PVCs are used in scaling out.

- Run the following command to obtain the YAML file of the StatefulSet:

`kubectl get sts xxx -n {namespaces} -oyaml > sts.yaml`

- Run the following command to back up the YAML file of the StatefulSet:

`cp sts.yaml sts-backup.yaml`

- Modify the definition of **volumeClaimTemplates** in the YAML file of the StatefulSet.

vi sts.yaml

Configuration example of **volumeClaimTemplates** for an EVS volume:

```
volumeClaimTemplates:
- metadata:
  name: pvc-161070049798261342
  namespace: default
  creationTimestamp: null
  annotations:
```

```

    everest.io/disk-volume-type: SAS
  spec:
    accessModes:
      - ReadWriteOnce
    resources:
      requests:
        storage: 10Gi
    storageClassName: csi-disk

```

The parameter value must be the same as the PVC of the EVS volume created in [Step 3](#).

Configuration example of **volumeClaimTemplates** for an SFS volume:

```

volumeClaimTemplates:
  - metadata:
      name: pvc-161063441560279697
      namespace: default
      creationTimestamp: null
    spec:
      accessModes:
        - ReadWriteMany
      resources:
        requests:
          storage: 10Gi
      storageClassName: csi-nas

```

The parameter value must be the same as the PVC of the SFS volume created in [Step 3](#).

Configuration example of **volumeClaimTemplates** for an OBS volume:

```

volumeClaimTemplates:
  - metadata:
      name: pvc-161070100417416148
      namespace: default
      creationTimestamp: null
      annotations:
        csi.storage.k8s.io/fstype: s3fs
        everest.io/obs-volume-type: STANDARD
    spec:
      accessModes:
        - ReadWriteMany
      resources:
        requests:
          storage: 1Gi
      storageClassName: csi-obs

```

The parameter value must be the same as the PVC of the OBS volume created in [Step 3](#).

- Delete the StatefulSet.

kubectl delete sts xxx -n {namespaces}

- Create the StatefulSet.

kubectl create -f sts.yaml

Step 5 Check service functions.

1. Check whether the application is running properly.
2. Checking whether the data storage is normal.

 NOTE

If a rollback is required, perform [Step 4](#). Select the PVC in FlexVolume format and upgrade the application.

Step 6 Uninstall the PVC in the FlexVolume format.

If the application functions normally, unbind the PVC in the FlexVolume format on the storage management page.

You can also run the kubectl command to delete the PVC and PV of the FlexVolume format.

 CAUTION

Before deleting a PV, change the persistentVolumeReclaimPolicy of the PV to **Retain**. Otherwise, the underlying storage will be reclaimed after the PV is deleted.

If the cluster has been upgraded before the storage migration, PVs may fail to be deleted. You can remove the PV protection field **finalizers** to delete PVs.

```
kubectl patch pv {pv_name} -p '{"metadata":{"finalizers":null}}'
```

----End

21.9.5 Custom Storage Classes

Background

When using storage resources in CCE, the most common method is to specify **storageClassName** to define the type of storage resources to be created when creating a PVC. The following configuration shows how to use a PVC to apply for an SAS (high I/O) EVS disk (block storage).

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: pvc-evs-example
  namespace: default
  annotations:
    everest.io/disk-volume-type: SAS
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 10Gi
  storageClassName: csi-disk
```

To specify the EVS disk type, you can set 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 **SATA** (common I/O) and **SSD** (ultra-high I/O).

This configuration method may not work if you want to:

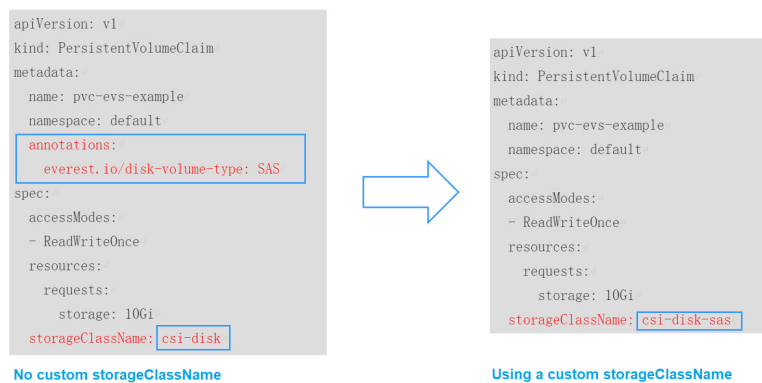
- Set **storageClassName** only, which is simpler than specifying the EVS disk type by using **everest.io/disk-volume-type**.

- 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

```

Default Storage Classes in CCE

Run the following command to query the supported storage classes.

```

# kubectl get sc
NAME          PROVISIONER          AGE

```

| | | | |
|-------------------------------|-------------------------|-----|--|
| csi-disk | everest-csi-provisioner | 17d | # Storage class for EVS disks |
| csi-disk-topology-association | everest-csi-provisioner | 17d | # Storage class for EVS disks with delayed association |
| csi-nas | everest-csi-provisioner | 17d | # Storage class for SFS file systems |
| csi-obs | everest-csi-provisioner | 17d | # Storage Class for OBS buckets |
| csi-sfsturbo | everest-csi-provisioner | 17d | # Storage class for SFS Turbo file systems |

Check the details of **csi-disk**. You can see that the type of the disk created by **csi-disk** is SAS by default.

```
# kubectl get sc csi-disk -oyaml
allowVolumeExpansion: true
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  creationTimestamp: "2021-03-17T02:10:32Z"
  name: csi-disk
  resourceVersion: "760"
  selfLink: /apis/storage.k8s.io/v1/storageclasses/csi-disk
  uid: 4db97b6c-853b-443d-b0dc-41cdbc8140f2
parameters:
  csi.storage.k8s.io/csi-driver-name: disk.csi.everest.io
  csi.storage.k8s.io/fstype: ext4
  everest.io/disk-volume-type: SAS
  everest.io/passthrough: "true"
provisioner: everest-csi-provisioner
reclaimPolicy: Delete
volumeBindingMode: Immediate
```

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.

 **NOTE**

The reclamation policy set here has no impact on the SFS Turbo storage.

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 StorageClass

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.

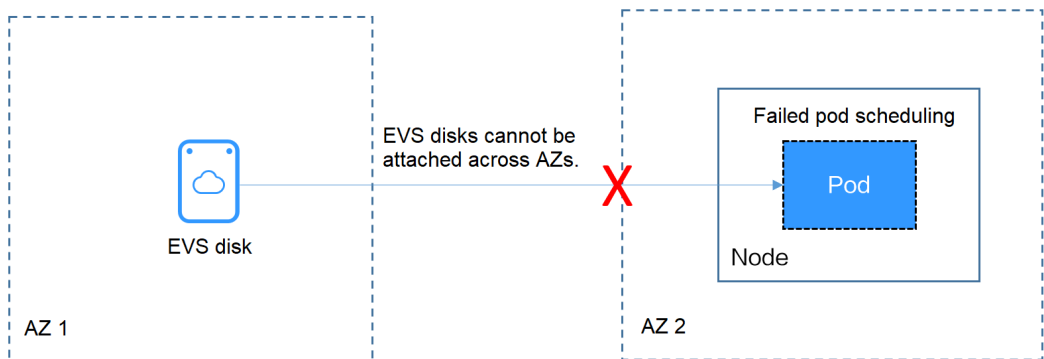
```
# kubectl create -f ssd-disk.yaml
persistentvolumeclaim/ssd-disk created
# kubectl get pvc
NAME      STATUS  VOLUME                                     CAPACITY  ACCESS MODES  STORAGECLASS  AGE
sas-disk  Bound   pvc-6e2f37f9-7346-4419-82f7-b42e79f7964c  10Gi      RWO           csi-disk-sas  16m
ssd-disk  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.

21.9.6 Enabling Automatic Topology for EVS Disks When Nodes Are Deployed in Different AZs (csi-disk-topology)

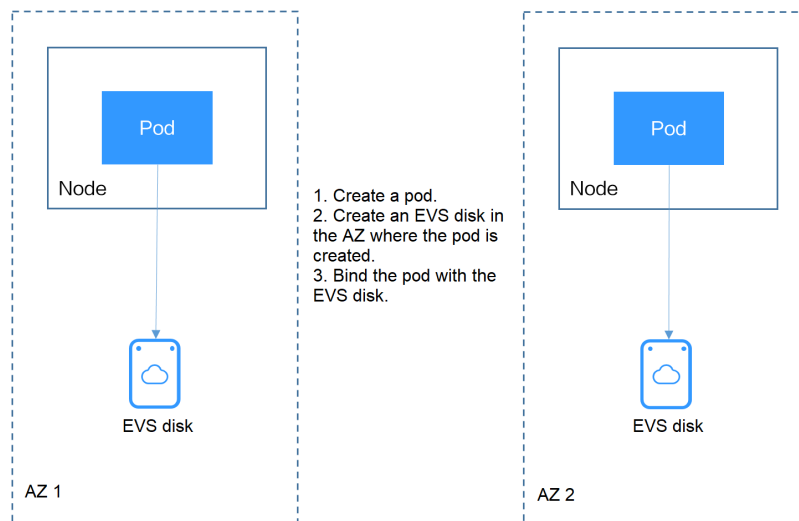
Background

EVS disks cannot be attached to a node deployed in another AZ. For example, the EVS disks in AZ 1 cannot be attached to a node in AZ 2. If the storage class `csi-disk` is used for StatefulSets, when a StatefulSet is scheduled, a PVC and a PV are created immediately (an EVS disk is created along with the PV), and then the PVC is bound to the PV. However, when the cluster nodes are located in multiple AZs, the EVS disk created by the PVC and the node to which the pods are scheduled may be in different AZs. As a result, the pods fail to be scheduled.



Solution

CCE provides a storage class named `csi-disk-topology`, which is a late-binding EVS disk type. When you use this storage class to create a PVC, no PV will be created in pace with the PVC. Instead, the PV is created in the AZ of the node where the pod will be scheduled. An EVS disk is then created in the same AZ to ensure that the EVS disk can be attached and the pod can be successfully scheduled.



Failed Pod Scheduling Due to csi-disk Used in Cross-AZ Node Deployment

Create a cluster with three nodes in different AZs.

Use the csi-disk storage class to create a StatefulSet and check whether the workload is successfully created.

```

apiVersion: apps/v1
kind: StatefulSet
metadata:
  name: nginx
spec:
  serviceName: nginx # Name of the headless Service
  replicas: 4
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
        - name: container-0
          image: nginx:alpine
          resources:
            limits:
              cpu: 600m
              memory: 200Mi
            requests:
              cpu: 600m
              memory: 200Mi
          volumeMounts: # Storage mounted to the pod
            - name: data
              mountPath: /usr/share/nginx/html # Mount the storage to /usr/share/nginx/html.
      imagePullSecrets:
        - name: default-secret
      volumeClaimTemplates:
        - metadata:
            name: data
            annotations:
              everest.io/disk-volume-type: SAS
          spec:
            accessModes:
              - ReadWriteOnce
            resources:

```

```
requests:
  storage: 1Gi
  storageClassName: csi-disk
```

The StatefulSet uses the following headless Service.

```
apiVersion: v1
kind: Service      # Object type (Service)
metadata:
  name: nginx
  labels:
    app: nginx
spec:
  ports:
    - name: nginx  # Name of the port for communication between pods
      port: 80     # Port number for communication between pods
  selector:
    app: nginx    # Select the pod whose label is app:nginx.
  clusterIP: None # Set this parameter to None, indicating the headless Service.
```

After the creation, check the PVC and pod status. In the following output, the PVC has been created and bound successfully, and a pod is in the Pending state.

