Cloud Container Engine

FAQs

Issue 01

Date 2025-07-17

HUAWEI CLOUD COMPUTING TECHNOLOGIES CO., LTD.

Trademarks and Permissions


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

Notice

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

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

Contents

1 Common FAQ

Cluster Management

- Why Cannot I Create a CCE Cluster?
- Is Management Scale of a Cluster Related to the Number of Master Nodes?
- How Do I Locate the Fault When a Cluster Is Unavailable?

Node/Node Pool Management

- What Should I Do If a Cluster Is Available But Some Nodes in It Are Unavailable?
- How Do I Collect Logs of Nodes in a CCE Cluster?
- How Do I Fix an Abnormal Container or Node Due to No Thin Pool Disk Space?
- What Should I Do If a Node Cannot Be Managed and an Error Message Appears Saying That the Node Failed to Install?

Workload Management

- How Can I Locate the Root Cause If a Workload Is Abnormal?
- What Should I Do If the Scheduling of a Pod Fails?
- What Should I Do If a Pod Fails to Pull the Image?
- What Should I Do If a Pod Startup Fails?
- What Should I Do If a Pod Remains in the Terminating State?
- What Should I Do If a Pod Fails to Be Evicted?
- How Can I Locate Faults Using an Exit Code?
- How Do I Evenly Distribute Multiple Pods to Each Node?
- How Do I Evict All Pods on a Node?

Network Management

- How Can I Configure a Security Group Rule for a Cluster?
- How Do I Locate a Workload Networking Fault?
- Why Does the Browser Return Error Code 404 When I Access a Deployed Application?

- What Should I Do If a Node Fails to Access the Internet?
- How Do I Optimize the Configuration If the External Domain Name Resolution Is Slow or Times Out?

Storage Management

- Why Cannot I Delete a PV or PVC Using the kubectl delete Command?
- How Do I Expand the Storage Capacity of a Container?

API & kubectl

Why Is "Error from server (Forbidden)" Displayed When I Use kubectl?

Image Repository

How Do I Create a Docker Image and Solve the Problem of Slow Image Pull?

2 Billing

2.1 How Is CCE Billed?

Billing Modes

There are yearly/monthly and pay-per-use billing modes to meet your requirements. For details, see **Billing Modes**.

- Yearly/Monthly: You pay upfront for the amount of time you expect to use a resource for. You will need to make sure you have a top-up account with a sufficient balance or have a valid payment method configured first.
- Pay-per-use: You can start using CCE resources first and then pay as you go.

After purchasing CCE clusters or cluster resources, you can change their billing modes if the current billing mode cannot meet your service requirements. For details, see **Billing Mode Changes**.

Billing Items

You will be billed for clusters, nodes, and other cloud service resources. For details about the billing factors and formulas for each billed item, see **Billed Items**.

1. **Clusters**: the cost of resources used by master nodes. It varies with the cluster type (VMs or BMSs and the number of master nodes) and size (the number of worker nodes).

   For more details, see **CCE Pricing Details**.

2. **Other cloud resources**: the cost of IaaS resources in use. Such resources, which are created either manually or automatically during cluster creation, include ECSs, EVS disks, EIPs, bandwidth, and load balancers.

   For more pricing details, see **Product Pricing Details**.

For more information about the billing samples and the billing for each item, see **Billing Examples**.

2.2 How Do I Change the Billing Mode of a CCE Cluster from Pay-per-Use to Yearly/Monthly?

Currently, clusters support **pay-per-use** and **yearly/monthly** billing modes. A pay-per-use cluster can be converted to a yearly/monthly-billed cluster.

Changing the Billing Mode of a Cluster

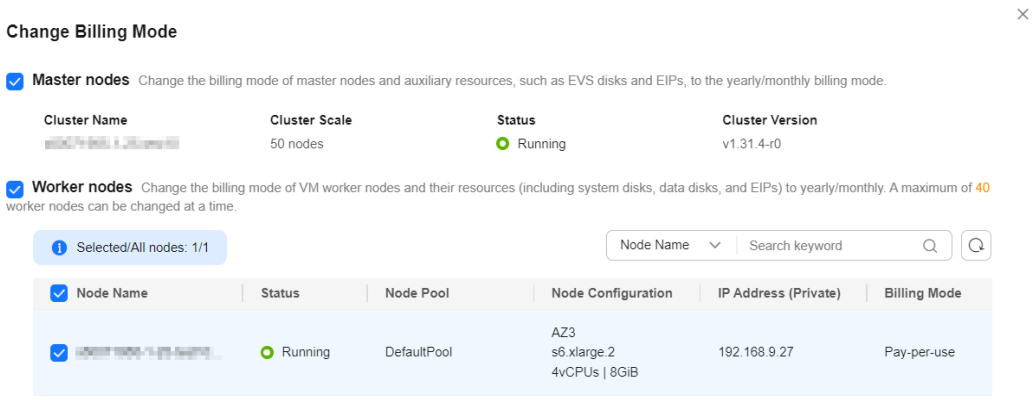
To change the billing mode of a cluster from pay-per-use to yearly/monthly, perform the following operations:

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

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

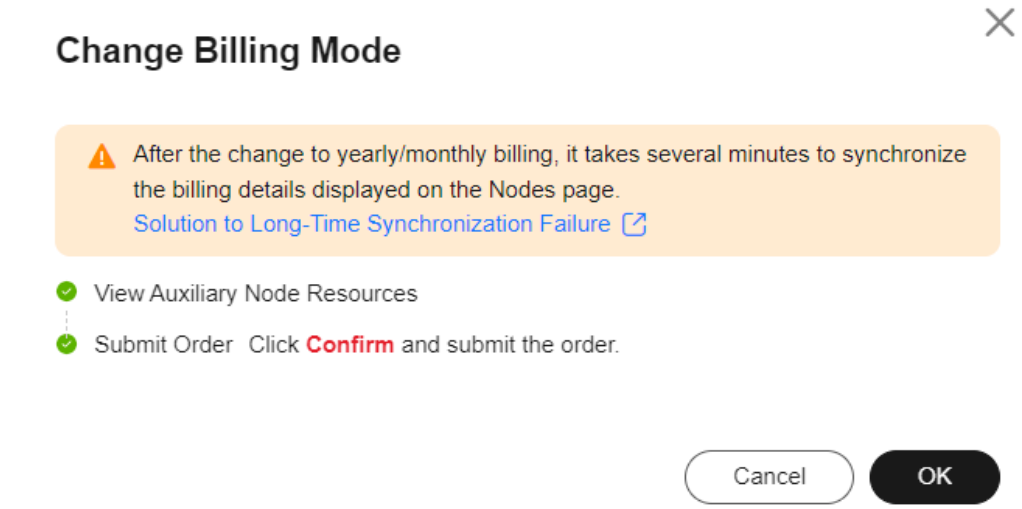
Step 3 On the page displayed, select the target cluster and click **OK**. You can also select the nodes whose billing modes you want to change.

Figure 2-1 Changing the billing mode of a cluster to yearly/monthly



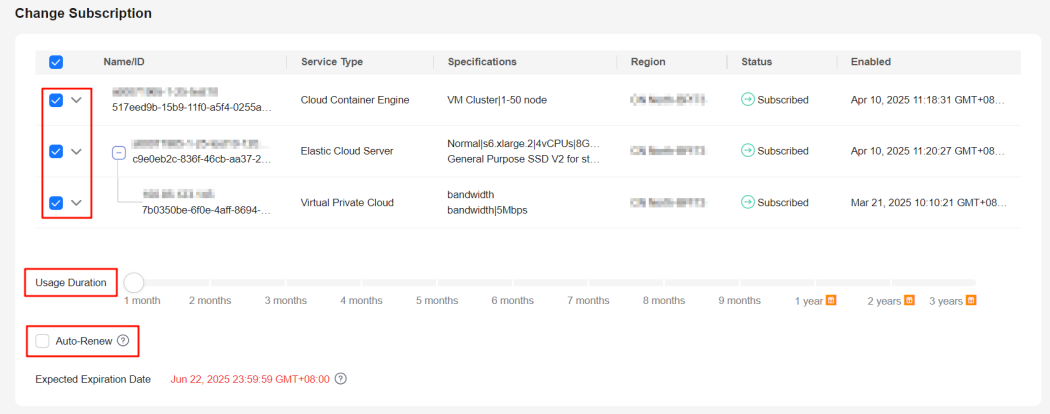
Step 4 Wait for the resource checks and the order generation, click **OK** to go to the Billing Center, and submit the order.

Figure 2-2 Generating an order



Step 5 Select the resources whose billing modes are to be changed to yearly/monthly, set the required duration, enable auto-renew, and click **Pay**.

Figure 2-3 Changing the billing mode from pay-per-use to yearly/monthly



----End

2.3 Can I Change the Billing Mode of CCE Nodes from Pay-per-Use to Yearly/Monthly?

Currently, nodes support **pay-per-use** and **yearly/monthly** billing modes.

Notes and Constraints

- To change the billing mode of a node in a pay-per-use node pool to yearly/monthly, you need to upgrade the cluster to v1.19.16-r40, v1.21.11-r0, v1.23.9-r0, v1.25.4-r0, or later.
- After a node in a pay-per-use node pool is changed to a yearly/monthly node, the node does not support elastic scale-in.

Changing the Billing Mode of a Node

NOTICE

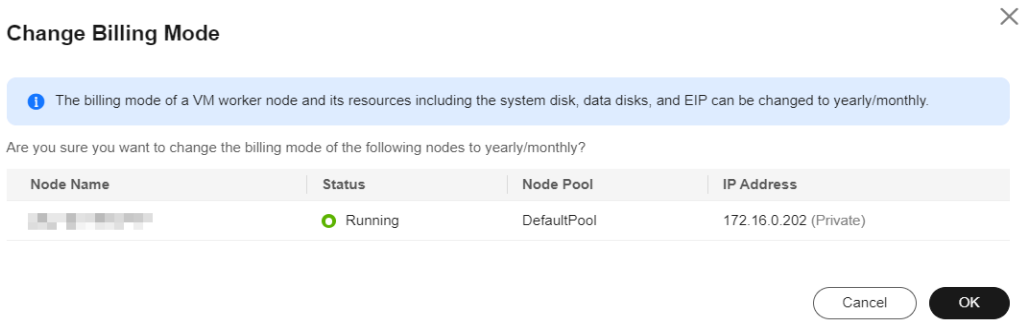
- The billing modes of resources like EVS disks and EIPs used by pay-per-use nodes cannot be changed simultaneously. For details, see **Pay-per-Use to Yearly/Monthly**.
- To change a pay-per-use node in a node pool to a yearly/monthly one, locate the target node in the node list, choose **More** > **Forbid node pool scale-in** above the list, and change the billing mode to yearly/monthly.

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

Step 1 Log in to the CCE console and click the cluster name to access the cluster console.

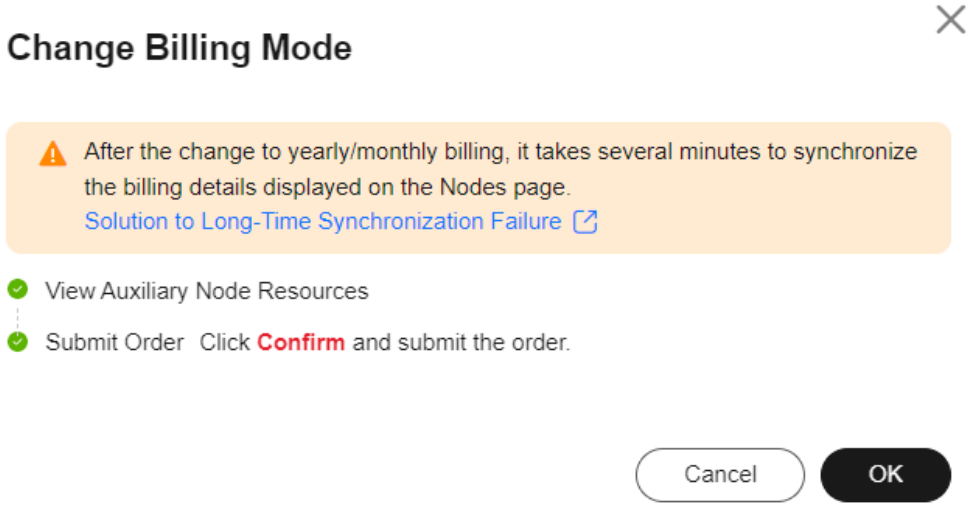
Step 2 In the navigation pane, choose **Nodes**. In the right pane, click the **Nodes** tab, locate the row containing the target node, and choose **More** > **Change Billing Mode** in the **Operation** column. In the dialog box displayed, click **OK**.

Figure 2-4 Changing the billing mode of a node to yearly/monthly



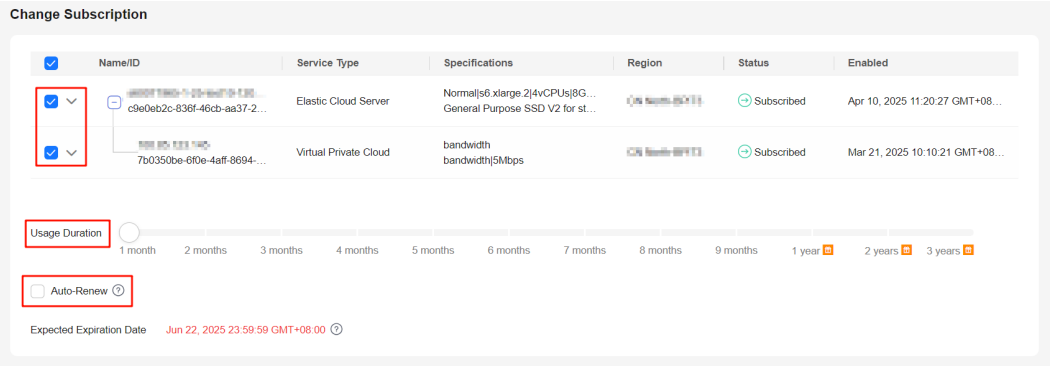
Step 3 Wait for the resource checks and the order generation, click **OK** to go to the Billing Center, and submit the order.

Figure 2-5 Generating an order



Step 4 Select the resources whose billing modes are to be changed to yearly/monthly, set the required duration, enable auto-renew, and click **Pay**.

Figure 2-6 Changing the billing mode from pay-per-use to yearly/monthly



----End

2.4 Which Invoice Modes Are Supported by Huawei Cloud?

Huawei Cloud allows you to issue invoices by billing cycle and by order.

You can issue invoices on the **Invoices** page in **Billing Center**.