```
# kubectl get pvc -owide
NAME          STATUS  VOLUME                                     CAPACITY  ACCESS MODES  STORAGECLASS
AGE  VOLUMEMODE
data-nginx-0  Bound  pvc-04e25985-fc93-4254-92a1-1085ce19d31e  1Gi      RWO           csi-disk
64s  Filesystem
data-nginx-1  Bound  pvc-0ae6336b-a2ea-4ddc-8f63-cfc5f9efe189  1Gi      RWO           csi-disk
47s  Filesystem
data-nginx-2  Bound  pvc-aa46f452-cc5b-4dbd-825a-da68c858720d  1Gi      RWO           csi-disk
30s  Filesystem
data-nginx-3  Bound pvc-3d60e532-ff31-42df-9e78-015cacb18a0b  1Gi      RWO           csi-disk
14s  Filesystem

# kubectl get pod -owide
NAME          READY  STATUS   RESTARTS  AGE  IP           NODE          NOMINATED NODE  READINESS GATES
nginx-0       1/1    Running  0          2m25s  172.16.0.12  192.168.0.121 <none>          <none>
nginx-1       1/1    Running  0          2m8s   172.16.0.136 192.168.0.211 <none>          <none>
nginx-2       1/1    Running  0          111s   172.16.1.7   192.168.0.240 <none>          <none>
nginx-3       0/1    Pending 0          95s    <none>       <none>        <none>          <none>
```

The event information of the pod shows that the scheduling fails due to no available node. Two nodes (in AZ 1 and AZ 2) do not have sufficient CPUs, and the created EVS disk is not in the AZ where the third node (in AZ 3) is located. As a result, the pod cannot use the EVS disk.

```
# kubectl describe pod nginx-3
Name:          nginx-3
...
Events:
  Type     Reason          Age    From          Message
  ----     -
Warning   FailedScheduling 111s   default-scheduler 0/3 nodes are available: 3 pod has unbound immediate PersistentVolumeClaims.
Warning   FailedScheduling 111s   default-scheduler 0/3 nodes are available: 3 pod has unbound immediate PersistentVolumeClaims.
Warning   FailedScheduling 28s    default-scheduler 0/3 nodes are available: 1 node(s) had volume node affinity conflict, 2 Insufficient cpu.
```

Check the AZ where the EVS disk created from the PVC is located. It is found that data-nginx-3 is in AZ 1. In this case, the node in AZ 1 has no resources, and only the node in AZ 3 has CPU resources. As a result, the scheduling fails. Therefore, there should be a delay between creating the PVC and binding the PV.

Storage Class for Delayed Binding

If you check the cluster storage class, you can see that the binding mode of `csi-disk-topology` is **WaitForFirstConsumer**, indicating that a PV is created and bound when a pod uses the PVC. That is, the PV and the underlying storage resources are created based on the pod information.

```
# kubectl get storageclass
```

| NAME | PROVISIONER | RECLAIMPOLICY | VOLUMEBINDINGMODE | AGE | 156m |
|-------------------|-------------------------|---------------|-----------------------------|-------|------|
| csi-disk | everest-csi-provisioner | Delete | Immediate | true | 156m |
| csi-disk-topology | everest-csi-provisioner | Delete | WaitForFirstConsumer | true | 156m |
| csi-nas | everest-csi-provisioner | Delete | Immediate | true | 156m |
| csi-obs | everest-csi-provisioner | Delete | Immediate | false | 156m |

VOLUMEBINDINGMODE is displayed if your cluster is v1.19. It is not displayed in clusters of v1.17 or v1.15.

You can also view the binding mode in the `csi-disk-topology` details.

```
# kubectl describe sc csi-disk-topology
Name:          csi-disk-topology
IsDefaultClass:  No
Annotations:    <none>
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
AllowVolumeExpansion: True
MountOptions:  <none>
ReclaimPolicy: Delete
VolumeBindingMode: WaitForFirstConsumer
Events:        <none>
```

Create PVCs of the `csi-disk` and `csi-disk-topology` classes. Observe the differences between these two types of PVCs.

- csi-disk**

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: disk
  annotations:
    everest.io/disk-volume-type: SAS
spec:
  accessModes:
  - ReadWriteOnce
  resources:
    requests:
      storage: 10Gi
  storageClassName: csi-disk # StorageClass
```

- csi-disk-topology**

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: topology
  annotations:
    everest.io/disk-volume-type: SAS
spec:
  accessModes:
  - ReadWriteOnce
  resources:
    requests:
      storage: 10Gi
  storageClassName: csi-disk-topology # StorageClass
```

View the PVC details. As shown below, the `csi-disk` PVC is in `Bound` state and the `csi-disk-topology` PVC is in `Pending` state.

```
# kubectl create -f pvc1.yaml
persistentvolumeclaim/disk created
# kubectl create -f pvc2.yaml
persistentvolumeclaim/topology created
# kubectl get pvc
NAME          STATUS  VOLUME                                     CAPACITY  ACCESS MODES  STORAGECLASS  AGE
disk          Bound   pvc-88d96508-d246-422e-91f0-8caf414001fc  10Gi      RWO           csi-disk      18s
topology      Pending                                     csi-disk-topology  2s
```

View details about the `csi-disk-topology` PVC. You can see that "waiting for first consumer to be created before binding" is displayed in the event, indicating that the PVC is bound after the consumer (pod) is created.

```
# kubectl describe pvc topology
Name:          topology
Namespace:     default
StorageClass:  csi-disk-topology
Status:        Pending
Volume:
Labels:        <none>
Annotations:   everest.io/disk-volume-type: SAS
Finalizers:    [kubernetes.io/pvc-protection]
Capacity:
Access Modes:
VolumeMode:   Filesystem
Used By:       <none>
Events:
  Type     Reason              Age           From              Message
  ----     -
  Normal   WaitForFirstConsumer  5s (x3 over 30s)  persistentvolume-controller  waiting for first consumer to be created before binding
```

Create a workload that uses the PVC. Set the PVC name to **topology**.

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
spec:
  selector:
    matchLabels:
      app: nginx
  replicas: 1
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
        - image: nginx:alpine
          name: container-0
          volumeMounts:
            - mountPath: /tmp # Mount path
              name: topology-example
          restartPolicy: Always
      volumes:
        - name: topology-example
          persistentVolumeClaim:
            claimName: topology # PVC name
```

After the PVC is created, check the PVC details. You can see that the PVC is bound successfully.

```
# kubectl describe pvc topology
Name:          topology
```



```

Namespace: default
StorageClass: csi-disk-topology
Status: Bound
....
Used By: nginx-deployment-fcd9fd98b-x6tbs
Events:
  Type Reason          Age
  From                                     Message
  ---- -
  Normal WaitForFirstConsumer 84s (x26 over 7m34s) persistentvolume-
  controller waiting for first consumer to be created before
  binding
  Normal Provisioning 54s everest-csi-provisioner_everest-csi-
  controller-7965dc48c4-5k799_2a6b513e-f01f-4e77-af21-6d7f8d4dbc98 External provisioner is provisioning
  volume for claim "default/topology"
  Normal ProvisioningSucceeded 52s everest-csi-provisioner_everest-csi-
  controller-7965dc48c4-5k799_2a6b513e-f01f-4e77-af21-6d7f8d4dbc98 Successfully provisioned volume
  pvc-9a89ea12-4708-4c71-8ec5-97981da032c9
  
```

Using csi-disk-topology in Cross-AZ Node Deployment

The following uses csi-disk-topology to create a StatefulSet with the same configurations used in the preceding example.

```

volumeClaimTemplates:
- metadata:
  name: data
  annotations:
    everest.io/disk-volume-type: SAS
  spec:
    accessModes:
    - ReadWriteOnce
    resources:
      requests:
        storage: 1Gi
    storageClassName: csi-disk-topology
  
```

After the creation, check the PVC and pod status. As shown in the following output, the PVC and pod can be created successfully. The nginx-3 pod is created on the node in AZ 3.

```

# kubectl get pvc -owide
NAME          STATUS  VOLUME                                     CAPACITY  ACCESS MODES  STORAGECLASS  AGE  VOLUMEMODE
data-nginx-0 Bound    pvc-43802cec-cf78-4876-bcca-e041618f2470 1Gi       RWO           csi-disk-    55s  Filesystem
topology
data-nginx-1 Bound    pvc-fc942a73-45d3-476b-95d4-1eb94bf19f1f 1Gi       RWO           csi-disk-    39s  Filesystem
topology
data-nginx-2 Bound    pvc-d219f4b7-e7cb-4832-a3ae-01ad689e364e 1Gi       RWO           csi-disk-    22s  Filesystem
topology
data-nginx-3 Bound    pvc-b54a61e1-1c0f-42b1-9951-410ebd326a4d 1Gi       RWO           csi-disk-    9s   Filesystem
topology

# kubectl get pod -owide
NAME    READY  STATUS   RESTARTS  AGE  IP           NODE           NOMINATED NODE  READINESS
GATES
nginx-0 1/1    Running  0          65s  172.16.1.8  192.168.0.240 <none>          <none>
nginx-1 1/1    Running  0          49s  172.16.0.13 192.168.0.121 <none>          <none>
nginx-2 1/1    Running  0          32s  172.16.0.137 192.168.0.211 <none>          <none>
nginx-3 1/1    Running  0          19s  172.16.1.9  192.168.0.240 <none>          <none>
  
```

21.10 Container

21.10.1 Properly Allocating Container Computing Resources

If a node has sufficient memory resources, a container on this node can use more memory resources than requested, but no more than limited. If the memory allocated to a container exceeds the upper limit, the container is stopped first. If the container continuously uses memory resources more than limited, the container is terminated. If a stopped container is allowed to be restarted, kubelet will restart it, but other types of run errors will occur.

Scenario 1

The node's memory has reached the memory limit reserved for the node. As a result, OOM killer is triggered.

Solution

You can either scale up the node or migrate the pods on the node to other nodes.

Scenario 2

The upper limit of resources configured for the pod is too small. When the actual usage exceeds the limit, OOM killer is triggered.

Solution

Set a higher upper limit for the workload.

Example

A pod will be created and allocated memory that exceeds the limit. As shown in the following configuration file of the pod, the pod requests 50 MiB memory and the memory limit is set to 100 MiB.

Example YAML file (memory-request-limit-2.yaml):

```
apiVersion: v1
kind: Pod
metadata:
  name: memory-demo-2
spec:
  containers:
  - name: memory-demo-2-ctr
    image: vish/stress
    resources:
      requests:
        memory: 50Mi
      limits:
        memory: "100Mi"
    args:
      - -mem-total
      - 250Mi
      - -mem-alloc-size
      - 10Mi
      - -mem-alloc-sleep
      - 1s
```

The **args** parameters indicate that the container attempts to request 250 MiB memory, which exceeds the pod's upper limit (100 MiB).

Creating a pod:

```
kubectl create -f https://k8s.io/docs/tasks/configure-pod-container/memory-request-limit-2.yaml --namespace=mem-example
```

Viewing the details about the pod:

```
kubectl get pod memory-demo-2 --namespace=mem-example
```

In this stage, the container may be running or be killed. If the container is not killed, repeat the previous command until the container is killed.

| NAME | READY | STATUS | RESTARTS | AGE |
|---------------|-------|-----------|----------|-----|
| memory-demo-2 | 0/1 | OOMKilled | 1 | 24s |

Viewing detailed information about the container:

```
kubectl get pod memory-demo-2 --output=yaml --namespace=mem-example
```

This output indicates that the container is killed because the memory limit is exceeded.

```
lastState:
  terminated:
    containerID: docker://7aae52677a4542917c23b10fb56fcb2434c2e8427bc956065183c1879cc0dbd2
    exitCode: 137
    finishedAt: 2020-02-20T17:35:12Z
    reason: OOMKilled
    startedAt: null
```

In this example, the container can be automatically restarted. Therefore, kubelet will start it again. You can run the following command several times to see how the container is killed and started:

```
kubectl get pod memory-demo-2 --namespace=mem-example
```

The preceding command output indicates how the container is killed and started back and forth:

```
$ kubectl get pod memory-demo-2 --namespace=mem-example
NAME          READY   STATUS    RESTARTS   AGE
memory-demo-2 0/1     OOMKilled 1           37s
$ kubectl get pod memory-demo-2 --namespace=mem-example
NAME          READY   STATUS    RESTARTS   AGE
memory-demo-2 1/1     Running   2           40s
```

Viewing the historical information of the pod:

```
kubectl describe pod memory-demo-2 --namespace=mem-example
```

The following command output indicates that the pod is repeatedly killed and started.

```
... Normal Created Created container with id
66a3a20aa7980e61be4922780bf9d24d1a1d8b7395c09861225b0eba1b1f8511
... Warning BackOff Back-off restarting failed container
```

21.10.2 Modifying Kernel Parameters Using a Privileged Container

Prerequisites

To access a Kubernetes cluster from a client, you can use the Kubernetes command line tool `kubectl`.

Procedure

Step 1 Create a DaemonSet in the background, select the Nginx image, enable the Privileged Container, configure the lifecycle, and add the **hostNetwork** field (value: **true**).

1. Create a **daemonSet** file.

vi daemonSet.yaml

An example YAML file is provided as follows:

NOTICE

The **spec.spec.containers.lifecycle** field indicates the command that will be run after the container is started.

```
kind: DaemonSet
apiVersion: apps/v1
metadata:
  name: daemonset-test
  labels:
    name: daemonset-test
spec:
  selector:
    matchLabels:
      name: daemonset-test
  template:
    metadata:
      labels:
        name: daemonset-test
    spec:
      hostNetwork: true
      containers:
      - name: daemonset-test
        image: nginx:alpine-perl
        command:
        - "/bin/sh"
        args:
        - "-c"
        - while ;; do time=$(date);done
        imagePullPolicy: IfNotPresent
        lifecycle:
          postStart:
            exec:
              command:
              - sysctl
              - "-w"
              - net.ipv4.tcp_tw_reuse=1
        securityContext:
          privileged: true
        imagePullSecrets:
        - name: default-secret
```

2. Create a DaemonSet.

kubectl create -f daemonSet.yaml

Step 2 Check whether the DaemonSet is successfully created.

kubectl get daemonset *DaemonSet name*

In this example, run the following command:

kubectl get daemonset daemonset-test

Information similar to the following is displayed:

| NAME | DESIRED | CURRENT | READY | UP-T0-DATE | AVAILABLE | NODE SELECTOR | AGE |
|----------------|---------|---------|-------|------------|-----------|---------------|-----|
| daemonset-test | 2 | 2 | 2 | 2 | <node> | 2h | |

Step 3 Query the container ID of DaemonSet on the node.

```
docker ps -a|grep DaemonSet name
```

In this example, run the following command:

```
docker ps -a|grep daemonset-test
```

Information similar to the following is displayed:

| | | | | |
|---------------------|---|-----------------------|----------------|-------|
| 897b99faa9ce | 3e094d5696c1 | "/bin/sh -c while..." | 31 minutes ago | Up 30 |
| minutes | ault_fa7cc313-4ac1-11e9-a716-fa163e0aalba_0 | | | |

Step 4 Access the container.

```
docker exec -it containerid /bin/sh
```

In this example, run the following command:

```
docker exec -it 897b99faa9ce /bin/sh
```

Step 5 Check whether the configured command is executed after the container is started.

```
sysctl -a |grep net.ipv4.tcp_tw_reuse
```

If the following information is displayed, the system parameters are modified successfully:

```
net.ipv4.tcp_tw_reuse=1
```

----End

21.10.3 Using Init Containers to Initialize an Application

Concepts

Before containers running applications are started, one or some init containers are started first. If there are multiple init containers, they will be started in the defined sequence. The application containers are started only after all init containers run to completion and exit. Storage volumes in a pod are shared. Therefore, the data generated in the init containers can be used by the application containers.

Init containers can be used in multiple Kubernetes resources, such as Deployments, DaemonSets, and jobs. They perform initialization before application containers are started.

Application Scenarios

Before deploying a service, you can use an init container to make preparations before the pod where the service is running is deployed. After the preparations are complete, the init container runs to completion and exit, and the container to be deployed will be started.

- **Scenario 1: Wait for other modules to be ready.** For example, an application contains two containerized services: web server and database. The web server service needs to access the database service. However, when the application is

started, the database service may have not been started. Therefore, web server may fail to access database. To solve this problem, you can use an init container in the pod where web server is running to check whether database is ready. The init container runs to completion only when database is accessible. Then, web server is started and initiates a formal access request to database.

- **Scenario 2: Initialize the configuration.** For example, the init container can check all existing member nodes in the cluster and prepare the cluster configuration information for the application container. After the application container is started, it can be added to the cluster using the configuration information.
- **Other scenarios:** For example, register a pod with a central database and download application dependencies.

For details, see [Init Containers](#).

Procedure

Step 1 Edit the YAML file of the init container workload.

vi deployment.yaml

An example YAML file is provided as follows:

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: mysql
spec:
  replicas: 1
  selector:
    matchLabels:
      name: mysql
  template:
    metadata:
      labels:
        name: mysql
    spec:
      initContainers:
      - name: getresource
        image: busybox
        command: ['sleep 20']
      containers:
      - name: mysql
        image: percona:5.7.22
        imagePullPolicy: Always
        ports:
        - containerPort: 3306
        resources:
          limits:
            memory: "500Mi"
            cpu: "500m"
          requests:
            memory: "500Mi"
            cpu: "250m"
        env:
        - name: MYSQL_ROOT_PASSWORD
          value: "mysql"
```

Step 2 Create an init container workload.

kubectl create -f deployment.yaml

Information similar to the following is displayed:

```
deployment.apps/mysql created
```

Step 3 Query the created Docker container on the node where the workload is running.

```
docker ps -a|grep mysql
```

The init container will exit after it runs to completion. The query result **Exited (0)** shows the exit status of the init container.

```

8dc822969e3f      percona          "docker-entrypoint..." 34 seconds ago      Up 33 seconds
ql_mysql-76598b8c64-mm9w9_default_522566ea-bda5-11e9-a219-fa163e8b288b_0
a745881214e7      busybox         "sh -c 'sleep 20'"      About a minute ago  Exited (0) 50 seconds ago
resource_mysql-76598b8c64-mm9w9_default_522566ea-bda5-11e9-a219-fa163e8b288b_0
615db9e60a80      cfe-pause:11.23.1  "/pause"                About a minute ago  Up About a minute
mysql-76598b8c64-mm9w9_default_522566ea-bda5-11e9-a219-fa163e8b288b_0

```

----End

21.10.4 Using hostAliases to Configure /etc/hosts in a Pod

Application Scenarios

If DNS or other related settings are inappropriate, you can use **hostAliases** to overwrite the resolution of the hostname at the pod level when adding entries to the **/etc/hosts** file of the pod.

Procedure

Step 1 Use kubectl to connect to the cluster.

Step 2 Create the **hostaliases-pod.yaml** file.

```
vi hostaliases-pod.yaml
```

The field in bold in the YAML file indicates the image name and tag. You can replace the example value as required.

```

apiVersion: v1
kind: Pod
metadata:
  name: hostaliases-pod
spec:
  hostAliases:
  - ip: 127.0.0.1
    hostnames:
    - foo.local
    - bar.local
  - ip: 10.1.2.3
    hostnames:
    - foo.remote
    - bar.remote
  containers:
  - name: cat-hosts
    image: tomcat:9-jre11-slim
    lifecycle:
      postStart:
        exec:
          command:
            - cat
            - /etc/hosts
    imagePullSecrets:
    - name: default-secret

```

Table 21-19 pod field description

| Parameter | Mandatory | Description |
|------------|-----------|---|
| apiVersion | Yes | API version number |
| kind | Yes | Type of the object to be created |
| metadata | Yes | Metadata definition of a resource object |
| name | Yes | Name of a pod |
| spec | Yes | Detailed description of the pod. For details, see Table 21-20 . |

Table 21-20 spec field description

| Parameter | Mandatory | Description |
|-------------|-----------|--|
| hostAliases | Yes | Host alias |
| containers | Yes | For details, see Table 21-21 . |

Table 21-21 containers field description

| Parameter | Mandatory | Description |
|-----------|-----------|----------------------|
| name | Yes | Container name |
| image | Yes | Container image name |
| lifecycle | No | Lifecycle |

Step 3 Create a pod.

kubectl create -f hostaliases-pod.yaml

If information similar to the following is displayed, the pod is created.

```
pod/hostaliases-pod created
```

Step 4 Query the pod status.

kubectl get pod hostaliases-pod

If the pod is in the **Running** state, the pod is successfully created.

```
NAME          READY   STATUS    RESTARTS   AGE
hostaliases-pod  1/1     Running   0           16m
```

Step 5 Check whether the **hostAliases** functions properly.

docker ps |grep hostaliases-pod

docker exec -ti *Container ID* /bin/sh


```
root@hostaliases-pod:/# cat /etc/hosts
# Kubernetes-managed hosts file.
127.0.0.1    localhost
::1        localhost ip6-localhost ip6-loopback
fe00::0    ip6-localnet
fe00::0    ip6-mcastprefix
fe00::1    ip6-allnodes
fe00::2    ip6-allrouters
10.0.0.25   hostaliases-pod

# Entries added by HostAliases.
127.0.0.1    foo.local    bar.local
10.1.2.3     foo.remote   bar.remote
```

----End

21.10.5 Configuring Core Dumps

Application Scenarios

Linux allows you to create a core dump file if an application crashes, which contains the data the application had in memory at the time of the crash. You can analyze the file to locate the fault.

Generally, when a service application crashes, its container exits and is reclaimed and destroyed. Therefore, container core files need to be permanently stored on the host or cloud storage. This topic describes how to configure container core dumps.

Constraints

When a container core dump is persistently stored to OBS (parallel file system or object bucket), the default mount option **umask=0** is used. As a result, although the core dump file is generated, the core dump information cannot be written to the core file.

Enabling Core Dump on a Node

Log in to the node, run the following command to enable core dump, and set the path and format for storing core files:

```
echo "/tmp/cores/core.%h.%e.%p.%t" > /proc/sys/kernel/core_pattern
```

%h, **%e**, **%p**, and **%t** are placeholders, which are described as follows:

- **%h**: hostname (or pod name). You are advised to configure this parameter.
- **%e**: program file name. You are advised to configure this parameter.
- **%p**: (optional) process ID.
- **%t**: (optional) time of the core dump.

After the core dump function is enabled by running the preceding command, the generated core file is named in the format of **core.{Host name}.{Program file name}.{Process ID}.{Time}**.

You can also configure a pre-installation or post-installation script to automatically run this command when creating a node.

Permanently Storing Core Dumps

A core file can be stored in your host (using a hostPath volume) or cloud storage (using a PVC). The following is an example YAML file for using a hostPath volume.

```
apiVersion: v1
kind: Pod
metadata:
  name: coredump
spec:
  volumes:
  - name: coredump-path
    hostPath:
      path: /home/coredump
  containers:
  - name: ubuntu
    image: ubuntu:12.04
    command: ["/bin/sleep","3600"]
    volumeMounts:
    - mountPath: /tmp/cores
      name: coredump-path
```

Create a pod using kubectl.

```
kubectl create -f pod.yaml
```

Verification

After the pod is created, access the container and trigger a segmentation fault of the current shell terminal.