2.5 Will I Be Notified When My Balance Is Insufficient?

In Billing Center, on the **Overview** page, click **Settings** in the **Available Credit** area. Then, you can enable or disable **Balance Alert**. Click **Modify** and configure 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.
- In Message Center, choose **Message Receiving Management** > **SMS & Email Settings** in the navigation pane, select **Account balance** under **Finance** to change contacts that receive the balance alerts.

2.6 Will I Be Notified When My Account Balance Changes?

The system will notify you via emails or SMS messages of your account balance changes, including whether your online topping up is successful.

2.7 Can I Delete a Yearly/Monthly-Billed CCE Cluster Directly When It Expires?

After a yearly/monthly-billed cluster expires, you can delete the cluster after all data is backed up.

If you do not renew or delete the cluster after it expires, the system will delete the cluster based on the resource expiration time. You are advised to renew the cluster and back up data in a timely manner.

2.8 How Do I Unsubscribe from CCE?

Yearly/monthly-billed CCE resources can be unsubscribed from, including the renewed part and currently used part. You cannot use these resources after unsubscription. A handling fee will be charged for unsubscribing from a resource.

Notes

- Unsubscribing from CCE resources involves the renewed resources and the resources that are being used. After the unsubscription, these resources become unavailable.
- Solution portfolios can only be unsubscribed from as a whole.
- If an order contains resources in a primary-secondary relationship, you need to unsubscribe from the resources separately.
- For details about unsubscribing from resources, see **Unsubscription Rules**.

Procedure

CAUTION

- Before requesting an unsubscription, ensure that you have migrated or backed up any data saved on CCE resources that will be unsubscribed from. After the unsubscription is complete, CCE resources and any data contained will be permanently deleted.
- The middle of the unsubscription page displays a message showing the number of unsubscriptions you have performed and the remaining allowed number.

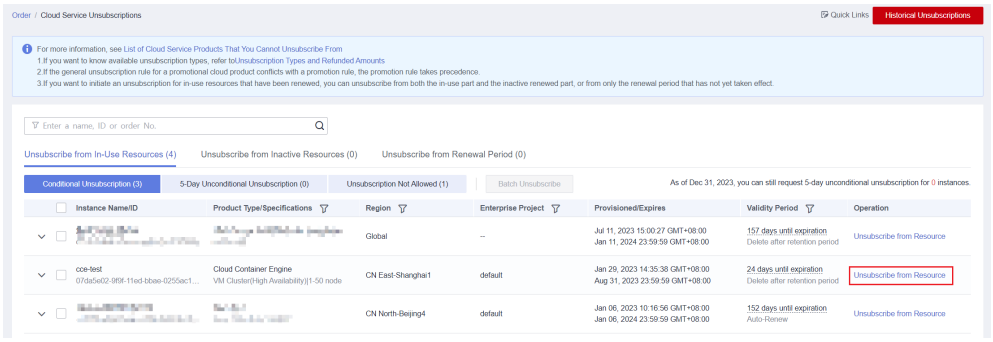
Step 1 Enter the **Unsubscriptions** page.

Step 2 Click the **Unsubscribe from In-Use Resources** tab.

Step 3 Unsubscribe from a single resource or from resources in a batch.

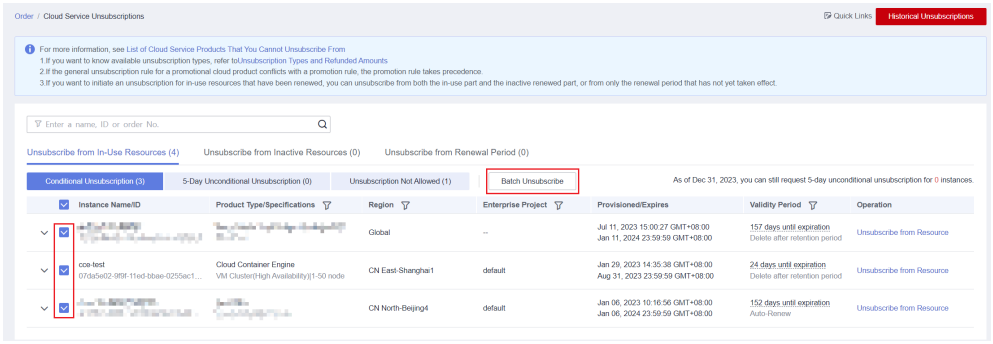
- To unsubscribe from a single resource, click **Unsubscribe from Resource** at the row of the target resource.

Figure 2-7 Unsubscribing from a single resource



- To unsubscribe from resources in a batch, select the target resources from the list and click **Batch Unsubscribe** above the list.

Figure 2-8 Unsubscribing from resources in a batch



Step 4 On the **Unsubscribe from In-Use Resources** page, confirm the information, select a reason for the unsubscription, and click **Confirm**.

----End

3 Cluster

3.1 Cluster Creation

3.1.1 Why Cannot 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**.
2. The cluster does not have sufficient underlying resources. You can create a new cluster with the appropriate type and scale.
3. Check whether your account is in arrears. If so, you cannot purchase resources, including using cash coupons. For details, see **Topping Up an Account (Prepaid Direct Customers)**.

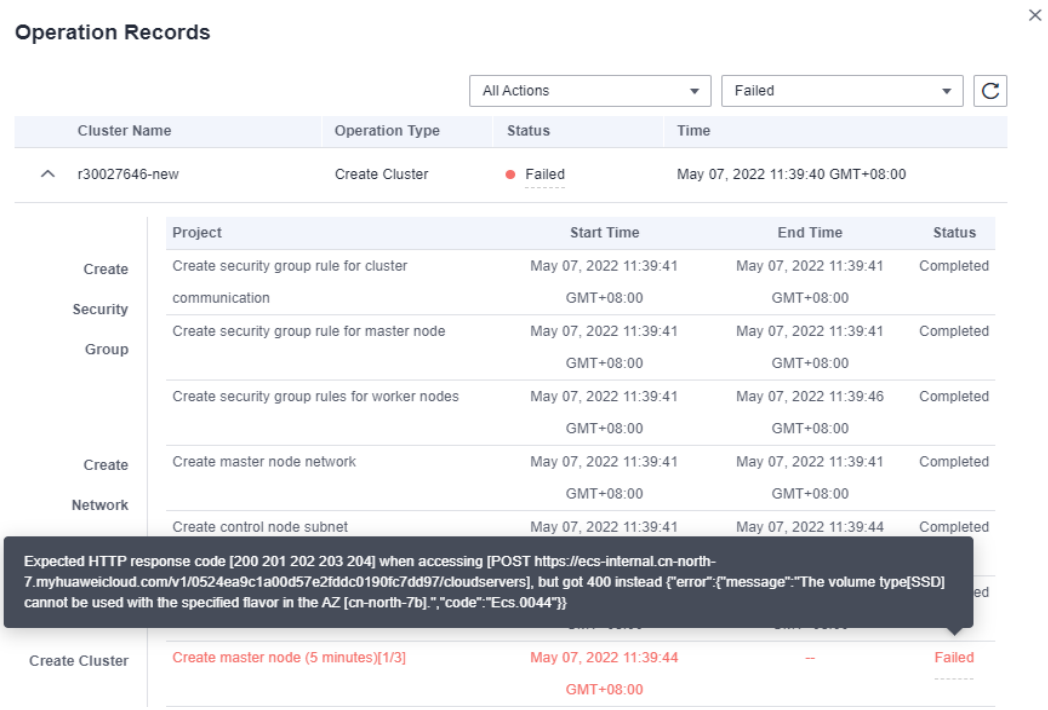
Locating the Failure Cause

View the cluster logs to identify the cause and rectify the fault.

Step 1 Log in to the CCE console and click **Operation Records** above the cluster list to view operation records.

Step 2 Click the record of the **Failed** status to view error information.

Figure 3-1 Viewing the operation details



Step 3 Rectify the fault based on the error information and create a cluster again.

----End

3.1.2 Is Management Scale of a Cluster Related to the Number of Master Nodes?

In a CCE cluster, the management scale is not directly related to the number of master nodes. These are cluster parameters that operate in different dimensions. Here are the details:

- Cluster management scale: indicates the maximum number of nodes that a cluster can manage. For example, if you choose 50 nodes, the cluster can manage up to 50 worker nodes.

  The flavors of master nodes may vary with the cluster scale. However, the number of master nodes is not affected by the management scale.
- Number of master nodes: The number of master nodes impacts the high availability of the cluster. To enhance the cluster's DR capabilities, you can choose multiple master nodes.

  For example, you can choose three masters for your cluster. If one of them is faulty, the cluster can still run properly. The services will not be affected.

3.1.3 How Do I Update the Root Certificate When Creating a CCE Cluster?

The root certificate of CCE clusters is the basic certificate for Kubernetes authentication. Both the Kubernetes cluster control plane and the certificate are hosted on Huawei Cloud CCE. CCE will periodically update the certificate. This certificate is not open to users but will not expire.

The X.509 certificate is enabled on Kubernetes clusters by default. CCE will automatically maintain and update the X.509 certificate.

Obtaining a Cluster Certificate

You can obtain a cluster certificate on the CCE console to access Kubernetes. For details, see **Obtaining a Cluster Certificate**.

3.1.4 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 Quota?

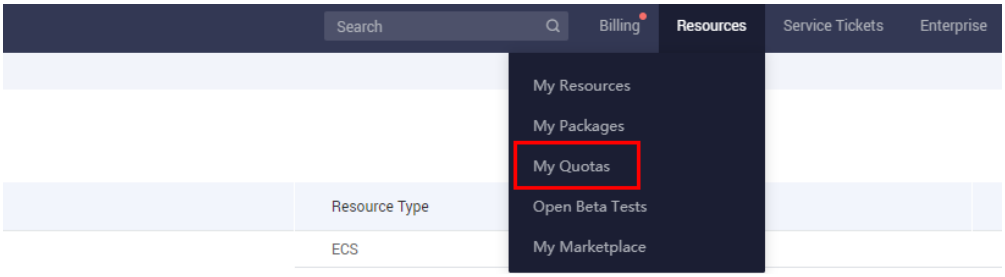
Quotas are enforced for service resources on the platform to prevent unforeseen spikes in resource usage. Quotas can limit the quantity and capacity of resources available to users, such as the maximum number of ECSs or EVS disks that can be created.

If a quota cannot meet your needs, apply for a higher quota.

How Do I View My Quota?

1. Log in to the management console.
2. Click in the upper left corner to select a region and a project.
3. In the upper right corner of the page, choose **Resources** > **My Quotas**.
   The **Service Quota** page is displayed.

   Figure 3-2 My Quotas

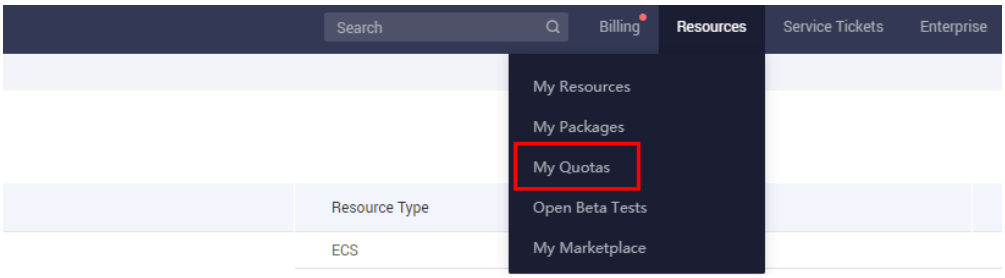


4. On this page, view the total quota and used quota of resources.
   If a quota cannot meet your service requirements, click **Increase Quota**.

How Do I Increase My Quota?

1. Log in to the management console.
2. In the upper right corner of the page, choose **Resources** > **My Quotas**.
   The **Service Quota** page is displayed.

Figure 3-3 My Quotas



3. Click **Increase Quota**.
4. On the **Create Service Ticket** page, configure parameters as required and submit a service ticket.

   In the **Problem Description** area, enter the required quota and reason for the adjustment.
5. Select **I have read and agree to the Ticket Service Protocol and Privacy Statement.** and click **Submit**.

3.2 Cluster Running

3.2.1 How Do I Locate the Fault When a Cluster Is Unavailable?

This section provides you with some operations to locate the fault when a cluster becomes unavailable.

Fault Locating

Possible causes are described here in order of how likely they are to occur.

If the fault persists after you have ruled out a cause, check other causes.

- **Check Item 1: Whether the Security Group Is Modified**
- **Check Item 2: Whether the Cluster Is Overloaded**
- **Check Item 3: Whether the KMS Key Used for Secret Encryption Is Valid**

If the fault persists, **submit a service ticket and** contact the customer service to help you locate the fault.

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, choose **Access Control** > **Security Groups** and 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, see **How Can I Configure a Security Group Rule for a Cluster?**

----End

Check Item 2: Whether the Cluster Is Overloaded

Symptom

The resource usage on the master nodes in the cluster reaches 100%.

Possible Cause

When a cluster has a large number of resources created simultaneously, it causes an overload on the API server. This, in turn, overloads the master nodes and leads to OOM issues.

Solution

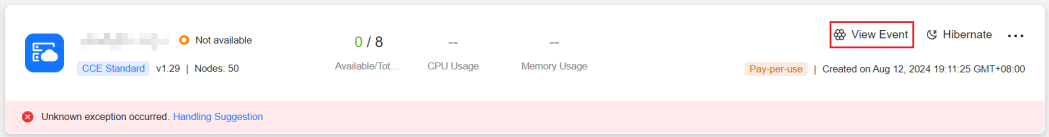
Increase the cluster management scale. A larger cluster management scale means higher capacity and improved performance of the master nodes. For details, see **Changing Cluster Scale**.

If a cluster is overloaded, you can submit a service ticket for technical support.

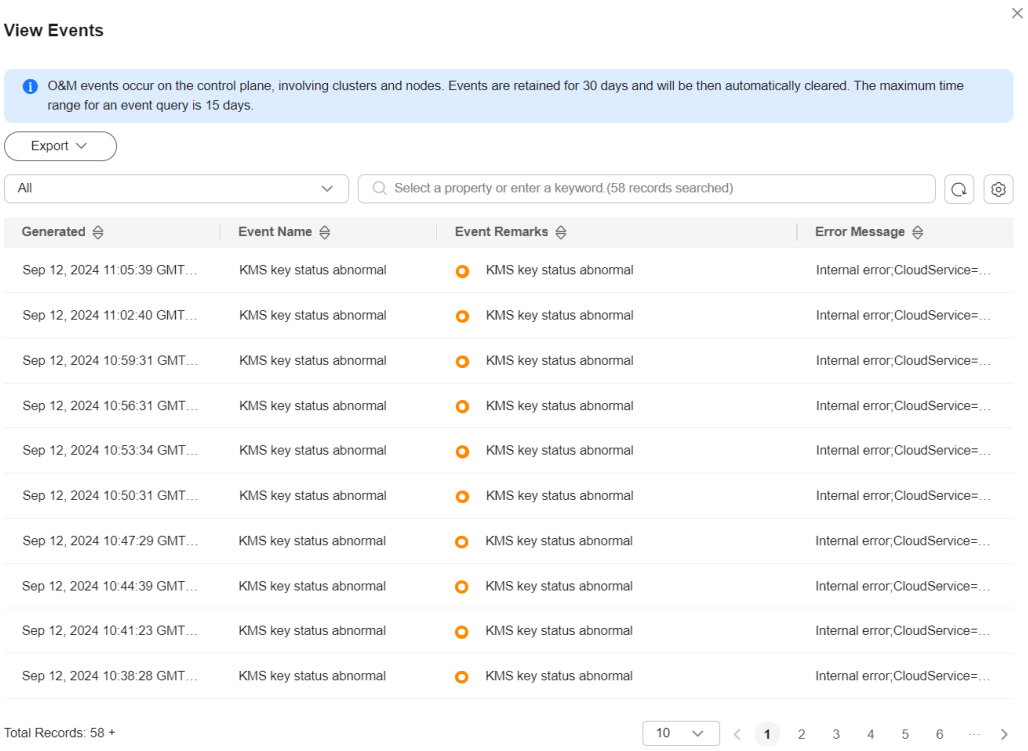
Check Item 3: Whether the KMS Key Used for Secret Encryption Is Valid

Symptom

If a cluster is unavailable, you can check the cluster event to locate the fault.



If **KMS key status abnormal** is displayed in the events, check whether the key used by the cluster is in the **Disabled** or **Pending deletion** state.



Solution

Step 1 Log in to the DEW console.

Step 2 In the custom key list, find the KMS key used by the cluster.

- For a key in the **Pending deletion** state, click **Cancel Deletion** in the **Operation** column. If the key remains in a **Disabled** state even after cancellation, then cancel the action of disabling the key.
- For a key in the **Disabled** state, click **Enable** in the **Operation** column.

Step 3 Verify whether the key has been enabled and wait for the cluster to be automatically restored. The restoration process should take about 5 to 10 minutes.

----End

3.2.2 How Do I Reset or Reinstall a CCE Cluster?

CCE clusters cannot be reset or reinstalled. If a cluster becomes unavailable, **submit a service ticket** or delete the cluster and purchase a new one.

CCE supports resetting nodes. For details, see **Resetting a Node**.

3.2.3 How Do I Check Whether a Cluster Is in Multi-Master Mode?

Log in to the CCE console and click the cluster. On the right of the cluster details page, view the number of master nodes.

- **3**: The cluster is in multi-master mode.

- **1**: The cluster is in single-master mode.

NOTICE

The number of master nodes cannot be changed after the cluster is created. If you want to adjust the number, you need to create a new cluster.

3.2.4 Can I Directly Connect to the Master Node of a CCE Cluster?

CCE allows you to use kubectl to connect a cluster. For details, see **Connecting to a Cluster Using kubectl**.

However, you are not allowed to log in to the master node to perform related operations.

3.2.5 How Do I Retrieve Data After a CCE Cluster Is Deleted?

Question

How do I retrieve data after a CCE cluster is deleted?

Answer

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.

3.2.6 Why Does CCE Display Node Disk Usage Inconsistently with Cloud Eye?

Symptom

The disk usage of a node on the CCE cluster details page is higher than 80%, but the disk usage displayed on the Cloud Eye console is lower than 40%.

After fault locating on the node, it is found that the usage of a PVC disk reaches 92%. After the disk is cleared, the disk usage on CCE is the same as that on Cloud Eye.

Does CCE display only the highest disk usage?

Answer

In the CCE cluster monitoring information, the disk with the highest disk usage on the node is monitored.

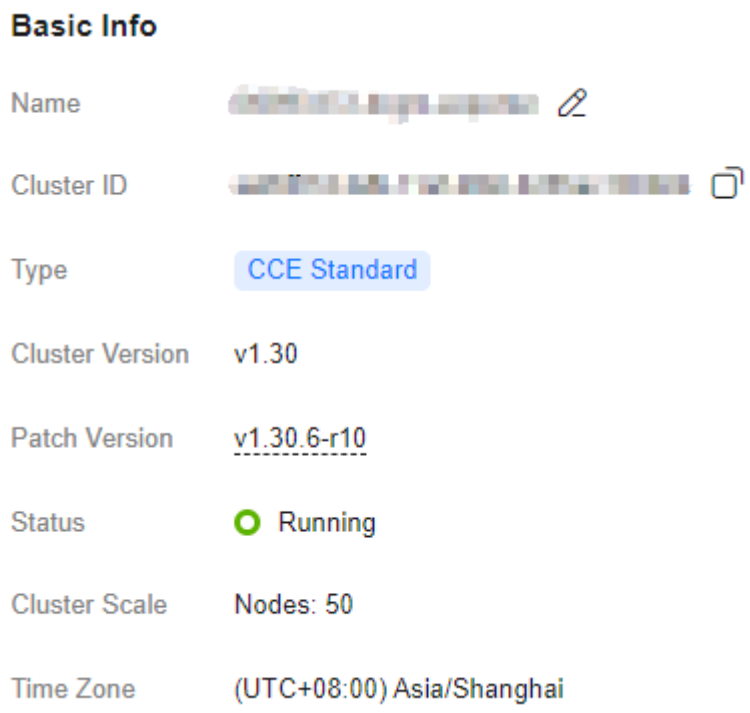
3.2.7 How Do I Change the Name of a CCE Cluster?

After a cluster is created, you can change its name.

Step 1 Log in to the CCE console and click the cluster name to access the cluster **Overview** page.

Step 2 In the **Basic Info** area, click  next to the cluster name.

Figure 3-4 Changing the cluster name



Step 3 On the page displayed, enter a new name, which must contain 4 to 128 characters, start with a lowercase letter, and not end with a hyphen (-). Only lowercase letters, digits, and hyphens (-) are allowed.

Click **Save**. Once the changes are saved, the cluster name will update automatically to reflect the new name.

NOTE

- The new name cannot be the same as its original name or the name of another cluster.
- After the name is changed, the CTS, RMS, and EPS records will be renamed accordingly.

----End

3.2.8 How Can I Identify the Cause of an Exception When There Is an Issue with Console Access?

An Error Reported When Pod Logs Are Accessed

If you encounter this problem when viewing pod logs in a cluster but other resources in the cluster can be accessed, take the following steps:

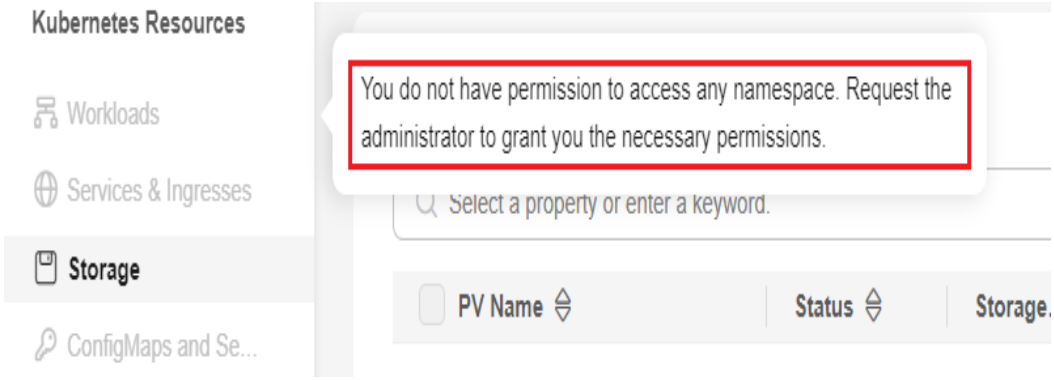
1. Log in to the CCE console and click the cluster name to access the cluster console.
2. In the navigation pane, choose **Workloads**. In the right pane, click the **Pods** tab and check whether the target pod is in the **Running** state. If it is not, locate the fault based on **How Can I Locate the Root Cause If a Workload Is Abnormal?**

3. In the navigation pane, choose **Overview**. In the **Networking Configuration** area, click the name of the default node security group to go to the details page and check the inbound rules.
4. Check all security group rules and see whether the inbound access from the VPC to TCP port 10250 is enabled. If it is not, add the preceding rules to the security group.
5. If there are other issues, submit a service ticket.

Required RBAC Permissions Not Granted to the Account

Symptom

When you access the console, an error message "You do not have permission to access any namespace. Request the administrator to grant you the necessary permissions." is displayed.



Possible Cause

Your account is not granted the required RBAC permissions.

Solution

1. Log in to the IAM management console using a Huawei Cloud account or an account with the administrator permissions. In the navigation pane, choose **Users**.
2. On the **Users** tab, locate the row containing the username that reports the error and click **Authorize**.
3. Select the required permissions and click **OK**.

Required IAM Permissions Not Granted to the Account

Symptom

When you try to access the console, you will see an error message indicating that you do not have the necessary permissions. The error code for this issue is CCE.01403001.

Possible Cause

Your account is not granted the required IAM permissions.

Solution

1. Log in to the IAM management console using a Huawei Cloud account or an account with IAM permissions.
2. Based on the error message, add **permissions required by the CCE console** to your account. For details about authorization, see **Granting Cluster Permissions to an IAM User**.

3.3 Cluster Deletion

3.3.1 What Can I Do If a Cluster Deletion Fails Due to Residual Resources in the Security Group?

When deleting a cluster, CCE obtains the cluster's resources, such as the elastic network interfaces or supplementary network interfaces bound to a CCE Turbo cluster 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

The cluster cannot be deleted, and the following error information is displayed:

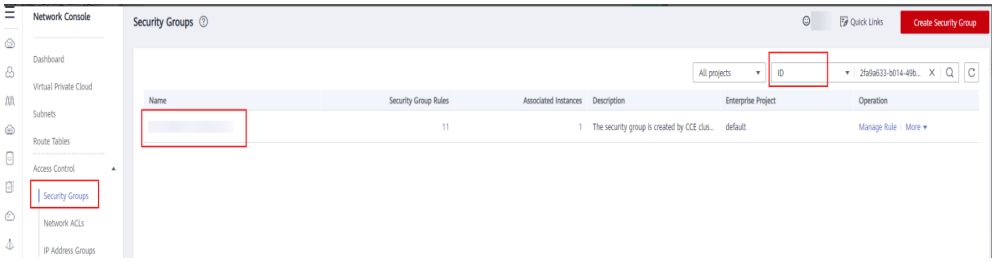
```
Expected HTTP response code [200 202 204 404] when accessing [DELETE https://vpc.***.com/v2.0/security-groups/46311976-7743-4c7c-8249-ccd293bcae91], but got 409 instead
{"code":"VPC.0602","message":"{\"NeutronError\":{\"message\": \"Security Group 46311976-7743-4c7c-8249-ccd293bcae91 in use.\",\"type\":\"SecurityGroupInUse\",\"detail\":\"\"}}"}
```

Possible Cause

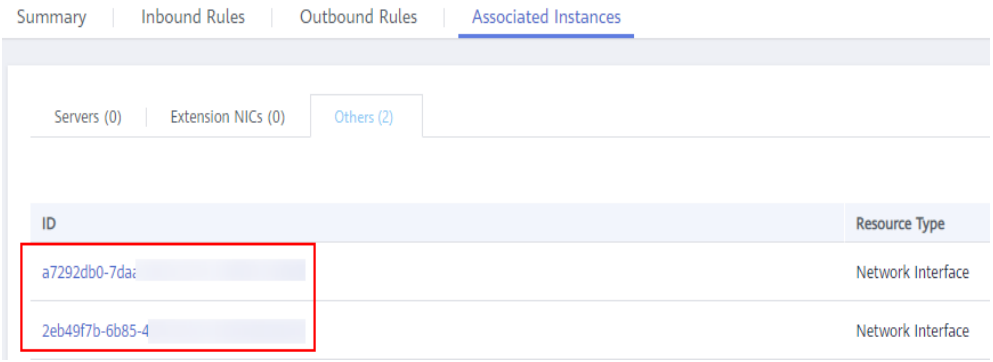
The cluster's security group has undeleted resources, preventing its deletion and causing the creation of the cluster to fail.

Procedure

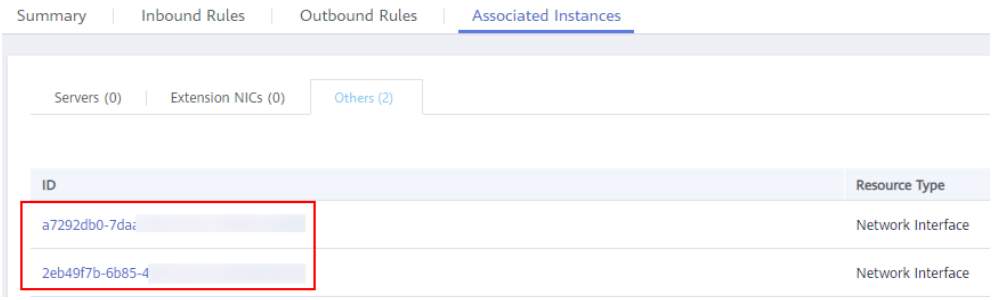
Step 1 Copy the resource ID in the error information, go to the **Security Groups** page of the VPC console, and obtain security groups by ID.



Step 2 Click the security group to view its details, and click the **Associated Instances** tab.

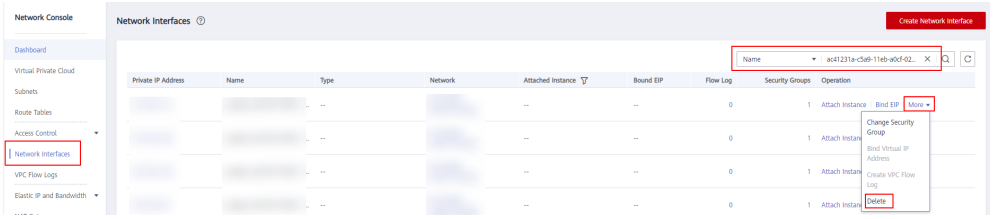


Obtain other resources associated with the security group, such as elastic network interfaces, supplementary network interfaces, and servers. You can delete the residual resources. The supplementary network interfaces will be automatically deleted.



Step 3 For a residual elastic network interface, go to the **Network Interfaces** page and delete the elastic network interface obtained in the previous step.

You can search for the elastic network interfaces to be deleted by IDs or names.



Step 4 Go to the **Security Groups** page to confirm that the security group is not associated with any instance. Then, go to the CCE console to delete the cluster.