```
$ kubectl get pod
NAME                READY STATUS RESTARTS AGE
coredump            1/1   Running 0       56s
$ kubectl exec -it coredump -- /bin/bash
root@coredump:/# kill -s SIGSEGV $$
command terminated with exit code 139
```

Log in to the node and check whether a core file is generated in the **/home/coredump** directory. The following example indicates that a core file is generated.

```
# ls /home/coredump
core.coredump.bash.18.1650438992
```

21.11 Permission

21.11.1 Configuring kubeconfig for Fine-Grained Management on Cluster Resources

Application Scenarios

By default, the kubeconfig file provided by CCE for users has permissions bound to the **cluster-admin** role, which are equivalent to the permissions of user **root**. It is difficult to implement refined management on users with such permissions.

Purpose

Cluster resources are managed in a refined manner so that specific users have only certain permissions (such as adding, querying, and modifying resources).

Precautions

Ensure that kubectl is available on your host. If not, download it from [here](#) (corresponding to the cluster version or the latest version).

Configuration Method

NOTE

In the following example, only pods and Deployments in the **test** space can be viewed and added, and they cannot be deleted.

Step 1 Set the service account name to **my-sa** and namespace to **test**.

```
kubectl create sa my-sa -n test
```

```
root@test-arm-54016 ~]#  
root@test-arm-54016 ~]# kubectl create sa my-sa -n test  
serviceaccount/my-sa created  
root@test-arm-54016 ~]#
```

Step 2 Configure the role table and assign operation permissions to different resources.

```
vi role-test.yaml
```

The content is as follows:

```
apiVersion: rbac.authorization.k8s.io/v1  
kind: Role  
metadata:  
  annotations:  
    rbac.authorization.kubernetes.io/autoupdate: "true"  
  labels:  
    kubernetes.io/bootstrapping: rbac-defaults  
  name: myrole  
  namespace: test  
rules:  
- apiGroups:  
  - ""  
  resources:  
  - pods  
  verbs:  
  - get  
  - list  
  - watch  
- apiGroups:  
  - apps  
  resources:
```

```
- pods
- deployments
verbs:
- get
- list
- watch
- create
```

Create a Role.

```
kubectl create -f role-test.yaml
```

```
[root@test-arm-54016 ~]# kubectl create -f role-test.yaml
role.rbac.authorization.k8s.io/myrole created
[root@test-arm-54016 ~]#
```

Step 3 Create a RoleBinding and bind the service account to the role so that the user can obtain the corresponding permissions.

```
vi myrolebinding.yaml
```

The content is as follows:

```
apiVersion: rbac.authorization.k8s.io/v1
kind: RoleBinding
metadata:
  name: myrolebinding
  namespace: test
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: Role
  name: myrole
subjects:
- kind: ServiceAccount
  name: my-sa
  namespace: test
```

Create a RoleBinding.

```
kubectl create -f myrolebinding.yaml
```

```
[root@test-arm-54016 ~]# kubectl create -f myrolebinding.yaml
rolebinding.rbac.authorization.k8s.io/myrolebinding created
[root@test-arm-54016 ~]#
```

The user information is configured. Now perform [Step 4](#) to [Step 6](#) to write the user information to the configuration file.

Step 4 Configure the cluster information.

1. Use the sa name **my-sa** to obtain the secret corresponding to the sa. In the following example, **my-sa-token-z4967** in the first column is the secret name.

```
kubectl get secret -n test |grep my-sa
```

```
[root@test-arm-54016 ~]# kubectl get secret -n test |grep my-sa
my-sa-token-5gpl4    kubernetes.io/service-account-token    3    21m
[root@test-arm-54016 ~]#
```

2. Decrypt the **ca.crt** file in the secret and export it.

```
kubectl get secret my-sa-token-5gpl4 -n test -oyaml |grep ca.crt: | awk '{print $2}' |base64 -d > /home/ca.crt
```

3. Set the cluster access mode. **test-arm** indicates the cluster to be accessed, **10.0.1.100** indicates the IP address of the API server in the cluster and **/home/test.config** indicates the path for storing the configuration file.

- If the internal API server address is used, run the following command:
kubectl config set-cluster **test-arm** --server=https://10.0.1.100:5443 --certificate-authority=/home/ca.crt --embed-certs=true --kubeconfig=/home/test.config
- If the public API server address is used, run the following command:
kubectl config set-cluster **test-arm** --server=https://10.0.1.100:5443 --kubeconfig=/home/test.config --insecure-skip-tls-verify=true

```
[root@test-arm-54016 home]# kubectl config set-cluster test-arm --server=https://10.0.1.100:5443 --certificate-authority=/home/ca.crt --embed-certs=true --kubeconfig=/home/test.config
Cluster "test-arm" set.
[root@test-arm-54016 home]# _
```

NOTE

If you **perform operations on a node in the cluster** or **the node that uses the configuration is a cluster node**, do not set the path of kubeconfig to `/root/.kube/config`.

The cluster API server address is an intranet API server address. After an EIP is bound to the cluster, the cluster API server address can also be a public API server address.

Step 5 Configure the cluster authentication information.

1. Obtain the cluster token. (If the token is obtained in GET mode, run **base64 -d** to decode the token.)

```
token=$(kubectl describe secret my-sa-token-5gpl4 -n test | awk '/token:/{print $2}')
```

2. Set the cluster user **ui-admin**.

```
kubectl config set-credentials ui-admin --token=$token --kubeconfig=/home/test.config
```

```
[root@test-arm-54016 home]# kubectl config set-credentials ui-admin --token=$token --kubeconfig=/home/test.config
User "ui-admin" set.
[root@test-arm-54016 home]#
```

Step 6 Configure the context information for cluster authentication. **ui-admin@test** is the context name.

```
kubectl config set-context ui-admin@test --cluster=test-arm --user=ui-admin --kubeconfig=/home/test.config
```

```
[root@test-arm-54016 home]# kubectl config set-context ui-admin@test --cluster=test-arm --user=ui-admin --kubeconfig=/home/test.config
Context "ui-admin@test" created.
[root@test-arm-54016 home]#
```

Step 7 Set the context. For details about how to use the context, see [Verification](#).

```
kubectl config use-context ui-admin@test --kubeconfig=/home/test.config
```

```
[paas@test-arm-54016 home]$ kubectl config use-context ui-admin@test --kubeconfig=/home/test.config
Switched to context "ui-admin@test".
[paas@test-arm-54016 home]$
```

NOTE

If you want to assign other users the above permissions to perform operations on the cluster, provide the generated configuration file `/home/test.config` to the user after performing step [Step 6](#). The user must ensure that the host can access the API server address of the cluster. When performing step [Step 7](#) on the host and using kubectl, the user must set the kubeconfig parameter to the path of the configuration file.

----End

Verification

1. Pods in the **test** namespace cannot access pods in other namespaces.
kubectl get pod -n **test** --kubeconfig=/home/test.config

```
lpaas@test-arm-54816 home1$ kubectl get pod -n test --kubeconfig=/home/test.config
NAME                                READY   STATUS    RESTARTS   AGE
test-pod-56cfcbf45b-12q92           0/1    CrashLoopBackOff   27         91m
lpaas@test-arm-54816 home1$
lpaas@test-arm-54816 home1$ kubectl get pod --kubeconfig=/home/test.config
Error from server (Forbidden): pods is forbidden: User "system:serviceaccount:test:mj-sa" cannot list resource "pods" in API group "" in the namespace "default"
lpaas@test-arm-54816 home1$
```

2. Pods in the **test** namespace cannot be deleted.