----End

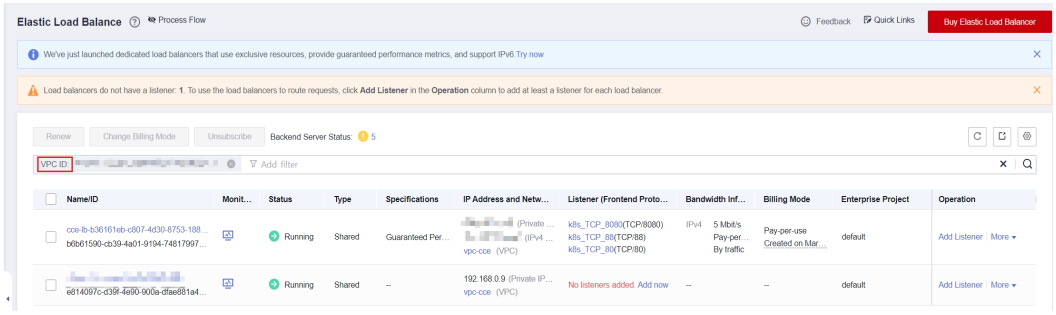
3.3.2 How Do I Clear Residual Resources After Deleting a Non-Running Cluster?

If a cluster is not in the running state (for example, frozen or unavailable), its resources such as PVCs, Services, and Ingresses 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.

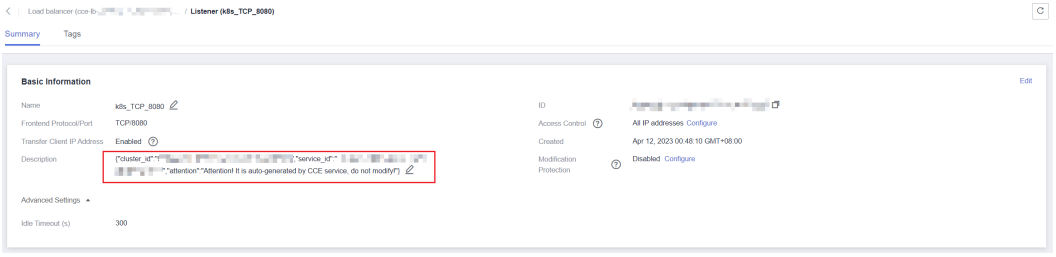
Deleting Residual ELB Resources

Step 1 Log in to the ELB console.

Step 2 Search for load balancers in the VPC by VPC ID used in the cluster.



Step 3 View the listener details of a load balancer. If the description contains the cluster ID and Service ID, the listener is created in the cluster.



Step 4 Delete the residual load balancer-related resources from the cluster based on the preceding information.

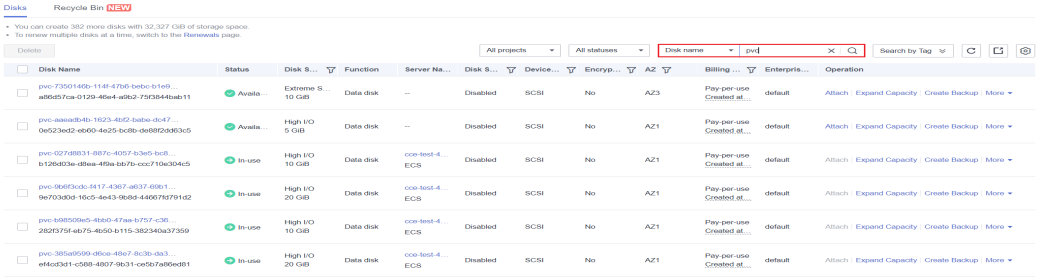
----End

Deleting Residual EVS Resources

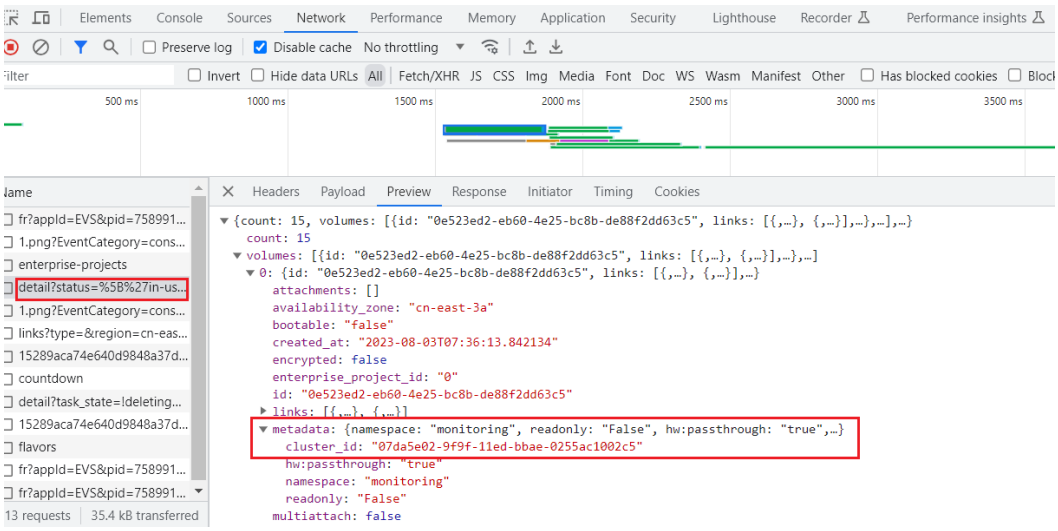
An EVS disk dynamically created using a PVC is named in the format of **pvc-*{UID}***. The **metadata** field in the API contains the cluster ID. You can use this cluster ID to obtain these EVS disks automatically created in the cluster and delete them as required.

Step 1 Go to the EVS console.

Step 2 Search for EVS disks by **pvc-*{UID}*** to get all automatically created EVS disks in the cluster.



Step 3 Press **F12** to open the developer tools. Check whether the **metadata** field in the **detail** API contains the cluster ID. If yes, the EVS disks are automatically created in this cluster.



Step 4 Delete the residual EVS disk-related resources from the cluster based on the preceding information.

NOTE

Deleted data cannot be restored. Exercise caution when performing this operation.

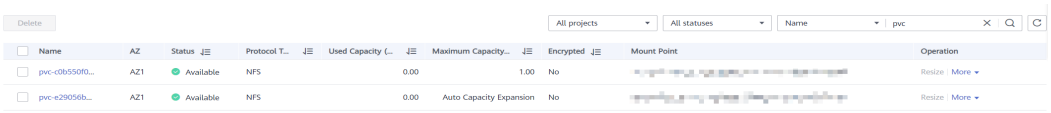
----End

Deleting Residual SFS Resources

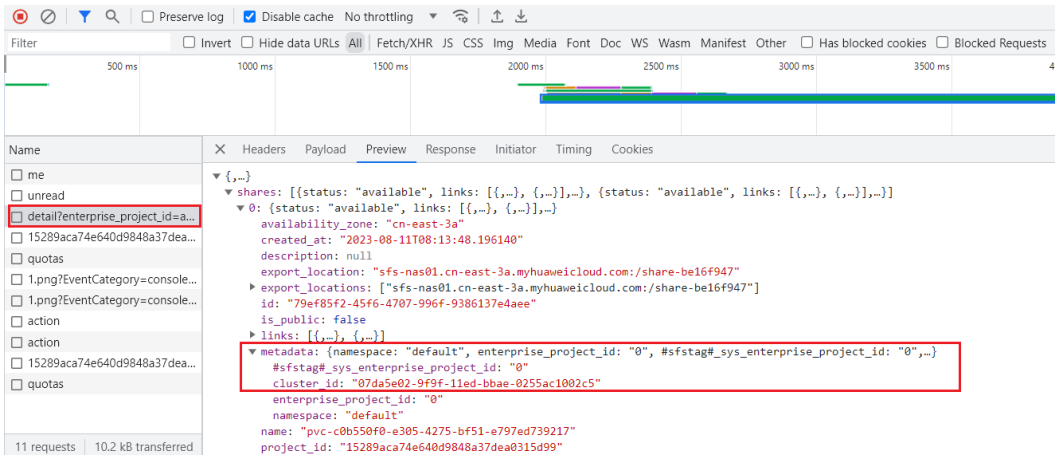
An SFS file system dynamically created using a PVC is named in the format of **pvc-*{UID}***. The **metadata** field in the API contains the cluster ID. You can use this cluster ID to obtain these SFS file systems automatically created in the cluster and delete them as required.

Step 1 Log in to the SFS console.

Step 2 Search for SFS file systems **pvc-*{UID}*** to get all automatically created SFS file systems in the cluster.



Step 3 Press **F12** to open the developer tools. Check whether the **metadata** field in the **detail** API contains the cluster ID. If yes, the SFS file systems are automatically created in the cluster.



Step 4 Delete the residual SFS file system-related resources from the cluster based on the preceding information.

NOTE

Deleted data cannot be restored. Exercise caution when performing this operation.

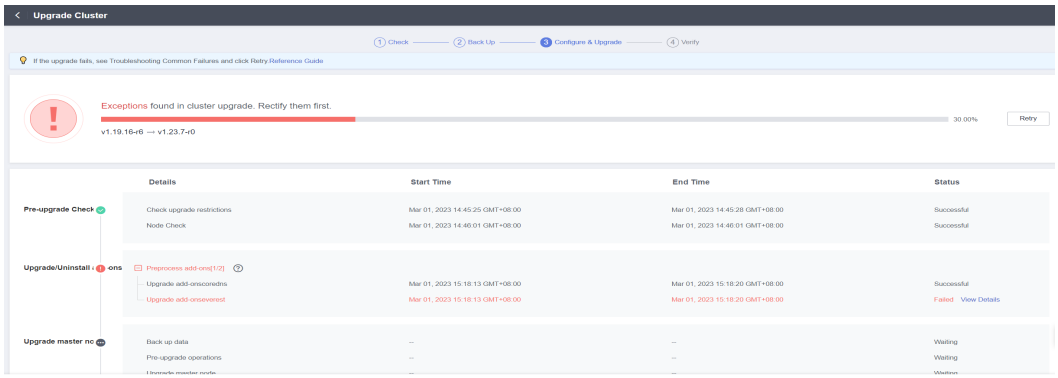
----End

3.4 Cluster Upgrade

3.4.1 What Do I Do If a Cluster Add-on Fails to be Upgraded During the CCE Cluster Upgrade?

Overview

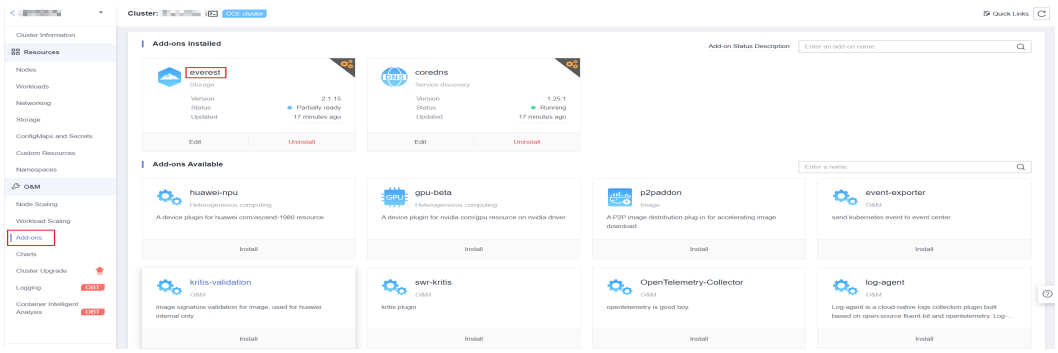
This section describes how to locate and rectify the fault if you fail to upgrade an add-on during the CCE cluster upgrade.



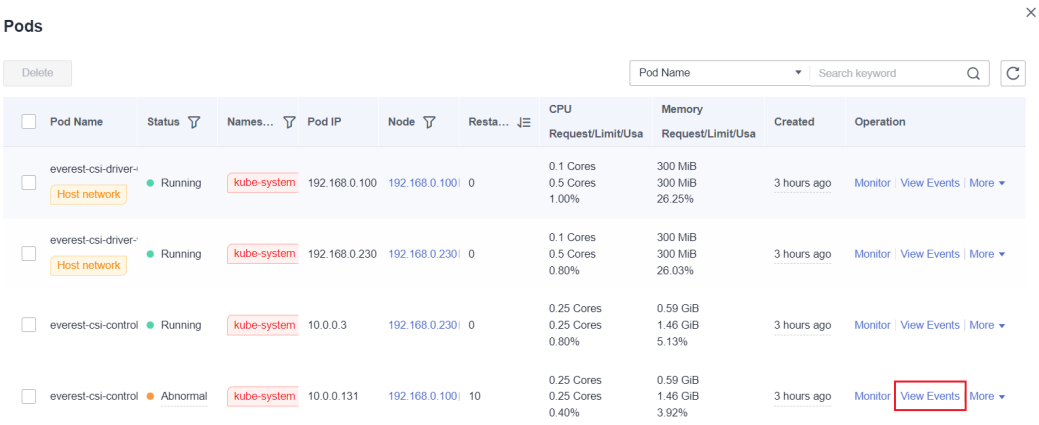
Procedure

Step 1 If the add-on fails to be upgraded, try again first. If the retry fails, perform the following operations to rectify the fault.

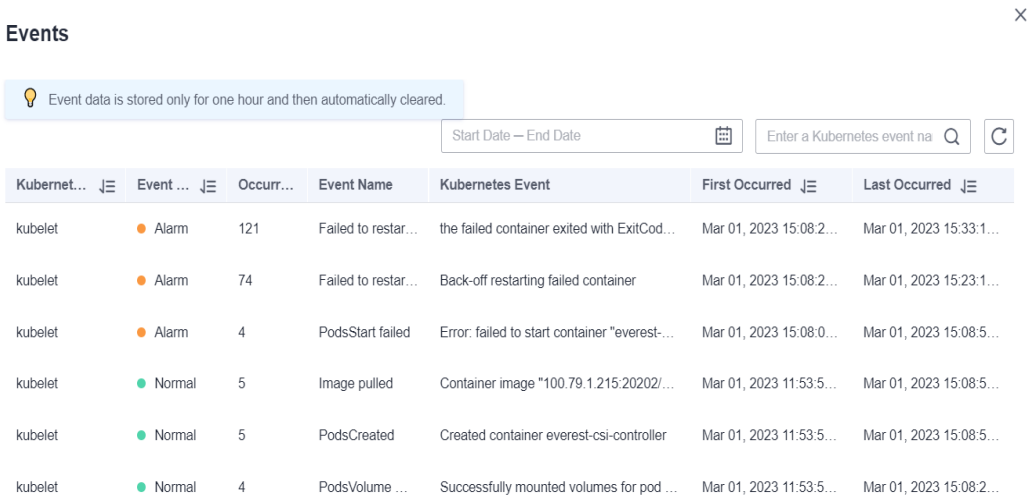
Step 2 If a failure message is displayed on the upgrade page, go to the **Add-ons** page to view the add-on status. For an abnormal add-on, click the add-on name to view details.



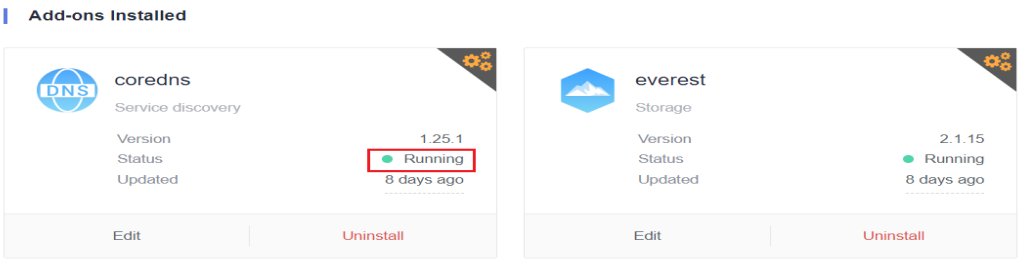
Step 3 On the pod details page, click **View Events** in the **Operation** column of the abnormal pod.



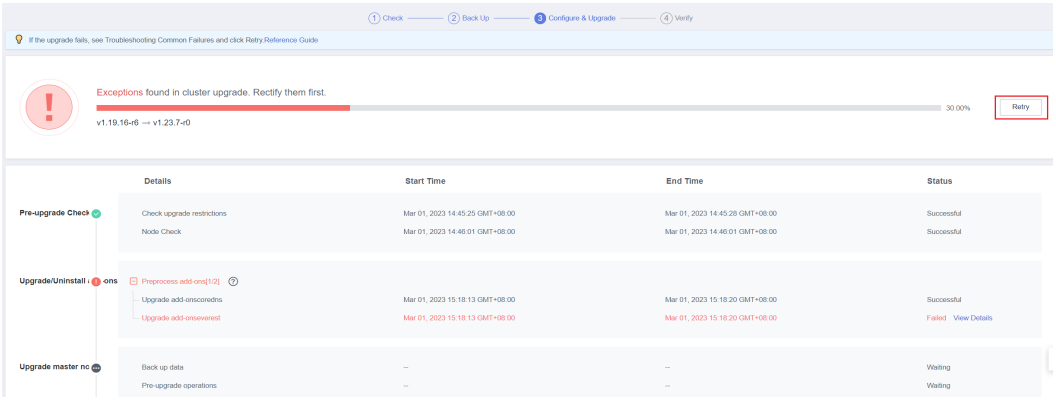
Step 4 Rectify the fault based on the exception information. For example, delete the pod that is not started or restart it.



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

3.4.2 What Should I Do If the LoadBalancer Ingress Configuration Is Inconsistent with the Load Balancer Configuration During a CCE Cluster Upgrade?

ELB Resources

In a CCE cluster, LoadBalancer ingresses are used to route external traffic to Services within the cluster. The parameters defined in an ingress are applied to configure a load balancer for effective traffic management and distribution. **Table 3-1** lists the mapping between load balancer configuration and ingress parameters.

Table 3-1 Mapping between load balancer configuration and LoadBalancer ingresses

Load Balancer Configuration		Description	Mapping in a LoadBalancer Ingress
Load balancer (**elb**)	-	Distribute incoming traffic across backend servers in one or more AZs.	**kubernetes.io/elb.id** in ingress annotations

Load Balancer Configuration		Description	Mapping in a LoadBalancer Ingress
Listener (**listener**)	-	Use the protocol and port you specify to check for requests from clients and route the requests to associated backend servers based on the routing rules you define.	**kubernetes.io/elb.port** in ingress annotations, which defaults to **80/443** if not defined
Forwarding policy (**l7policy**)	Forwarding policy	**Load balancer forwarding rules**: domain names or paths	Domain name and path in the forwarding rule: **spec.rules[].host** and **spec.rules[].http.paths[].path** in the ingress parameters, respectively
		Load balancer forwarding actions: forward to a backend server group	Backend server group: the backend service of the associated Service specified by **spec.rules[].http.paths[].backend.service** in the ingress
	Advanced forwarding policy	**Load balancer forwarding rules**: domain names, paths, HTTP request methods, HTTP headers, query strings, or CIDR blocks	The annotations related to advanced forwarding policies such as **kubernetes.io/elb.actions** and **kubernetes.io/elb.conditions** in the ingress annotations For details, see: • **Configuring Advanced Forwarding Actions for a LoadBalancer Ingress**
		Load balancer forwarding actions: forward to a backend server group, redirect to another listener, redirect to another URL, return a specific response body, rewrite, write header, remove header, and limit request	
Backend server group (**pool**)	-	A backend server group is a logical collection of one or more backend servers to receive massive requests concurrently. A backend server can be an ECS, supplementary network interface, or IP address.	**spec.rules[].http.paths[].backend** in the ingress parameters

Load Balancer Configuration		Description	Mapping in a LoadBalancer Ingress
Backend server (**member**)	-	Backend servers receive and process requests from the associated load balancer. For example, you can add an ECS as a backend server of a load balancer. The listener checks connection requests from clients using the configured protocol and port and forwards the requests to the backend servers in the backend server groups based on the load balancing algorithm you set.	Endpoint of the Service associated with the ingress

Troubleshooting Inconsistency Between Ingress and Load Balancer Configuration

During a pre-upgrade cluster check, many discrepancies between ingress and load balancer configuration may be identified. For each inconsistent item, CCE provides the expected configuration details. An example is as follows:

inconsistent: **ingress(default/nginx) need create listener(HTTP/8000) of elb(4a090bf3-9d11-45dc-8d04-f9cf010ba2ec)**, the wanted configuration is {"name":"k8s_HTTP_8000","protocol_port":8000,"protocol":"HTTP","loadbalancer_id":"4a090bf3-9d11-45dc-8d04-f9cf010ba2ec","sni_container_refs":null,"http2_enable":false}

inconsistent: **ingress(default/nginx) need create pool(k8s_default_nginx-80_HTTP-8000)**, the wanted configuration is {"name":"k8s_default_nginx-80_HTTP-8000","protocol":"HTTP","loadbalancer_id":"4a090bf3-9d11-45dc-8d04-f9cf010ba2ec","lb_algorithm":"ROUND_ROBIN","slow_start":{"enable":false,"duration":30}}

inconsistent: **need create member(address: 172.16.0.54) of pool(k8s_default_nginx-80_HTTP-8000)**, the wanted configuration is {"name":"2c2405d1abc17da48c70e9edc9a340fc","subnet_cidr_id":"77b9ad29-e30f-451b-92e7-949b83220b0f","address":"172.16.0.54","protocol_port":30838,"weight":6}

inconsistent: **ingress(default/nginx) need create l7policy(k8s_default_nginx_6666cd76) of listener(k8s_HTTP_8000/HTTP/8000)**, the wanted configuration is {"name":"k8s_default_nginx_6666cd76","listener_id":"k8s_HTTP_8000/HTTP/8000","action":"REDIRECT_TO_POOL","redirect_pool_id":"k8s_default_nginx-80_HTTP-8000"}

inconsistent: **ingress(default/nginx) need create rule of l7policy(k8s_default_nginx_6666cd76)**, the wanted configuration is {"type":"PATH","compare_type":"STARTS_WITH","value":"/"}

You can perform the following operations in sequence.

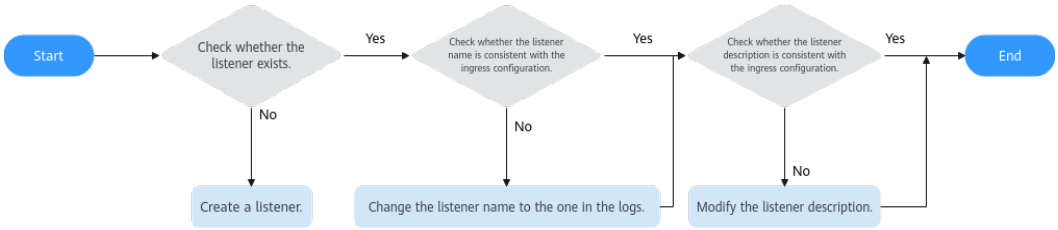
No.	Exception	Cause
1	Listener (**listener**)	• **Inconsistent Listeners: A Listener Needs to Be Created** • **Inconsistent Listeners: A Listener Needs to Be Updated**

No.	Exception	Cause
2	Forwarding policy (**l7policy**)	• **Inconsistent Forwarding Policies: A Forwarding Policy Needs to Be Created** • **Inconsistent Forwarding Policies: A Forwarding Policy Needs to Be Updated** • **Inconsistent Forwarding Policies: A Forwarding Policy Needs to Be Deleted**
3	Forwarding rule (**rule**)	**Inconsistent Forwarding Rules: A Forwarding Rule Needs to Be Created or Deleted**
4	Backend server group (**pool**)	**Inconsistent Backend Server Groups: A Backend Server Group Needs to Be Created**
5	Backend server (**member**)	• **Inconsistent Backend Servers: A Backend Server Needs to Be Created** • **Inconsistent Backend Servers: A Backend Server Needs to Be Updated** • **Inconsistent Backend Servers: A Backend Server Needs to Be Deleted**
6	Health check (**healthMonitor**)	• **Inconsistent Health Check Settings: Health Checks Need to Be Removed** • **Inconsistent Health Check Settings: Health Checks Need to Be Updated**

Inconsistent Listeners: A Listener Needs to Be Created

```
inconsistent: ingress(xxx) need create listener ...
```

When a listener's name or description is modified or a listener is deleted from the ELB console, this problem arises. In this case, you can use the port to check whether the listener is available on the ELB console and if its name and description match the expected configuration.



The listener description should resemble the following, where **cluster_id** represents the ID of the cluster:

```
{"attention":"Auto-generated by CCE service, do not
modify!","cluster_id":"5d4d44bd-0891-11f0-84d3-0255ac10003e"}
```

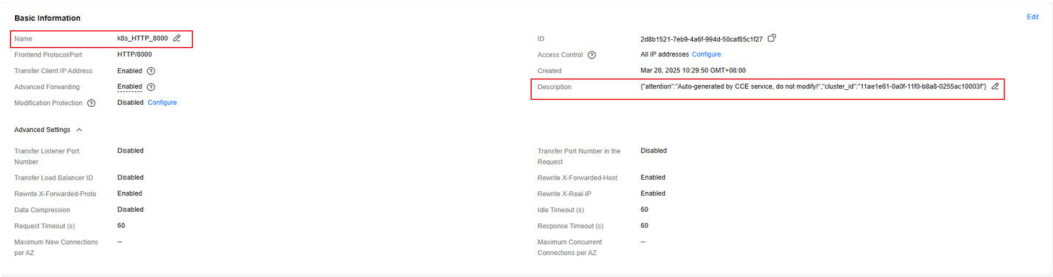
For example:

```
inconsistent: ingress(default/nginx) need create listener(HTTP/8000) of elb(4a090bf3-9d11-45dc-8d04-
f9cf010ba2ec), the wanted configuration is
```



```
{"name":"k8s_HTTP_8000","protocol_port":8000,"protocol":"HTTP","loadbalancer_id":"4a090bf3-9d11-45dc-8d04-f9cf010ba2ec","sni_container_refs":null,"http2_enable":false}
```

This indicates that an Nginx ingress in the **default** namespace needs a listener with port 8000 and protocol HTTP under the load balancer **4a090bf3-9d11-45dc-8d04-f9cf010ba2ec**. The listener should be named **k8s_HTTP_8000**, with **sni_container_refs** set to **null** and **http2_enable** set to **false**. **The description also needs to be specified.**



Inconsistent Listeners: A Listener Needs to Be Updated

```
inconsistent: ingress(xxx) need update listener ...
```

When the listener configuration is modified on the ELB console but is not synchronized with CCE, this problem arises. In this case, you can check the ELB console for the actual configuration and compare it with the expected configuration. If there is any inconsistency, you can update the listener manually or modify the ingress as needed.

For example:

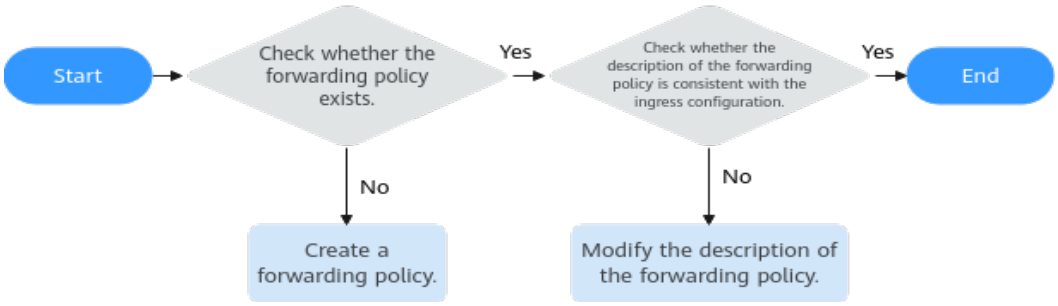
```
inconsistent: ingress(default/cce-elb-ingress) need update listener(5ca8a931-9d7a-465e-b503-08f88d528cd8), the wanted configuration is {"id":"5ca8a931-9d7a-465e-b503-08f88d528cd8","sni_container_refs":null}
```

The expected configuration of the listener is displayed after **the wanted configuration is**. In this example, the expected configuration is **sni_container_refs: null**. In this case, you can check whether the SNI function is enabled for the ELB listener but is missing in the ingress. If so, add the SNI configuration to the ingress. For details, see **Configuring SNI for a LoadBalancer Ingress**.

Inconsistent Forwarding Policies: A Forwarding Policy Needs to Be Created

```
inconsistent: ingress(xxx) need create l7policy(xxx) ...
```

When the description of a forwarding policy created by CCE is modified on the ELB console or a forwarding policy created by CCE is deleted on the ELB console, this problem arises. In this case, you can verify the policy's existence on the ELB console by its name and check whether the description matches the expected configuration.