```
lpaas@test-arm-54816 home1$ kubectl delete pod -n test test-pod-56cfcbf45b-12q92 --kubeconfig=/home/test.config
Error from server (Forbidden): pods "test-pod-56cfcbf45b-12q92" is forbidden: User "system:serviceaccount:test:mj-sa" cannot delete resource "pods" in API group "" in the namespace "test"
lpaas@test-arm-54816 home1$
```

Further Readings

For more information about users and identity authentication in Kubernetes, see [Authenticating](#).

21.12 Release

21.12.1 Overview

Background

When switching between old and new services, you may be challenged in ensuring the system service continuity. If a new service version is directly released to all users at a time, it can be risky because once an online accident or bug occurs, the impact on users is great. It could take a long time to fix the issue. Sometimes, the version has to be rolled back, which severely affects user experience.

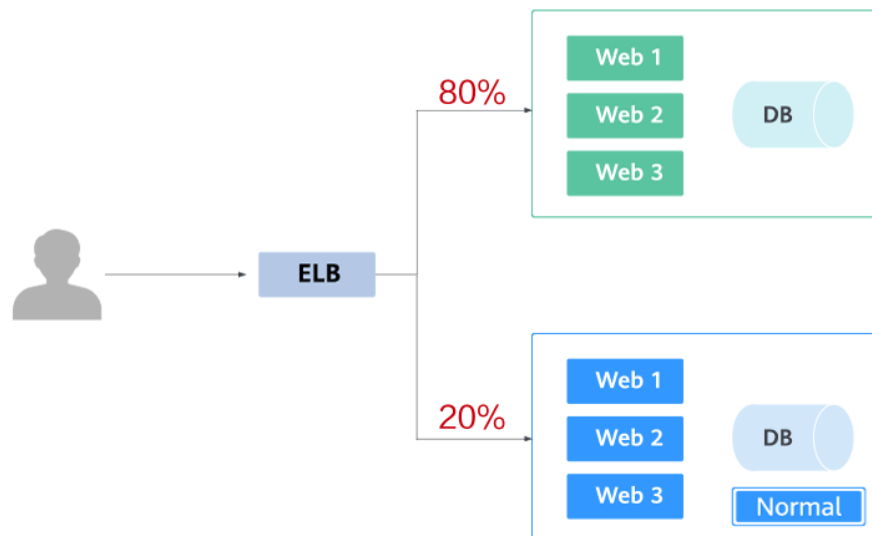
Solution

Several release policies are developed for service upgrade: grayscale release, blue-green deployment, A/B testing, rolling upgrade, and batch suspension of release. Traffic loss or service unavailability caused by releases can be avoided as much as possible.

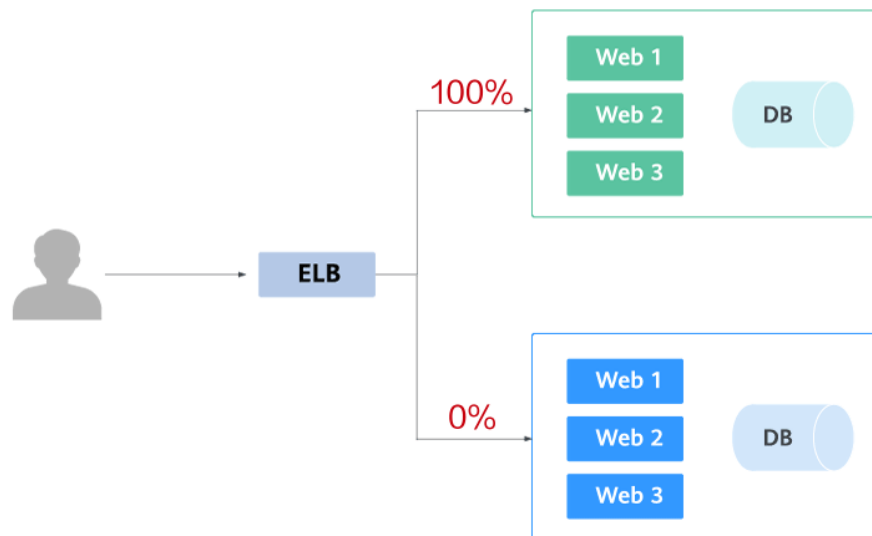
This document describes the principles and practices of grayscale release and blue-green deployment.

- Grayscale release, also called canary release, is a smooth iteration mode for version upgrade. During the upgrade, some users use the new version, while other users continue to use the old version. After the new version is stable and ready, it gradually takes over all the live traffic. In this way, service risks brought by the release of the new version can be minimized, the impact of faults can be reduced, and quick rollback is supported.

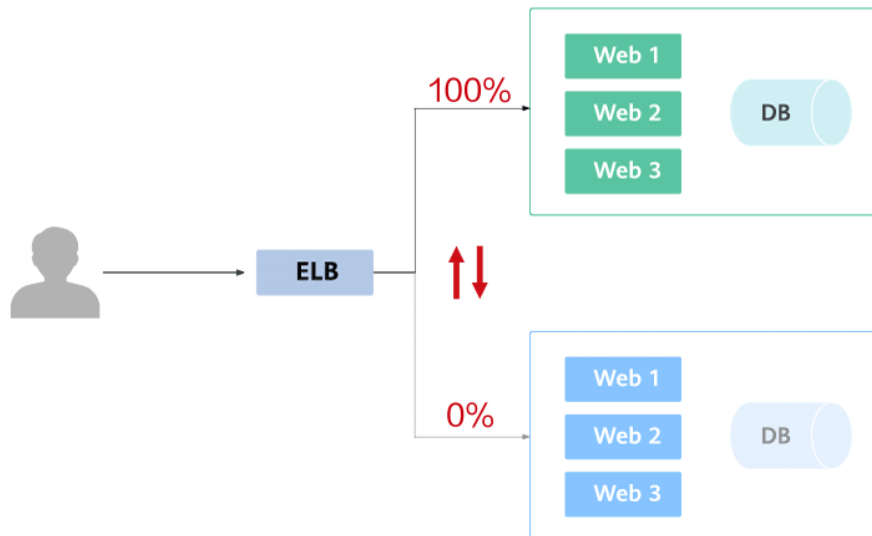
The following figure shows the general process of grayscale release. First, divide 20% of all service traffic to the new version. If the service version runs normally, gradually increase the traffic proportion and continue to test the performance of the new version. If the new version is stable, switch all traffic to it and bring the old version offline.



If an exception occurs in the new version when 20% of the traffic goes to the new version, you can quickly switch back to the old version.



- Blue-green deployment provides a zero-downtime, predictable manner for releasing applications to reduce service interruption during the release. A new version is deployed while the old version is retained. The two versions are online at the same time. The new and old versions work in hot backup mode. The route weight is switched (0 or 100) to enable different versions to go online or offline. If a problem occurs, the version can be quickly rolled back.



21.12.2 Using Services to Implement Simple Grayscale Release and Blue-Green Deployment

To implement grayscale release for a CCE cluster, deploy other open-source tools, such as Nginx Ingress, to the cluster or deploy services to a service mesh. These solutions are difficult to implement. If your grayscale release requirements are simple and you do not want to introduce too many plug-ins or complex configurations, you can refer to this section to implement simple grayscale release and blue-green deployment based on native Kubernetes features.

Principles

Users usually use Kubernetes objects such as Deployments and StatefulSets to deploy services. Each workload manages a group of pods. The following figure uses Deployment as an example.

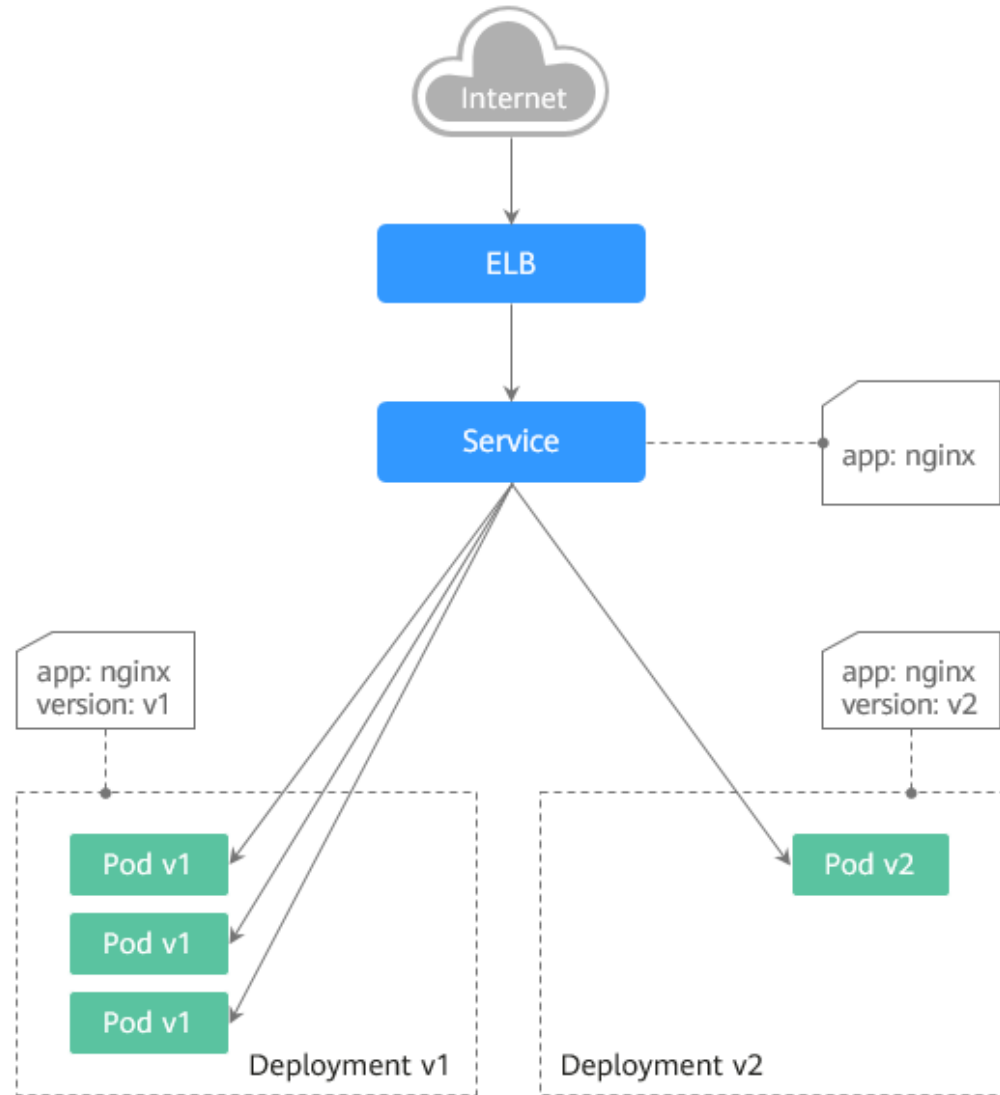


Generally, a Service is created for each workload. The Service uses the selector to match the backend pod. Other Services or objects outside the cluster can access the pods backing the Service. If a pod needs to be exposed, set the Service type to LoadBalancer. The ELB load balancer functions as the traffic entrance.

- **Grayscale release principles**

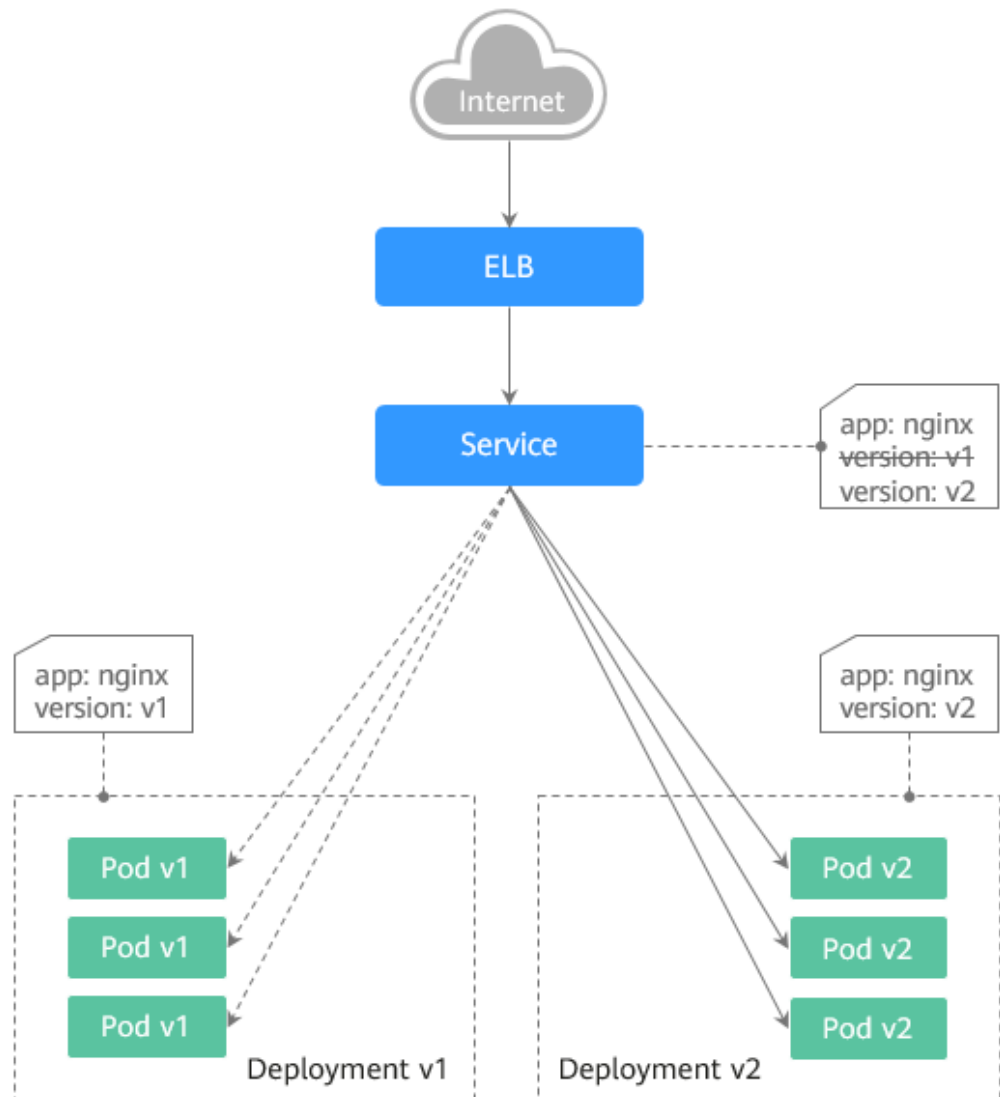
Take a Deployment as an example. A Service, in most cases, will be created for each Deployment. However, Kubernetes does not require that Services and

Deployments correspond to each other. A Service uses a selector to match backend pods. If pods of different Deployments are selected by the same selector, a Service corresponds to multiple versions of Deployments. You can adjust the number of replicas of Deployments of different versions to adjust the weights of services of different versions to achieve grayscale release. The following figure shows the process:



- **Blue-green deployment principles**

Take a Deployment as an example. Two Deployments of different versions have been deployed in the cluster, and their pods are labeled with the same key but different values to distinguish versions. A Service uses the selector to select the pod of a Deployment of a version. In this case, you can change the value of the label that determines the version in the Service selector to change the pod backing the Service. In this way, you can directly switch the service traffic from one version to another. The following figure shows the process:



Prerequisites

The Nginx image has been uploaded to SWR. The Nginx images have two versions: v1 and v2. The welcome pages are **Nginx-v1** and **Nginx-v2**.

Resource Creation

You can use YAML to deploy Deployments and Services in either of the following ways:

- On the **Create Deployment** page, click **Create YAML** on the right and edit the YAML file in the window.
- Save the sample YAML file in this section as a file and use `kubectl` to specify the YAML file. For example, run the `kubectl create -f xxx.yaml` command.

Step 1: Deploy Services of Two Versions

Two versions of Nginx services are deployed in the cluster to provide external access through ELB.

Step 1 Create a Deployment of the first version. The following uses nginx-v1 as an example. Example YAML:

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-v1
spec:
  replicas: 2           # Number of replicas of the Deployment, that is, the number of pods
  selector:           # Label selector
    matchLabels:
      app: nginx
      version: v1
  template:
    metadata:
      labels:         # Pod label
        app: nginx
        version: v1
    spec:
      containers:
        - image: {your_repository}/nginx:v1 # The image used by the container is nginx:v1.
          name: container-0
      resources:
        limits:
          cpu: 100m
          memory: 200Mi
        requests:
          cpu: 100m
          memory: 200Mi
      imagePullSecrets:
        - name: default-secret
```