The following shows an example for the description of a forwarding policy. The value of **cluster_id** is the ID of the current cluster, and the value of **ingress_id** is the UID of the current ingress.

```
{"attention":"Auto-generated by CCE service, do not modify!","cluster_id":"5d4d44bd-0891-11f0-84d3-0255ac10003e","ingress_id":"8a79a17b-f9ab-4431-95f6-0e1125093aac"}
```

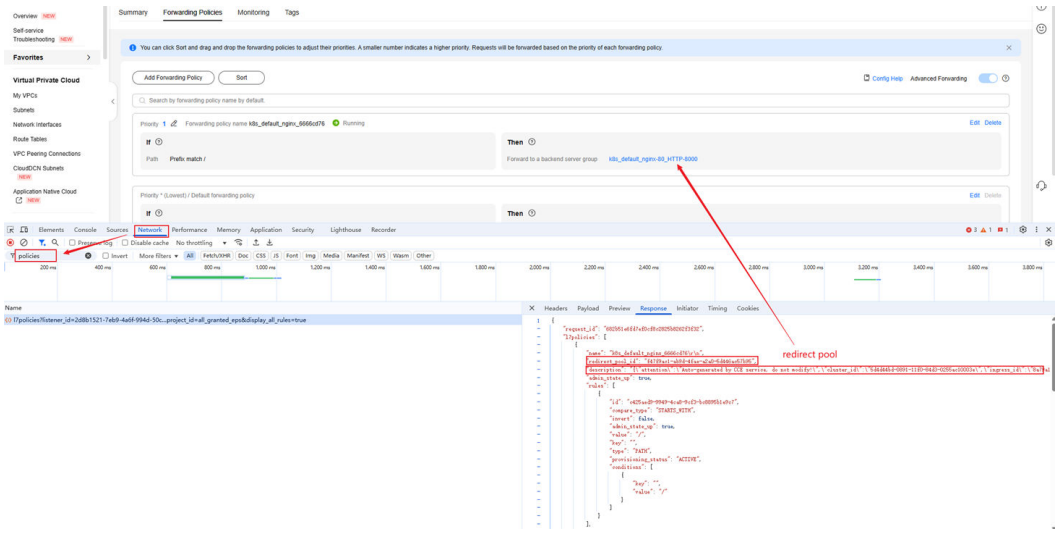
The ELB console does not provide a direct view of forwarding policy descriptions. To access this information, you must retrieve it from the data returned through the browser interface.

For example:

```
inconsistent: ingress(default/nginx) need create l7policy(k8s_default_nginx_6666cd76) of listener(0bdabd92-a85b-4f9b-96df-0fde706ea028), the wanted configuration is {"name":"k8s_default_nginx_6666cd76","listener_id":"0bdabd92-a85b-4f9b-96df-0fde706ea028","action":"REDIRECT_TO_POOL","redirect_pool_id":"k8s_default_nginx-80_HTTP-8000"}
```

This indicates that the Nginx ingress in the **default** namespace needs a forwarding policy named **k8s_default_nginx_6666cd76** under the listener **0bdabd92-a85b-4f9b-96df-0fde706ea028**, and traffic needs to be forwarded to the backend server group named **k8s_default_nginx-80_HTTP-8000**. **The description also needs to be specified.** You can call the **API for creating a forwarding policy**.

You can view the created forwarding policy in the returned information via the browser interface.



Inconsistent Forwarding Policies: A Forwarding Policy Needs to Be Updated

```
inconsistent: ingress(xxx) need update l7policy(xxx) of listener(xxx) ...
```

When the backend server group (**redirect_pool_id**) of the forwarding policy created by CCE is modified on the ELB console, this problem arises. For example:

```
inconsistent: ingress(default/nginx) need update l7policy(c4beb313-0a32-4add-9d0e-68b8f0f90e01) of listener(0bdabd92-a85b-4f9b-96df-0fde706ea028), the wanted configuration is {"id":"c4beb313-0a32-4add-9d0e-68b8f0f90e01","redirect_pool_id":"k8s_default_nginx-80_HTTP-8000","redirect_pools_config":[]}
```

This indicates that for the Nginx ingress in the **default** namespace, the **redirect_pool_id** in the forwarding policy **c4beb313-0a32-4add-9d0e-68b8f0f90e01** must be updated to the backend server group ID of **k8s_default_nginx-80_HTTP-8000** under the listener **0bdabd92-a85b-4f9b-96df-0fde706ea028**. Additionally, the **redirect_pools_config** should be deleted.

Inconsistent Forwarding Policies: A Forwarding Policy Needs to Be Deleted

```
inconsistent: ingress(xxx) need delete l7policy(id: xxx) ...
```

When a forwarding policy is created on the ELB console for the listener that is created by CCE, this problem arises. In this case, you can locate the forwarding policy based on its ID and then delete it.

For example:

```
inconsistent: need delete l7policy(id: f4eda0af-869a-47fb-a699-2f27f286b641)
```

You can call **the API for viewing details of a forwarding policy** to see which ELB listener the forwarding policy applies and then delete the forwarding policy. If you want to keep the forwarding policy, add the corresponding configuration to the ingress.

Inconsistent Forwarding Rules: A Forwarding Rule Needs to Be Created or Deleted

```
inconsistent: ingress(xxx) need create rule of l7policy(xxx) ...
```

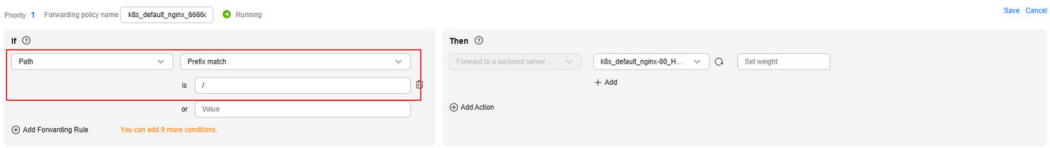
When the forwarding rule of a forwarding policy created by CCE is modified on the ELB console, this problem arises. Typically, changes to the rule type, matching mode, or path result in discrepancies. CCE handles forwarding rule updates by deleting and recreating them, which often leads to simultaneous creation and deletion issues.

For example:

```
inconsistent: ingress(default/nginx) need create rule of l7policy(k8s_default_nginx_6666cd76), the wanted configuration is {"type":"PATH","compare_type":"STARTS_WITH","value":"/"}

inconsistent: need delete rule(6cf87a3e-373e-4378-bf7a-16a709feee63) of l7policy(c4beb313-0a32-4add-9d0e-68b8f0f90e01)
```

This indicates that for the Nginx ingress in the **default** namespace, a forwarding rule needs to be created for the **k8s_default_nginx_6666cd76** policy. The rule's type should be **PATH**, the matching mode should be **STARTS_WITH** (prefix matching), and the value should be **/**. Additionally, you need to delete the forwarding rule **6cf87a3e-373e-4378-bf7a-16a709feee63** from the forwarding policy **c4beb313-0a32-4add-9d0e-68b8f0f90e01**.



Inconsistent Backend Server Groups: A Backend Server Group Needs to Be Created

inconsistent: **need create pool(xxx)**...

When the backend server group (**redirect_pool_id**) of the forwarding policy created by CCE is modified on the ELB console, this problem arises. Typically, the system displays the message "need update l7policy ...". In this case, you can check if the backend server group exists by its name or listener. If the group exists, update the backend server group of the forwarding policy.

For example:

```
inconsistent: ingress(default/nginx) need create pool(k8s_default_nginx-80_HTTP-8000), the wanted configuration is {"name":"k8s_default_nginx-80_HTTP-8000","protocol":"HTTP","loadbalancer_id":"4a090bf3-9d11-45dc-8d04-f9cf010ba2ec","lb_algorithm":"ROUND_ROBIN","slow_start":{"enable":false,"duration":30}}
```

If the group does not exist, create one based on the printed configuration items.

Inconsistent Backend Servers: A Backend Server Needs to Be Created

inconsistent: **need create member(address: xxx) of pool(xxx)** ...

When a backend server in a backend server group created by CCE is modified on the ELB console, this problem arises.

For example:

```
inconsistent: need create member(address: 172.16.0.54) of pool(dab7b0c1-623e-4499-8c70-7477f36fc2ef), the wanted configuration is {"name":"2c2405d1abc17da48c70e9edc9a340fc","subnet_cidr_id":"77b9ad29-e30f-451b-92e7-949b83220b0f","address":"172.16.0.54","protocol_port":30838,"weight":6}
```

This indicates that a backend server **2c2405d1abc17da48c70e9edc9a340fc** with the IP address **172.16.0.54**, port 30838, weight 6, and subnet **77b9ad29-e30f-451b-92e7-949b83220b0f** needs to be added to the backend server group **dab7b0c1-623e-4499-8c70-7477f36fc2ef**. You cannot enter a backend server name on the ELB console, but you can call the **API for adding a backend server**.

Inconsistent Backend Servers: A Backend Server Needs to Be Updated

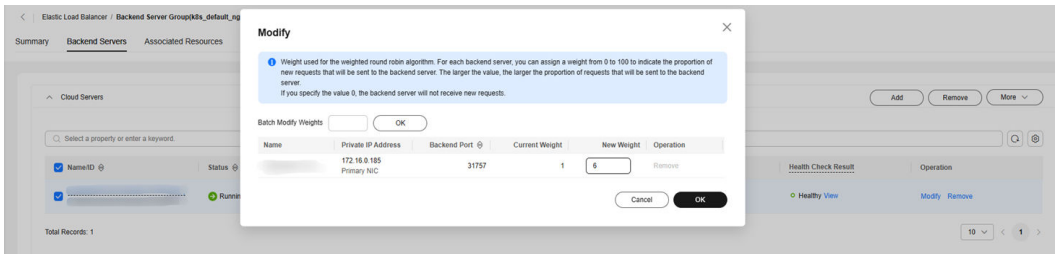
inconsistent: **need update member(address: xxx) of pool(xxx)** ...

When the weight of a backend server in the backend server group created by CCE is changed on the ELB console, this problem arises.

For example:

```
inconsistent: need update member(9a65b96c-891a-439e-a692-508c3e095095) of pool(dab7b0c1-623e-4499-8c70-7477f36fc2ef), the wanted configuration is {"id":"9a65b96c-891a-439e-a692-508c3e095095","name":"2c2405d1abc17da48c70e9edc9a340fc","weight":6}
```

This indicates that you need to change the weight of the backend server **9a65b96c-891a-439e-a692-508c3e095095** in the backend server group **dab7b0c1-623e-4499-8c70-7477f36fc2ef** to **6**.



Inconsistent Backend Servers: A Backend Server Needs to Be Deleted

inconsistent: **need delete member(address: xxx) of pool(xxx)** ...

When a backend server in a backend server group created by CCE is modified on the ELB console, this problem arises.

For example:

```
inconsistent: need delete members([d0891296-daf9-496b-9cac-ad1a1101b303]) of
pool(dab7b0c1-623e-4499-8c70-7477f36fc2ef)
```

This indicates that you need to delete the backend server **d0891296-daf9-496b-9cac-ad1a1101b303** from the backend server group **dab7b0c1-623e-4499-8c70-7477f36fc2ef**.

Inconsistent Health Check Settings: Health Checks Need to Be Removed

inconsistent: **need delete healthMonitor(xxx) of pool(xxx)** ...

When the health checks of a backend server group created by CCE are enabled on the ELB console but are not configured in the ingress, this problem arises.

For example:

```
inconsistent: need delete healthMonitor(ef7ad2c3-bdda-4727-8a1c-de37eeb48b55) of
pool(402dbff8-58b8-4a49-bb99-0dc5cf87c35b)
```

This indicates that the health check associated with the backend server group **402dbff8-58b8-4a49-bb99-0dc5cf87c35b** needs to be removed. You can call the **API for deleting a health check**. You can also enable the health check in the ingress.

Inconsistent Health Check Settings: Health Checks Need to Be Updated

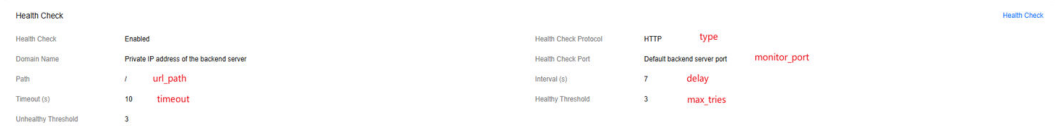
inconsistent: **need update healthMonitor(xxx) of pool(xxx)** ...

When the health checks of a backend server group created by CCE are modified on the ELB console, this problem arises.

For example:

```
inconsistent: need update healthMonitor(9d438b19-4342-42af-a876-77d49bbc9447) of
pool(402dbff8-58b8-4a49-bb99-0dc5cf87c35b), the wanted configuration is
{"delay":5,"max_retries":3,"timeout":10,"type":"HTTP","url_path":"/","monitor_port":null,"admin_state_up":tr
ue}
```

This indicates that the health check **9d438b19-4342-42af-a876-77d49bbc9447** must be updated to match the desired settings.



Common Issues

- Why is "need create" displayed when the listener, Layer 7 policy (**l7policy**), or pool exists?

  Solution

  Resources managed by CCE must have descriptions. You need to check whether the descriptions exist.

4 Node

4.1 How Can I Locate a Fault That Occurs with a Node?

Fault Locating

CCE allows you to locate a node fault using the CCE Node Problem Detector add-on (**Locating a Node Fault Using the CCE Node Problem Detector Add-on**). You can also refer to **Locating a Node Fault by Performing a Self-Check** to locate the fault.

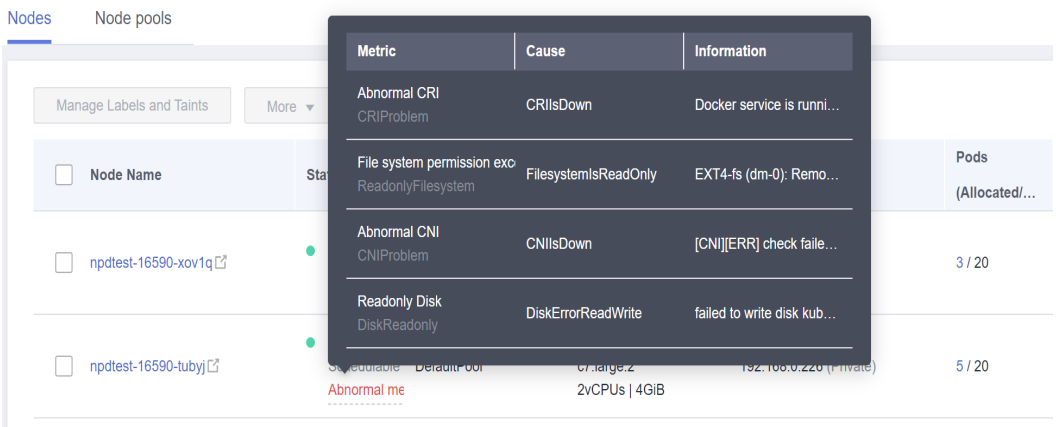
If the fault persists, submit a service ticket.

Locating a Node Fault Using the CCE Node Problem Detector Add-on

CCE provides an add-on called **CCE Node Problem Detector** for you to locate faults occurred with nodes. In 1.16.0 and later versions of this add-on, a large number of check items have been added. They allow you to detect the exceptions of resources and components on nodes and locate the faults.

It is strongly recommended that you install this add-on or upgrade it to **version 1.16.0 or later**.

With this add-on, if an exception occurs with a node, you can view the abnormal metrics on the console.



You can also view the events reported by the add-on in node events and locate the faults based on the events.

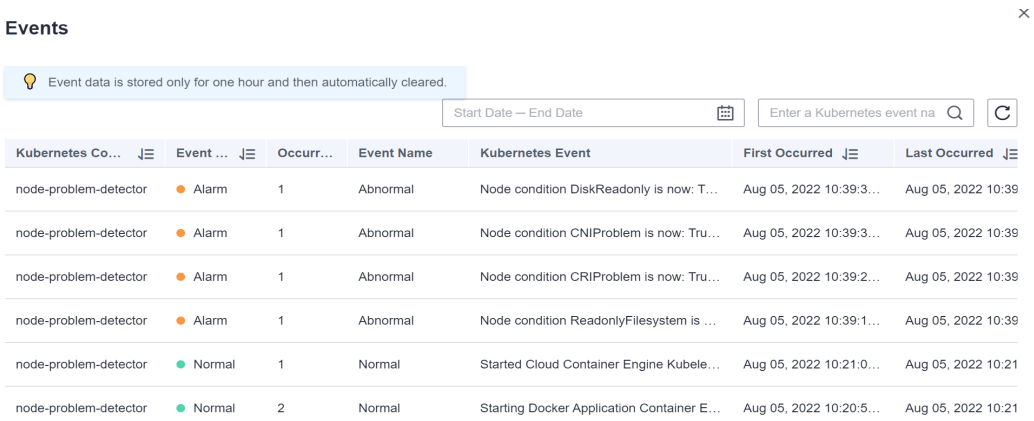


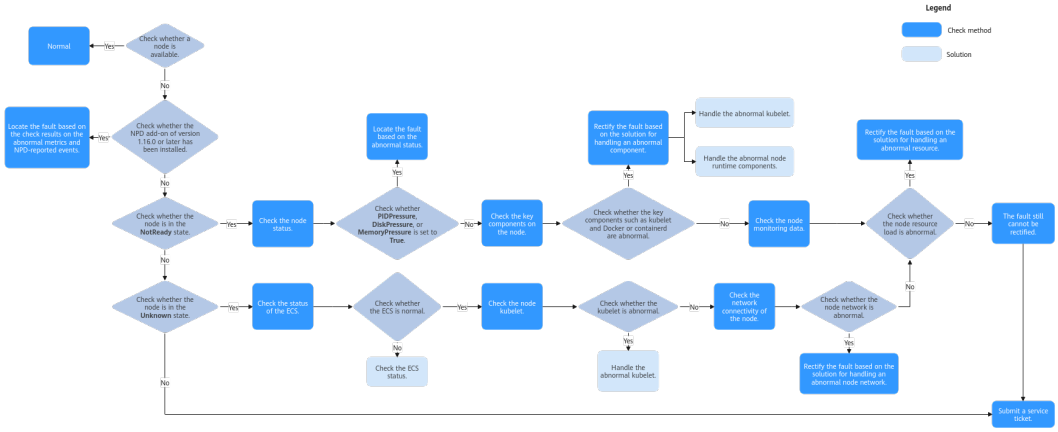
Table 4-1 Fault events

Event	Description
OOMKilling	Check whether OOM events occurred and are reported. Handling suggestions: **Check Item 1: Whether the Node Is Overloaded**
TaskHung	Check whether taskHung events occurred and are reported.
KernelOops	Check the kernel null pointer panic errors.
ConntrackFull	Check whether the conntrack table is full.
FrequentKubeletRestart	Check whether kubelet restarts frequently.
FrequentDockerRestart	Check whether Docker restarts frequently.
FrequentContainerdRestart	Check whether containerd restarts frequently.
CRIProblem	Check the CRI components.
KUBELETProblem	Check the kubelet.
NTPProblem	Check the NTP service.
PIDProblem	Check whether PIDs are sufficient.
FDProblem	Check whether file handles are sufficient.
MemoryProblem	Check whether the overall node memory is sufficient.
CNIProblem	Check the CNI components.
KUBEPROXYProblem	Check the kube-proxy.

Event	Description
ReadonlyFilesystem	Check whether the **Remount root filesystem read-only** error occurred in the system kernel. Possible cause: The data disks were detached from the ECS by mistake, or the VDB disk of the node has been deleted. Handling suggestions: • **Check Item 6: Whether the Disk Is Abnormal** • **Check Item 9: Whether the vdb Disk on the Node Has Been Deleted**
DiskReadonly	Check whether the system disk, Docker disk, and kubelet disk are read-only. Possible cause: The data disks were detached from the ECS by mistake, or the VDB disk of the node has been deleted. Handling suggestions: • **Check Item 6: Whether the Disk Is Abnormal** • **Check Item 9: Whether the vdb Disk on the Node Has Been Deleted**
DiskProblem	Check the disk usage and whether the key logical disk is properly attached to the node. Check the usage of the system disk, Docker disk, and kubelet disk, and check whether the Docker and kubelet disks are properly attached to the ECS.
PIDPressure	Check whether PIDs are sufficient. Handling suggestions: If there are not enough PIDs available, adjust the upper limit of PIDs as needed. For details, see **Changing Process ID Limits (kernel.pid_max)**.
MemoryPressure	Check whether the allocable memory for the containers is sufficient.
DiskPressure	Check the usage of kubelet and Docker disks and inodes. Handling suggestions: Expand the capacity of the data disks. For details, see **Expanding the Storage Space**.

Locating a Node Fault by Performing a Self-Check

Figure 4-1 Performing a self-check



1. Log in to the CCE console.
2. Click the cluster name to access the cluster console. Choose **Nodes** in the navigation pane. In the right pane, click the **Nodes** tab.
3. Locate the row containing the target node, choose **More** > **View YAML** in the **Operation** column, and check the **Status** field of the node.

   The node is in the NotReady state.

   - Check the node status and verify whether the value of **PIDPressure**, **DiskPressure**, or **MemoryPressure** becomes **True**. If any of them becomes **True**, you can find the appropriate solution based on the exception keyword.
   - Check the key components on the node and the logs of these components. The key components on a node include a kubelet and the node runtime (Docker or containerd). For details, see **Checking Key Components of the Node**.
     - Check the kubelet.

       Check whether the kubelet and its logs are normal. If there is an exception, see **Abnormal kubelet**.
     - Check the runtime (Docker or containerd).
       - Check the runtime of the node. If you are not sure whether the runtime is Docker or containerd, log in to the CCE console and view the runtime of the node.
       - If there is an exception, see **Abnormal Runtime**.
     - Check the NTP.
       - Check whether the NTP, its logs, and the configurations are normal.
       - If there is an exception, see **Abnormal NTP**.
   - Check the node monitoring data and see whether the CPU, memory, and network resources of the node are normal. If there is an exception, rectify the fault by referring to **Memory Pressure**.

The node is in the Unknown state.

- Log in to the ECS console and check whether the node is present in the ECS list.
- Check whether the node is running properly.
- Check the key components on the node and the logs of these components. The key components on a node include a kubelet and the node runtime (Docker or containerd). For details, see **Checking Key Components of the Node**.
  - Check the kubelet.
    - Check whether the kubelet and its logs are normal. If there is an exception, see **Abnormal kubelet**.
- Check the network connectivity of the node.

Common Problems and Troubleshooting Methods

Checking a Node

1. Log in to the CCE console.
2. Click the name of the target cluster to access the cluster console.
3. In the navigation pane, choose **Nodes**. In the right pane, click the **Nodes** tab, locate the row containing the unavailable node, and view its status. (If NPD 1.6.10 or a later version is installed in the cluster, you will see a message indicating that the metrics for the unavailable node are abnormal. In this case, you can move the cursor to the upper part to view the specific problem. If the add-on is not installed, you can rectify the fault by referring to the check items.)

Checking the Node Monitoring Data

1. Log in to the CCE console.
2. Click the name of the target cluster to access the cluster console.
3. In the navigation pane, choose **Nodes**. In the right pane, click the **Nodes** tab, locate the row containing the abnormal node, and click **Monitor** in the **Operation** column.

   On the top of the displayed page, click **More Monitoring Data** to go to the AOM console and view historical monitoring records. If the CPU or memory usage of the node is too high, it can lead to high network latency or trigger system OOM, causing the node to be marked as unavailable.

Checking the Node Events

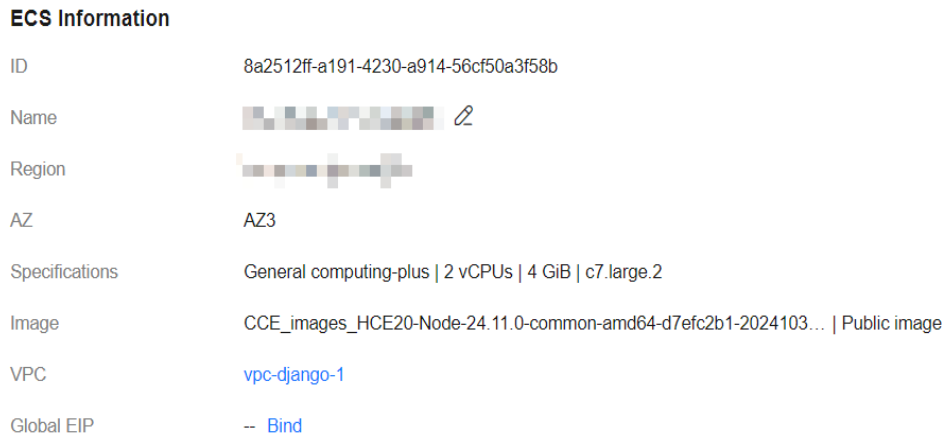
1. Log in to the CCE console.
2. Click the name of the target cluster to access the cluster console.
3. In the navigation pane, choose **Nodes**. In the right pane, click the **Nodes** tab, locate the row containing the abnormal node, click **View Events** in the **Operation** column, and check whether any abnormal event is reported. (The NPD add-on must be installed.)

Verifying Whether the ECS Has Been Deleted or Is Faulty

1. Log in to the CCE console, click the name of the target cluster to access the cluster console, and view the name of the unavailable node.
2. Log in to the ECS console, search for the node, and check the ECS status.
   - If the ECS has been deleted, go back to the CCE console, delete the node from the node list, and create another one.
   - If the ECS is stopped or frozen, restore it first. It takes about 3 minutes to restore the node.
   - If the ECS is faulty, restart it to rectify the fault.
   - If the ECS is available, rectify the fault by referring to **Checking Key Components of the Node**.

Verifying Whether the ECS Can Be Logged In

1. Log in to the ECS console.
2. Check whether the node name displayed on the ECS console 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 or key cannot be used to log in to the node, Cloud-Init problems occurred when the ECS was created. In this case, you can restart the node and then **submit a service ticket to the ECS personnel** to locate the root cause.

Checking the Node Security Group

- The node security group was changed.
  a. Log in to the VPC console. In the navigation pane, choose **Access Control** > **Security Groups** and find the master node security group of the cluster.
  b. Search for the name of the security group that contains the cluster name and **-cce-control-**. The name of a master node security group is in the format of *cluster-name*-**cce-control-***random-ID*.
  c. Check whether the security group rules have been changed. For details about security groups, see **How Can I Configure a Security Group Rule for a Cluster?**

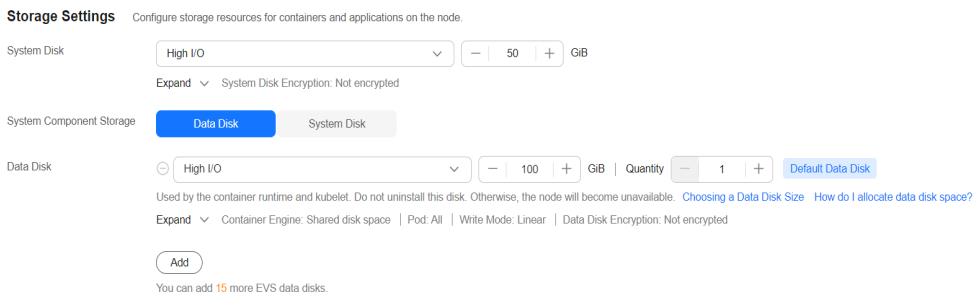
- The node security group rules must contain a policy that allows the communication between the master nodes and the worker nodes.
  - Check whether such a security group policy is present.
  - When adding a node to the cluster, add the security group rules listed in **Table 4-2** to the *cluster-name*-**cce-control-***random-ID* security group to ensure the availability of the added node. This is necessary if a secondary CIDR block is added to the VPC of the node subnet and the subnet is in the secondary CIDR block. However, if a secondary CIDR block has already been added to the VPC during cluster creation, this step is not required.
  - For details about security groups, see **How Can I Configure a Security Group Rule for a Cluster?**

Table 4-2 Security group rules to be added

Protocol and Port	Type	Source IP Address
TCP port 8445	IPv4	The new secondary CIDR block where the subnet is in
TCP port 9443	IPv4	The new secondary CIDR block where the subnet is in
TCP port 5444	IPv4	The new secondary CIDR block where the subnet is in

Checking the Disks Attached to the Node

- By default, a 100-GiB data disk is attached to a node for runtime purposes. You have the option to attach additional data disks to the node if needed. If the data disk is removed or damaged, the runtime will be disrupted and the node will become unavailable.



- You need to check whether the data disks of the node are detached from it. If they are, you are advised to create a node and delete the unavailable node. (To minimize risks, you are not advised to perform operations on the CCE nodes through the ECS console.)

Checking Key Components of the Node

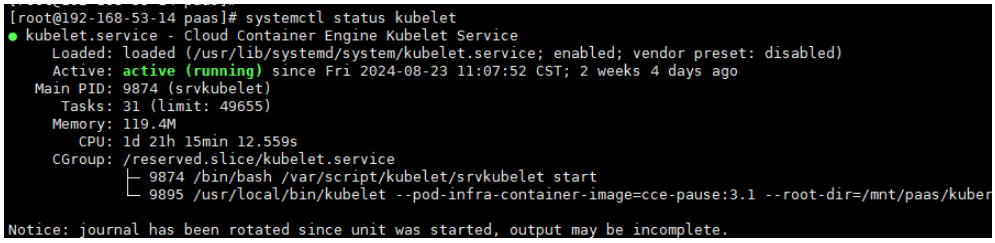
kubelet

- Check the kubelet.

  Log in to the target node, run the following command on it, and check the kubelet:

  ```
  systemctl status kubelet
  ```

  The expected output is shown in the figure below.