Step 2 Create a Deployment of the second version. The following uses nginx-v2 as an example. Example YAML:

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-v2
spec:
  replicas: 2           # Number of replicas of the Deployment, that is, the number of pods
  selector:           # Label selector
    matchLabels:
      app: nginx
      version: v2
  template:
    metadata:
      labels:         # Pod label
        app: nginx
        version: v2
    spec:
      containers:
        - image: {your_repository}/nginx:v2 # The image used by the container is nginx:v2.
          name: container-0
      resources:
        limits:
          cpu: 100m
          memory: 200Mi
        requests:
          cpu: 100m
          memory: 200Mi
      imagePullSecrets:
        - name: default-secret
```

You can log in to the CCE console to view the deployment status.

----End

Step 2: Implement Grayscale Release

- Step 1** Create a LoadBalancer Service for the Deployment. Do not specify the version in the selector. Enable the Service to select the pods of the Deployments of two versions. Example YAML:

```
apiVersion: v1
kind: Service
metadata:
  annotations:
    kubernetes.io/elb.id: 586c97da-a47c-467c-a615-bd25a20de39c # ID of the ELB load balancer. Replace it
    with the actual value.
  name: nginx
spec:
  ports:
    - name: service0
      port: 80
      protocol: TCP
      targetPort: 80
  selector: # The selector does not contain version information.
    app: nginx
  type: LoadBalancer # Service type (LoadBalancer)
```

- Step 2** Run the following command to test the access:

```
for i in {1..10}; do curl <EXTERNAL_IP>; done;
```

<EXTERNAL_IP> indicates the IP address of the ELB load balancer.

The command output is as follows (Half of the responses are from the Deployment of version v1, and the other half are from version v2):

```
Nginx-v2
Nginx-v1
Nginx-v1
Nginx-v1
Nginx-v2
Nginx-v1
Nginx-v2
Nginx-v1
Nginx-v2
Nginx-v2
```

- Step 3** Use the console or kubectl to adjust the number of replicas of the Deployments. Change the number of replicas to 4 for v1 and 1 for v2.

```
kubectl scale deployment/nginx-v1 --replicas=4
```

```
kubectl scale deployment/nginx-v2 --replicas=1
```

- Step 4** Run the following command to test the access again:

```
for i in {1..10}; do curl <EXTERNAL_IP>; done;
```

<EXTERNAL_IP> indicates the IP address of the ELB load balancer.

In the command output, among the 10 access requests, only two responses are from the v2 version. The response ratio of the v1 and v2 versions is the same as the ratio of the number of replicas of the v1 and v2 versions, that is, 4:1. Grayscale release is implemented by controlling the number of replicas of services of different versions.

```
Nginx-v1
Nginx-v1
Nginx-v1
Nginx-v1
```

```

Nginx-v2
Nginx-v1
Nginx-v2
Nginx-v1
Nginx-v1
Nginx-v1

```

 **NOTE**

If the ratio of v1 to v2 is not 4:1, you can set the number of access times to a larger value, for example, 20. Theoretically, the more the times, the closer the response ratio between v1 and v2 is to 4:1.

----End

Step 3: Implement Blue-Green Deployment

Step 1 Create a LoadBalancer Service for a deployed Deployment and specify that the v1 version is used. Example YAML:

```

apiVersion: v1
kind: Service
metadata:
  annotations:
    kubernetes.io/elb.id: 586c97da-a47c-467c-a615-bd25a20de39c # ID of the ELB load balancer. Replace it
    with the actual value.
  name: nginx
spec:
  ports:
    - name: service0
      port: 80
      protocol: TCP
      targetPort: 80
  selector: # Set the version to v1 in the selector.
    app: nginx
    version: v1
  type: LoadBalancer # Service type (LoadBalancer)

```

Step 2 Run the following command to test the access:

```
for i in {1..10}; do curl <EXTERNAL_IP>; done;
```

<EXTERNAL_IP> indicates the IP address of the ELB load balancer.

The command output is as follows (all responses are from the v1 version):

```

Nginx-v1
Nginx-v1
Nginx-v1
Nginx-v1
Nginx-v1
Nginx-v1
Nginx-v1
Nginx-v1
Nginx-v1
Nginx-v1

```

Step 3 Use the console or kubectl to modify the selector of the Service so that the v2 version is selected.

```
kubectl patch service nginx -p '{"spec":{"selector":{"version":"v2"}}}'
```

Step 4 Run the following command to test the access again:

```
for i in {1..10}; do curl <EXTERNAL_IP>; done;
```

<EXTERNAL_IP> indicates the IP address of the ELB load balancer.

The returned results show that all responses are from the v2 version. The blue-green deployment is successfully implemented.

```
Nginx-v2  
Nginx-v2  
Nginx-v2  
Nginx-v2  
Nginx-v2  
Nginx-v2  
Nginx-v2  
Nginx-v2  
Nginx-v2  
Nginx-v2
```

----End

22 Migrating Data from CCE 1.0 to CCE 2.0

Perform the following operations to migrate data from CCE 1.0 to CCE 2.0:

1. **Migrating Images:** Pull images from the image repository of CCE 1.0, and push them to the image repository of CCE 2.0.
2. **Migrating clusters:** Create VM clusters in CCE 2.0 for application deployment.
3. **Migrating applications:** Move applications deployed in CCE 1.0 to the new clusters of CCE 2.0.

22.1 Differences Between CCE 1.0 and CCE 2.0

CCE 2.0 inherits and modifies the features of CCE 1.0, and release new features.

- Modified features:
 - Clusters in CCE 1.0 are equivalent to VM clusters in CCE 2.0.
 - CCE 2.0 does not have the concept of component template. In CCE 1.0, you can set parameters in the component template when creating an application.
 - Applications in CCE 1.0 are equivalent to Deployments in CCE 2.0. In addition, the concept of StatefulSet is added in CCE 2.0.
- New Features:
 - EVS disks and file storage can be created and directly mounted during workload creation on the CCE console.
 - Auto scaling for cluster nodes.
 - Docker upgraded to the 1706 version.
 - Customized helm charts for orchestrating workloads.
- In CCE 2.0, before uploading an image, you must create an image repository.

The following table compares the two versions.

Table 22-1 Comparison between CCE 1.0 and CCE 2.0

| CCE 1.0 | CCE 2.0 |
|--------------------|----------------------------------|
| Container registry | Image repository |
| Cluster | Resource Management > VM Cluster |
| Component template | - |
| App Designer | - |
| App Manager | Deployment |

22.2 Migrating Images

Migrate the images stored in the image repository of CCE 1.0 to CCE 2.0.

This section takes the Apache image as an example to show how to migrate images step by step.

Procedure

Before pushing container images, ensure that the local Docker client can access your private container registry.

Step 1 Log in to the Docker client as the **root** user.

Step 2 Create an organization. Organizations are used to isolate images and assign access permissions (read, edit, and manage) to different users.

1. Log in to the CCE console. In the navigation pane, choose **Image Repository**.
2. On the Image Repository dashboard page, click **Create Organization**.
3. Enter an organization name, that is, the domain name. When the configuration is complete, click **OK**.

Step 3 Log in to the Docker client as the **root** user.

Step 4 Pull the images stored in the image repository of CCE 1.0 to a local directory.

 **NOTE**


To pull images, log in to the CCE 1.0 console. In the navigation pane, choose **Image Repository**. Click the name of the target images. On the image details pages, you can find corresponding image pull command.

Example:

docker pull *10.154.57.150:443/test/apache-php:latest*

Step 5 Connect to CCE 2.0.

1. Log in to the CCE 2.0 console. In the navigation pane, click **Image Repository**.
2. On the page displayed, click **Upload Through Docker Client**. Click **Generate a temporary Docker login command**.

3. Click  to copy the command.

Step 6 Log in to the server where Docker is installed. Run the Docker login command copied in [Step 5](#).

After you log in to the Docker client, the system displays the message "login succeeded".

Step 7 Run the following command to tag the nginx:1.10 image:

```
docker tag {imageid} {image address}:tag
```

{image address} is the image repository address. If you push an image from an external network, enter the external image address. If you push an image from an internal network, enter the internal image address.

Example:

```
docker tag 2e233ad9329b 10.125.1.15:20202/test/apache-php:latest
```

Step 8 Run the following command to push the image to the image repository:

```
docker push {image address}:tag
```

Example:

```
docker push 10.125.1.15:20202/test/apache-php:latest
```

If the following information is displayed, the push is successful:

```
6d6b9812c8ae: Pushed
695da0025de6: Pushed
fe4c16cbf7a4: Pushed
1.10: digest: sha256:eb7e3bbd8e3040efa71d9c2cacfa12a8e39c6b2ccd15eac12bdc49e0b66cee63 size: 948
```

On the **SWR** console, choose **My Images**. The pushed images are displayed in the image list.

----End

22.3 Migrating Clusters

Create VM clusters on the CCE 2.0 console. These new VM clusters should have the same specifications with those created on CCE 1.0.

To create clusters using APIs, see *Cloud Container Engine API Reference 2.0*.

Procedure

Step 1 Log in to the CCE console. In the navigation pane, choose **Resource Management > Clusters**. Click **Create VM Cluster**.

Step 2 Set cluster parameters. Parameters with * are mandatory.

Table 22-2 Parameters for creating a cluster

| Parameter in CCE 2.0 | Parameter in CCE 1.0 | Configuration |
|----------------------|--|--|
| * Cluster Name | Name | Name of the cluster to be created. |
| *Version | This parameter does not exist in CCE 1.0. Retain the default value. | Cluster version, namely, corresponding Kubernetes baseline version. |
| *Management Scale | This parameter does not exist in CCE 1.0. Set this parameter based on your requirements. | Maximum number of nodes that can be managed by the cluster. |
| * High Availability | Cluster type | <ul style="list-style-type: none"> ● Yes: The cluster has three master nodes. The cluster is still available even when two master nodes are down. ● No: The cluster has only one master node. If the master node is down, the whole cluster becomes unavailable, but existing applications are not affected. |
| * VPC | VPCs created in CCE 1.0 can be used in CCE 2.0. | VPC where the cluster will be located.
If no VPCs are available, click Create a VPC . |
| * Subnet | Subnets created in CCE 1.0 can be used in CCE 2.0. | Subnet in which the cluster will run. |

| Parameter in CCE 2.0 | Parameter in CCE 1.0 | Configuration |
|----------------------|--|--|
| *Network Model | This parameter does not exist in CCE 1.0. Set this parameter based on your requirements. | <ul style="list-style-type: none">• Container tunnel network: indicates a network built on top of a VPC network and can be applied to common scenarios.• VPC network: delivers better performance and applies to high-performance and intensive interaction scenarios. Only one cluster using the VPC network model can be created in a single VPC. |

| Parameter in CCE 2.0 | Parameter in CCE 1.0 | Configuration |
|---------------------------|--|---|
| Container Network Segment | This parameter does not exist in CCE 1.0. Set this parameter based on your requirements. | <p>An IP address range that can be allocated to container pods.</p> <ul style="list-style-type: none"> • If Automatically select 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 10.0.0.0/12-19, 172.16.0.0/16-19, and 192.168.0.0/16-19. If different clusters share a container CIDR block, an IP address conflict will occur and access to the applications in the clusters may fail. • If Automatically select is selected, the system automatically assigns a CIDR block that does not conflict with any subnet CIDR block. <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 nodes supported by the current CIDR block will be displayed.</p> |

| Parameter in CCE 2.0 | Parameter in CCE 1.0 | Configuration |
|-------------------------|--|---|
| Service Network Segment | This parameter does not exist in CCE 1.0. Set this parameter based on your requirements. | <p>This parameter is left unspecified, by default. This parameter applies only to clusters of v1.11.7 and later versions.</p> <p>This parameter indicates a CIDR block of Kubernetes services. The mask of the service CIDR block must be appropriate. It determines the number of available nodes in a cluster.</p> |
| Open EIP | This parameter does not exist in CCE 1.0. Set this parameter based on your requirements. | <p>An independent public IP address that is reachable from public networks. Select an EIP that has not been bound to any node. A cluster's EIP is preset in the cluster's certificate. Do not delete the EIP after the cluster has been created. Otherwise, two-way authentication will fail.</p> <ul style="list-style-type: none"> • Do not configure: The cluster's master node will not have an EIP. • Configure now: If no EIP is available for selection, create one. |
| Authorization Mode | This parameter does not exist in CCE 1.0. Set this parameter based on your requirements. | <p>By default, RBAC is selected. Read CCE Role Management Instructions and select I am aware of the above limitations and read the CCE Role Management Instructions.</p> <p>After RBAC is enabled, users access resources in the cluster according to fine-grained permissions policies.</p> |

| Parameter in CCE 2.0 | Parameter in CCE 1.0 | Configuration |
|----------------------|--|---|
| Authentication Mode | This parameter does not exist in CCE 1.0. Set this parameter based on your requirements. | <p>Permission control on resources in a cluster. For example, you can configure user A to read and write application data in a namespace, while user B to only read resource data in a cluster.</p> <ul style="list-style-type: none"> • By default, X.509 authentication instead of Enhanced authentication is enabled. X.509 is a standard defining the format of public key certificates. X.509 certificates are used in many Internet protocols. • If permission control on a cluster is required, select Enhanced authentication and then Authenticating Proxy. Click Upload next to CA Root Certificate to upload a valid certificate. Select the check box to confirm that the uploaded certificate is valid. If the certificate is invalid, the cluster cannot be created. The uploaded certificate file must be smaller than 1 MB and in .crt or .cer format. |
| Cluster Description | Description | Description of the cluster. |

Step 3 After the configuration is complete, click **Next** to add a node.

Step 4 Continue to add a node.

Step 5 Set the parameters based on [Table 22-3](#).

Table 22-3 Parameters for adding a node

| Parameter in CCE 2.0 | Parameter in CCE 1.0 | Configuration |
|----------------------|----------------------|--|
| Scope | | |
| Current Region | Scope | Physical location of the node. |
| AZ. | | Physical region where resources use independent power supplies and networks. AZs are physically isolated but interconnected through an internal network. |
| Specifications | | |
| Node Name | Specifications | Name of the node. |

| Parameter in CCE 2.0 | Parameter in CCE 1.0 | Configuration |
|----------------------|----------------------|--|
| Node specifications. | | <ul style="list-style-type: none"> • General-purpose: provides general computing, storage, and network configurations for the majority of application scenarios. General-purpose instances can be used in web servers, development and test environments, and small database applications. • Memory-optimized: provides higher memory capacity than general-purpose nodes and is suitable for relational databases, NoSQL, and other workloads that are both memory-intensive and data-intensive. • General computing-plus: provides stable performance and exclusive resources to enterprise-class workloads with high and stable computing performance. • GPU-accelerated: provides powerful floating-point computing and is suitable for real-time, highly concurrent massive computing.
Calculation scenario. Graphical processing units (GPUs) of P series are suitable for deep learning, scientific computing, and CAE. GPUs of G series are suitable for 3D animation rendering and CAD.
Currently, only clusters of v1.11 support GPU-accelerated nodes. If the cluster version is v1.13 or later, GPU-accelerated is not displayed on the page. • Ultra-high I/O: provides ultra-low SSD access latency and ultra-high IOPS performance.
This type of specifications is suitable for high-performance relational databases, NoSQL databases (such as Cassandra and MongoDB), and Elasticsearch. |
| OS | | <p>Select an operating system for the node pool.</p> <p>Reinstalling OSs or modifying OS configurations could make nodes unavailable. Exercise caution when performing these operations. For details, see Risky Operations on Cluster Nodes.</p> |

| Parameter in CCE 2.0 | Parameter in CCE 1.0 | Configuration |
|--|--|---|
| Virtual Private Cloud | This parameter does not exist in CCE 1.0. Set this parameter based on your requirements. | The node pool inherits VPC settings from the cluster to which it belongs. This parameter is supported only in v1.13.10-r0 and later versions of clusters. It is not displayed in versions earlier than v1.13.10-r0. |
| Subnet | This parameter does not exist in CCE 1.0. Set this parameter based on your requirements. | A subnet provides dedicated network resources that are logically isolated from other networks for network security.

You can select any subnet in the cluster VPC. Cluster nodes can belong to different subnets. This parameter is supported only in v1.13.10-r0 and later versions of clusters. It is not displayed in versions earlier than v1.13.10-r0. |
| Nodes | Quantity | Number of nodes to be created. |
| <p>Network</p> <p>NOTE
If the nodes to be created require public network access, select Automatically assign or Use existing for EIP. If an EIP is not bound to a node, applications running on the node cannot be accessed by the external network.</p> | | |
| EIP | EIP | <p>A public IP address that is reachable from public networks.</p> <ul style="list-style-type: none"> • Do Not Use: A node without an EIP cannot access the Internet. It can be used only as a cloud server for deploying services or clusters on a private network. • Automatically assign: An EIP with exclusive bandwidth is automatically assigned to each ECS. When creating an ECS, ensure that the EIP quota is sufficient. Set the specifications, required quantity, billing mode, and bandwidth as required. • Use Existing EIP: An existing EIP is assigned to the node. |

| Parameter in CCE 2.0 | Parameter in CCE 1.0 | Configuration |
|-----------------------|--|---|
| Disks | Storage | <p>Disk type, which can be System Disk or Data Disk.</p> <ul style="list-style-type: none"> The system disk capacity ranges from 40 to 1024 GB. The default value is 40 GB. The data disk capacity ranges from 100 to 32678 GB. The default value is 100 GB. <p>Data disks deliver three levels of I/O performance:</p> <ul style="list-style-type: none"> Common I/O: SATA drives are used to store data. EVS disks of this level provide reliable block storage and a maximum IOPS of 1,000 per disk. They are suitable for key applications. High I/O: SAS drives are used to store data. EVS disks of this level provide a maximum IOPS of 3,000 and a minimum read/write latency of 1 ms. They are suitable for RDS, NoSQL, data warehouse, and file system applications. Ultra-high I/O: SSD drives are used to store data. EVS disks of this level provide a maximum IOPS of 20,000 and a minimum read/write latency of 1 ms. They are suitable for RDS, NoSQL, and data warehouse applications. |
| Login information | | |
| Key pair | Key pair | 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 and create one. |
| Advanced ECS Settings | | |
| ECS Group | This parameter does not exist in CCE 1.0. Set this parameter based on your requirements. | <p>Select an existing ECS group, or click Create ECS Group to create one. After the ECS group is created, click the refresh icon.</p> <p>An ECS group allows you to create ECSs on different hosts, thereby improving service reliability.</p> |

| Parameter in CCE 2.0 | Parameter in CCE 1.0 | Configuration |
|--|----------------------|--|
| Resource tag. | | <p>By adding tags to resources, you can classify resources.</p> <p>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.</p> <p>CCE will automatically create the "CCE-Dynamic-Provisioning-Node=<i>Node ID</i>" tag. A maximum of 5 tags can be added.</p> |
| Agency Management | | <p>The agency is created by the account 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 ECS or BMS to call cloud services, select Cloud service as the agency type, click Select, and then select ECS BMS.</p> |
| Script required before the installation. | | <p>Script commands. Enter 0 to 1000 characters.</p> <p>The script will be executed before Kubernetes software is installed. Note that if the script is incorrect, Kubernetes software may not be installed successfully. The script is usually used to format data disks.</p> |
| Script required after the installation. | | <p>Script commands. Enter 0 to 1000 characters.</p> <p>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.</p> |
| Add Data Disk | | <p>Click Add Data Disk to add a data disk and set the capacity of the data disk. Enter a disk formatting command in the input box of Pre-installation Script.</p> |
| Subnet IP Address | | <p>Select Automatically assign IP address (recommended) or Manually assigning IP addresses.</p> |
| Advanced Kubernetes Settings | | |

| Parameter in CCE 2.0 | Parameter in CCE 1.0 | Configuration |
|----------------------------------|--|--|
| Maximum Instances | This parameter does not exist in CCE 1.0. Set this parameter based on your requirements. | The maximum number of pods that can be created on a node, including the system's default pods. Value range: 16 to 250.
This limit prevents the node from being overloaded by managing too many pods. |
| insecure-registries | | Click Add insecure-registry and enter a repository address.
Add the address of the custom image repository with no valid SSL certificate to the docker startup option to avoid unsuccessful image pulling from the personal image repository. The address is in the format of IP address:Port number (or domain name). Post-installation script and insecure-registries cannot be used together. |
| Maximum Data Space per Container | This parameter does not exist in CCE 1.0. Set this parameter based on your requirements. | The maximum data space that can be used by a container. Value range: 10 GB to 80 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 supported only in v1.13.10-r0 and later versions of clusters. It is not displayed in versions earlier than v1.13.10-r0. |

Step 6 Click **Next** to install cluster add-ons.

System resource add-ons must be installed. Advanced functional add-ons are optional.

You can also install optional 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-on Management](#).

Step 7 Click **Create Now**. Check all the configurations, and click **Submit**.

It takes about 6 to 10 minutes to create a cluster. Information indicating the progress of the creation process will be displayed.

----End

22.4 Migrating Applications

This section describes how to create a Deployment with the same specifications as that in CCE 1.0 on the CCE 2.0 console.

It is advised to delete the applications on CCE 1.0 only after you have successfully created these applications on CCE 2.0.

22.4.1 Applications Created Through APIs or kubectl

If the application in CCE 1.0 is created using APIs or kubectl, use the same method to create applications of the same specifications on the cluster created in CCE 2.0. For details, see *Cloud Container Engine API Reference 2.0* or [Connecting to a Cluster Through kubectl](#).

- If you connect to the cluster through kubectl, you are advised to perform the operation in CCE 2.0. While in CCE 1.0, you need to download three certificate files and manually perform multiple steps on the client. In CCE 2.0, you only need to perform the following steps to connect to the cluster.
 - a. Download the kubectl CLI configuration file. (All authentication information and cluster addresses are in the same file.)
 - b. Install and configure the kubectl CLI tool. (Run the required Linux commands to complete the interconnection.)

For details, see [Connecting to a Cluster Through kubectl](#).
- If you access the CCE console through APIs, the cluster management APIs need to be updated. The native Kubernetes interface does not change, so you can use Kubernetes APIs in the same way. For details, see *Cloud Container Engine API Reference 2.0*.

22.4.2 Applications Created Through Component Templates

If the application in CCE 1.0 is created using a component template, follow the steps this section to migrate the application to CCE 2.0.

Migration Method

Create workloads in the CCE 2.0 console. Delete applications from in CCE 1.0 only after applications are successfully run in CCE 2.0.

Procedure

- Step 1** In the navigation pane, choose **Workloads > Deployments**. Click **Create Deployment**.
- Step 2** Set basic workload parameters as described in [Table 22-4](#). The parameters marked with an asterisk (*) are mandatory.

Table 22-4 Basic parameters

| Name | Configuration |
|-----------------|--|
| * Workload Name | Name of a workload, which must be unique. |
| * Cluster Name | Cluster to which the workload belongs. |
| * Namespace | Namespace to which the new workload belongs. By default, this parameter is set to default . |

| Name | Configuration |
|---------------------------|--|
| * Instances | <p>Number of instances in the workload. Each workload has at least one instance. You can specify the number of instances as required.</p> <p>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.</p> |
| Time Zone Synchronization | <p>If this parameter is enabled, the container and the node use the same time zone.</p> <p>NOTICE
After time zone synchronization is enabled, disks of the hostPath type will be automatically added and listed in the Data Storage > Local Volume area. Do not modify or delete the disks.</p> |
| Description | Description of the workload. |

Step 3 Click **Next** to add a container.

1. Click **Add Container** and select the image to be deployed.
2. Set image parameters according to [Table 22-5](#).

Table 22-5 Setting image parameters

| Parameter in CCE 2.0 | Parameter in CCE 1.0 | Configuration |
|----------------------|----------------------|--|
| Image Name | Container Images | Name of the image. You can click Change Image to select another image. |
| Image Version | | Version of the image to be deployed. |
| Container name. | | Container name, which is modifiable. |
| Privileged Container | | <p>Programs in a privileged container have certain privileges.</p> <p>If Privileged Container is enabled, the container is granted superuser permissions. For example, privileged containers can manipulate network devices on the host machine and modify kernel parameters.</p> |

| Parameter in CCE 2.0 | Parameter in CCE 1.0 | Configuration |
|--------------------------|----------------------|---|
| Container Specifications | Memory and CPU | <p>CPU Quota:</p> <ul style="list-style-type: none"> - Request: minimum number of CPU cores required by a container. The default value is 0.25 cores. - Limit: maximum number of CPU cores available for a container. Do not leave Limit unspecified. Otherwise, intensive use of container resources will occur and your workload may exhibit unexpected behavior. <p>Memory Quota:</p> <ul style="list-style-type: none"> - Request: minimum amount of memory required by a container. The default value is 512 MiB. - Limit: maximum amount of memory available for a container. When memory usage exceeds the specified memory limit, the container will be terminated. <p>For details, see Setting Container Specifications.</p> <p>GPU: configurable only when the cluster contains GPU nodes.</p> <ul style="list-style-type: none"> - GPU: the percentage of GPU resources reserved for a container. Select Use 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 Use or set this parameter to 0, no GPU resources can be used. |

| Parameter in CCE 2.0 | Parameter in CCE 1.0 | Configuration |
|----------------------|----------------------|---|
| | | <ul style="list-style-type: none"> - GPU/Graphics Card: The workload's pods will be scheduled to the node with the specified GPU. If Any GPU type is selected, the container uses a random GPU in the node. If you select a specific GPU, the container uses that GPU accordingly. |

3. Set the environment variables, data storage, and user-defined log.

Table 22-6 Configuring advanced settings

| Parameter in CCE 2.0 | Parameter in CCE 1.0 | Configuration |
|----------------------|---|--|
| Lifecycle | This parameter does not exist in CCE 1.0. For migrated applications, you do not need to set this parameter. | <p>Set the commands for starting and running containers.</p> <ul style="list-style-type: none"> - Startup Command: Executed to start a container. For details, see Setting Container Startup Commands. - Post-Start: Executed after a container runs successfully. For details, see Setting Container Lifecycle Parameters. - Pre-Stop: Executed to delete logs or temporary files before a container is stopped. For details, see Setting Container Lifecycle Parameters. |

| Parameter in CCE 2.0 | Parameter in CCE 1.0 | Configuration |
|------------------------------|---|--|
| Health check | This parameter does not exist in CCE 1.0. For migrated applications, you do not need to set this parameter. | <p>Configure the health check function to check whether containers and services are running properly. Two types of probes are provided: liveness probes and readiness probes. For details, see Setting Health Check for a Container.</p> <ul style="list-style-type: none"> - 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. |
| Adding environment variables | Adding environment variables | <p>In the Environment Variables area, click Add Environment Variable. Currently, environment variables can be added using any of the following methods:</p> <ul style="list-style-type: none"> - Added manually: Set Variable Name and Variable Value. - Added from Secret: Set Variable Name and select the desired secret name and data. You need to create a secret in advance. For details, see Creating a Secret. - Added from ConfigMap: Set Variable Name and select the desired ConfigMap name and data. You need to create a ConfigMap in advance. For details, see Creating a ConfigMap. |

| Parameter in CCE 2.0 | Parameter in CCE 1.0 | Configuration |
|----------------------|---|--|
| Data Storage | Volume | <p>For the applications using the old component template, perform the following operations:</p> <ol style="list-style-type: none"> 1. Choose Data Storage > Local Volume. Click Add Local Volume. 2. Select HostPath. 3. Set the following parameters: <ul style="list-style-type: none"> ▪ Host Path: Path of the host to which the local volume is to be mounted, corresponding to /tmp of volumes. ▪ Click Add Container Path, enter the container path to which the data volume is mounted. It corresponds to /test of Volumes. ▪ Permission: Read/Write. 4. When the configuration is complete, click OK. |
| Security Settings | This parameter does not exist in CCE 1.0. For migrated applications, you do not need to set this parameter. | Set container permissions to protect the system and other containers from being affected. Enter the user ID to set container permissions and prevent systems and other containers from being affected. |
| Container Logs | This parameter does not exist in CCE 1.0. For migrated applications, you do not need to set this parameter. | Set a policy and log directory for collecting application logs and preventing logs from being over-sized. For details, see Collecting Container Logs from Specified Paths . |

Step 4 Click **Next**. 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 [Network Management](#).


- **Access Type:** Select **LoadBalancer (ELB)**.
- **Service Name:** Specify a service name. You can use the workload name as the service name.
- **Service Affinity:**
 - **Cluster level:** External traffic is routed to all nodes in the cluster while masking clients' source IP addresses.
 - **Node level:** External traffic is routed to the node where the load balancer used by the service is located, without masking clients' source IP addresses.
- **Elastic Load Balancer:** A load balancer automatically distributes Internet access traffic to multiple nodes running the workload. 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).
 - **Public network:** You can select an existing public network load balancer or have the system automatically create a new public network load balancer.

If you have the system automatically create a public network load balancer, you can click **Change Configuration** to modify its name, EIP type, billing mode, and bandwidth.
 - **Private network:** You can select an existing private network load balancer or have the system automatically create a new private network load balancer.
- **Port Settings**
 - **Protocol:** protocol used by the Service.
 - **Container Port:** a port that is defined in the container image and on which the workload listens. The Nginx application listens on port 80.
 - **Access Port:** a 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 5 Click **OK**, and then click **Next**. Skip advanced settings.

Step 6 Click **Create** after the configuration is complete. Click **Back to Deployment List**.

If the deployment is in the **Running** state, the deployment is successfully created.

Workload status is not updated in real time. Click  in the upper right corner or press **F5** to refresh the page.

Step 7 To access the workload in a browser, copy the workload's **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)**.

----End

22.4.3 Applications Created Through App Designer

If the application in CCE 1.0 version is created through App Designer, follow the steps in this section to migrate the application to CCE 2.0. In CCE 2.0, use the custom Helm chart to create applications. For more information, see [Chart Management](#).

Procedure

- Step 1** Log in to the console of CCE 1.0. In the navigation pane, click **App Manager**. Then, click the name of the application to be migrated to go to the details page. Obtain **Component Name** and **Service Name**.
- Step 2** Connect to a cluster. For details, see [Connecting to a Cluster Through kubectl](#).
- Step 3** Run the following commands on the kubectl client to obtain the YAML files related to the application.

```
kubectl get rc new-appcomponent-1-11d3 -o yaml > new-appcomponent-1-11d3.yaml
```

```
kubectl get svc new-port-3 -o yaml > new-port-3.yaml
```

In the preceding commands, replace *new-appcomponent-1-11d3* and *new-port-3* with the values obtained in [Step 1](#).

- Step 4** Edit the *new-appcomponent-1-11d3.yaml* file by deleting the lines in bold and modifying the lines in Italic.

```
apiVersion: v1
kind: ReplicationController
metadata:
  annotations:
    cce/app-createTimestamp: 2018-04-19-11-39-27
    cce/app-description: ""
    cce/app-updateTimestamp: 2018-04-19-11-39-27
creationTimestamp: 2018-04-19T11:37:17Z
generation: 1
  labels:
    cce/appgroup: app-design
    name: new-appcomponent-1-11d3
    rollingupdate: "false"
  name: new-appcomponent-1-11d3
  namespace: default
resourceVersion: "8325781"
selfLink: /api/v1/namespaces/default/replicationcontrollers/new-appcomponent-1-11d3
uid: 039ada96-43c6-11e8-8f34-fa163e738aa3
spec:
  replicas: 1
  selector:
    cce/appgroup: app-design
    name: new-appcomponent-1-11d3
    rollingupdate: "false"
  template:
    metadata:
```

```

annotations:
  scheduler.alpha.kubernetes.io/affinity: '{"nodeAffinity":
{"requiredDuringSchedulingIgnoredDuringExecution":{"nodeSelectorTerms":[{"matchExpressions":
[{"key":"failure-domain.beta.kubernetes.io/zone","operator":"NotIn","values":[""]}]}]}}'
  creationTimestamp: null
labels:
  cce/appgroup: app-design
  name: new-appcomponent-1-11d3
  rollingupdate: "false"
spec:
  affinity:
    nodeAffinity:
      requiredDuringSchedulingIgnoredDuringExecution:
        nodeSelectorTerms:
          - matchExpressions:
            - key: failure-domain.beta.kubernetes.io/zone
              operator: NotIn
              values:
                - ""
  containers:
    - image: 10.125.1.108:6443/test/apache-php:latest -->Change it to the image address of the later
CCE version
      imagePullPolicy: Always
      name: new-container-2-11d3f16
      ports:
        - containerPort: 3333
          protocol: TCP
      resources: {}
      securityContext:
        privileged: true
      terminationMessagePath: /dev/termination-log
      terminationMessagePolicy: File
      dnsPolicy: ClusterFirst
      imagePullSecrets:
        - name: myregistry -----> Change it to default-secret
      restartPolicy: Always
      schedulerName: default-scheduler
      securityContext: {}
      terminationGracePeriodSeconds: 30
status:
availableReplicas: 1
fullyLabeledReplicas: 1
observedGeneration: 1
readyReplicas: 1
replicas: 1

```

Step 5 Open the **new-port-3.yaml** file, and delete the lines in bold.

```

apiVersion: v1
kind: Service
metadata:
  creationTimestamp: 2018-04-19T11:37:17Z
  labels:
    cce/appgroup: app-design
    name: new-appcomponent-1-11d3
  name: new-port-3
  namespace: default
  resourceVersion: "8325766"
  selfLink: /api/v1/namespaces/default/services/new-port-3
  uid: 039c918f-43c6-11e8-8f34-fa163e738aa3
spec:
  clusterIP: 10.247.145.136
  externalTrafficPolicy: Cluster
  ports:
    - name: new-port-30
      nodePort: 31471
      port: 3333
      protocol: TCP
      targetPort: 3333
  selector:

```

```
cce/appgroup: app-design
  name: new-appcomponent-1-11d3
  sessionAffinity: None
  type: NodePort
status:
loadBalancer: {}
```

Step 6 Connect the kubectl client to the CCE cluster of the latest version. For details, see [Connecting to a Cluster Through kubectl](#).

Step 7 Run the following commands to create the application again:

```
kubectl create -f new-appcomponent-1-11d3.yaml
```

```
kubectl create -f new-port-3.yaml
```

```
----End
```