  
  ```
  [root@192-168-53-14 paas]# 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 Fri 2024-08-23 11:07:52 CST; 2 weeks 4 days ago
     Main PID: 9874 (srvkubelet)
        Tasks: 31 (limit: 49655)
       Memory: 119.4M
          CPU: 1d 21h 15min 12.559s
       CGroup: /reserved.slice/kubelet.service
               ├─ 9874 /bin/bash /var/script/kubelet/srvkubelet start
               └─ 9895 /usr/local/bin/kubelet --pod-infra-container-image=cce-pause:3.1 --root-dir=/mnt/paas/kuber

  Notice: journal has been rotated since unit was started, output may be incomplete.
  ```

- View the kubelet logs.

  Log in to the target node, run the following command on it, and check the kubelet:

  ```
  journalctl -u kubelet
  ```

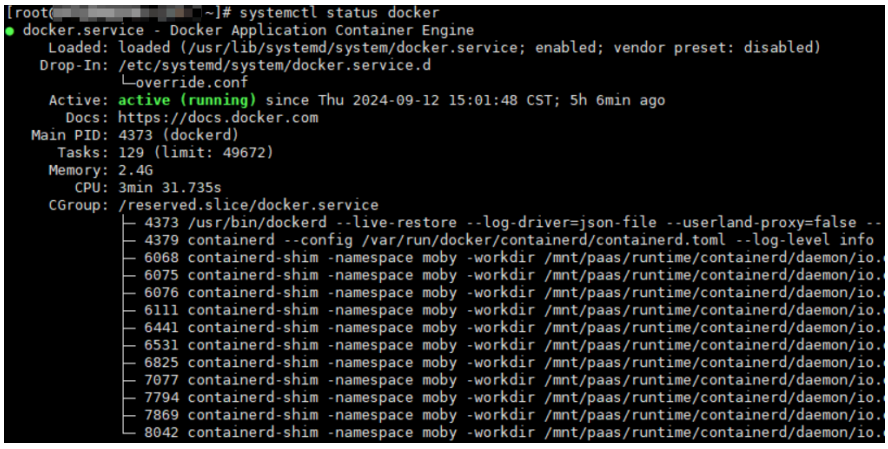
Runtime

- Docker
  - Check the Docker runtime.

    Log in to the target node, run the following command on it, and check the Docker process:

    ```
    systemctl status docker
    ```

    The expected output is shown in the figure below.

    
    ```
    [root(                    ~]# systemctl status docker
    ● docker.service - Docker Application Container Engine
         Loaded: loaded (/usr/lib/systemd/system/docker.service; enabled; vendor preset: disabled)
        Drop-In: /etc/systemd/system/docker.service.d
                 └─override.conf
         Active: active (running) since Thu 2024-09-12 15:01:48 CST; 5h 6min ago
           Docs: https://docs.docker.com
       Main PID: 4373 (dockerd)
          Tasks: 129 (limit: 49672)
         Memory: 2.4G
            CPU: 3min 31.735s
         CGroup: /reserved.slice/docker.service
                 ├─ 4373 /usr/bin/dockerd --live-restore --log-driver=json-file --userland-proxy=false --
                 ├─ 4379 containerd --config /var/run/docker/containerd/containerd.toml --log-level info
                 ├─ 6068 containerd-shim -namespace moby -workdir /mnt/paas/runtime/containerd/daemon/io.
                 ├─ 6075 containerd-shim -namespace moby -workdir /mnt/paas/runtime/containerd/daemon/io.
                 ├─ 6076 containerd-shim -namespace moby -workdir /mnt/paas/runtime/containerd/daemon/io.
                 ├─ 6111 containerd-shim -namespace moby -workdir /mnt/paas/runtime/containerd/daemon/io.
                 ├─ 6441 containerd-shim -namespace moby -workdir /mnt/paas/runtime/containerd/daemon/io.
                 ├─ 6531 containerd-shim -namespace moby -workdir /mnt/paas/runtime/containerd/daemon/io.
                 ├─ 6825 containerd-shim -namespace moby -workdir /mnt/paas/runtime/containerd/daemon/io.
                 ├─ 7077 containerd-shim -namespace moby -workdir /mnt/paas/runtime/containerd/daemon/io.
                 ├─ 7794 containerd-shim -namespace moby -workdir /mnt/paas/runtime/containerd/daemon/io.
                 ├─ 7869 containerd-shim -namespace moby -workdir /mnt/paas/runtime/containerd/daemon/io.
                 └─ 8042 containerd-shim -namespace moby -workdir /mnt/paas/runtime/containerd/daemon/io.
    ```

  - View the Docker logs.

    Log in to the target node, run the following command on it, and check the Docker logs:

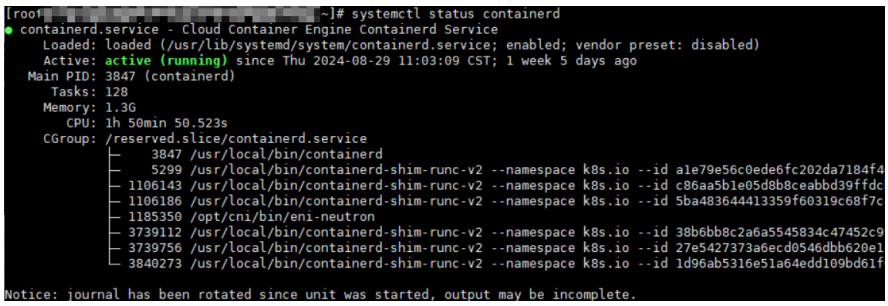
    ```
    journalctl -u docker
    ```

- containerd
  - Check the containerd runtime.

    Log in to the target node, run the following command on it, and check the containerd process:


```
systemctl status containerd
```

The expected output is shown in the figure below.



  - View the containerd logs.

    Log in to the target node, run the following command on it, and check the containerd logs:

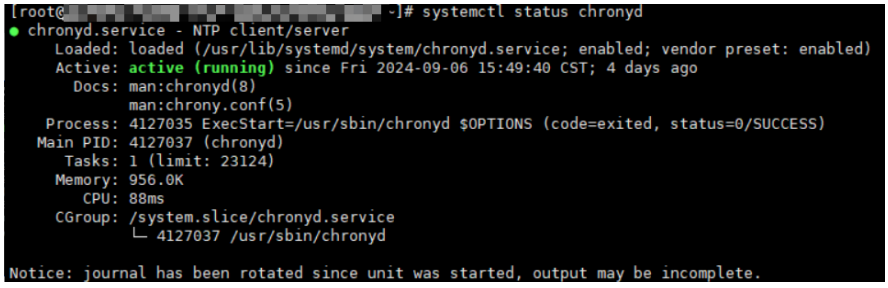
    ```
    journalctl -u containerd
    ```

- NTP
  - Check whether the NTP is normal.

    Log in to the target node, run the following command on it, and check the chronyd process:

    ```
    systemctl status chronyd
    ```

    The expected output is shown in the figure below.



  - View the NTP logs.

    Log in to the target node, run the following command on it, and check the NTP logs:

    ```
    journalctl -u chronyd
    ```

Verifying Whether the Node DNS Address Is Properly Configured

- Log in to the node and check whether any domain name resolution failure is recorded in **/var/log/cloud-init-output.log**:

  ```
  cat /var/log/cloud-init-output.log | grep resolv
  ```

  If information similar to the following is displayed, the domain name cannot be resolved:

  ```
  Could not resolve host: xxx ; Unknown error
  ```

- Ping the domain name that cannot be resolved on the node:

  ```
  ping xxx
  ```

  If the domain name cannot be pinged, the DNS cannot resolve the IP address. You need to verify if the DNS address in the **/etc/resolv.conf** file matches the configuration on the VPC subnet. In most cases, the DNS address in the file is improperly configured, leading to the inability to resolve domain names.

Verifying Whether the Node Is a Yearly/Monthly One and Is Being Unsubscribed

Once a node is unsubscribed, it will take some time to process the order, rendering the node unavailable during this period. Typically, the node is expected to be automatically cleared within 5 to 10 minutes.

Common Issues and Solutions

PID Pressure

Possible Cause

The pods on the node are using up a large number of PIDs, causing a shortage of available PIDs on the node. By default, CCE reserves 10% of the available PIDs for pods.

Symptom

If the number of available PIDs on a node is lower than the specified value of **pid.available**, the **PIDPressure** of the node will be set to **True**, resulting in the eviction of pods running on that node. For details about node eviction, see **Node-pressure Eviction**.

Solution

1. Check the maximum number of PIDs on the node and the processes that use the most PIDs:
   ```
   sysctl kernel.pid_max # Check the maximum number of PIDs.
   ps -eLf|awk '{print $2}' | sort -rn| head -n 1 #: Check the processes that use the most PIDs on the node.
   ```
2. Check the top five processes that use the most PIDs:
   ```
   ps -elT | awk '{print $4}' | sort | uniq -c | sort -k1 -g | tail -5
   ```
   The following shows an example of the expected output:
   ```
   17 1211619
   18 3739112
   18 5299
   24 964
   25 3739756
   ```
   The first column shows the number of PIDs used by each process, while the second column displays the current process IDs. You can locate the process and associated pods with the given process ID, analyze the reason for excessive PID usage, and optimize the relevant code accordingly.
3. Reduce the load of the node.
4. To restart the node, go to the ECS console and restart it. (**Be cautious when restarting the node because it may cause interruptions to your services.**)

Memory Pressure

Possible Cause

The pods on the node are using up a large amount of memory, causing a shortage of available memory on the node. By default, the available memory of a CCE node is 100 MiB.

Symptom

- If the number of available memory resources on a node is lower than the specified value of **memory.available**, the **MemoryPressure** of the node will be set to **True**, resulting in the eviction of pods running on that node. For details about node eviction, see **Node-pressure Eviction**.
- If the memory of a node is not enough, the following message will show:
  - The value of **MemoryPressure** becomes **True**.
  - When some pods on the node are evicted:
    - You can see "The node was low on resource: memory" in the events of the evicted pods.
    - You can see "attempting to reclaim memory" in the node events.
  - The system OOM may behave abnormally. If such an error occurs, you can see "System OOM" in the node events.

Solution

- Check the node memory usage through the node monitoring data, identify the time when the exception occurs, and verify if there is any memory leak in the processes on the node. For details, see **Checking the Node Monitoring Data**.
- Reduce the load of the node.
- To restart the node, go to the ECS console and restart it. (**Be cautious when restarting the node because it may cause interruptions to your services.**)

Disk Pressure

Possible Cause

The root file system, image file system, or container file system on the node is consuming excessive disk space and inodes, surpassing the eviction threshold. This leads to the **nodefs.available**, **nodefs.inodesFree**, **imagefs.available**, or **imagefs.inodesFree** metric met the eviction threshold, causing disk pressure. The table below shows the default values for these parameters.

Parameter	Description	Default Value
nodefs.available	Percentage of the available capacity in the file system used by kubelet.	10%
nodefs.inodesFree	Percentage of available inodes in the file system used by kubelet.	5%
imagefs.available	Percentage of the available capacity in the file system used by container runtimes to store resources such as images.	10%

Parameter	Description	Default Value
imagefs.inodesFree	Percentage of available inodes in the file system used by container runtimes to store resources such as images.	5%

Symptom

- If the available disk space on the node is less than the value of **imagefs.available**, the value of **DiskPressure** of the node becomes **True**.
- If the available disk space is less than the value of **nodefs.available**, all pods on the node will be evicted. For details about node eviction, see **Node-pressure Eviction**.
- If the disk space on the node is insufficient, the following message will show:
  - The value of **DiskPressure** becomes **True**.
  - If the disk space remains insufficient to meet the healthy threshold (defaulted at 80%) even after the image reclaim policy is triggered, you can see "failed to garbage collect required amount of images" in the node events.
  - When some pods on the node are evicted:
    - You can see "The node was low on resource: [DiskPressure]" in the events of the evicted pods.
    - You can see "attempting to reclaim ephemeral-storage" or "attempting to reclaim nodefs" in the node events.

Solution

- View the node disk usage through the node monitoring data, identify the time when the exception occurs, and check if processes on the node are consuming excessive disk space. For details, see **Checking the Node Monitoring Data**.
- If a large number of files are not deleted from the node disks, delete these files.
- Restrict the ephemeral-storage configurations of the pods based on service requirements.
- Use cloud storage services instead of hostPath volumes.
- Expand the capacity of the node disks. For details, see **Expanding the Storage Space**.
- Reduce the load of the node.

Abnormal kubelet

Possible Cause

The kubelet process is not functioning properly or the kubelet configuration is improper. Typically, CCE has set up health checks for kubelet as a default

configuration. There is a greater chance of startup failure if the configuration is incorrect.

Symptom

kubelet is inactive.

Solution

1. Log in to the abnormal node and restart kubelet: (Restarting kubelet does not affect the running containers.)
   ```
   systemctl restart kubelet
   ```
2. Check whether the kubelet status becomes normal:
   ```
   systemctl status kubelet
   ```
3. If the kubelet status is still abnormal after the restart, log in to the node and view the kubelet logs:
   ```
   journalctl -u kubelet
   ```
   - If there are error messages in the logs, find the cause by looking for specific keywords associated with the error.
   - If there is an issue with the kubelet configuration, find the node pool that the node belongs to, click **Manage** in the **Operation** column, and make changes to the kubelet configuration.

Abnormal Runtime

Possible Cause

The Docker or containerd configuration or process is not functioning properly.

Symptom

- Docker
  - Docker is inactive.
  - Docker is active and running, but the node is experiencing issues and not functioning properly, resulting in abnormal behavior. In this case, the **docker ps** or **docker exec** command fails to be executed.
  - The value of **RuntimeOffline** of the node becomes **True**.
- containerd
  - containerd is inactive.
  - The value of **RuntimeOffline** of the node becomes **True**.

Solution

1. Log in to the abnormal node.
2. Restart the runtime:
   ```
   # Docker
   systemctl restart docker
   # containerd
   systemctl restart containerd
   ```
3. After the command is executed, check whether the running status is normal.
   ```
   # Docker
   systemctl status docker
   # containerd
   systemctl status containerd
   ```


4. If the component status is still abnormal after the restart, check the component logs:

```
# Docker
journalctl -u docker
# containerd
journalctl -u containerd
```

Abnormal NTP

Possible Cause

The NTP process is abnormal.

Symptom

- chronyd is inactive.
- The value of **NTPProblem** becomes **True**.

Solution

1. Log in to the abnormal node and restart chronyd:

```
systemctl restart chronyd
```

2. After the restart, check whether the chronyd status becomes normal:

```
systemctl status chronyd
```

3. If the chronyd status is still abnormal after the restart, log in to the node and view the chronyd logs:

```
journalctl -u chronyd
```

Abnormal Node Restart

Possible Cause

The node is experiencing abnormal load.

Symptom

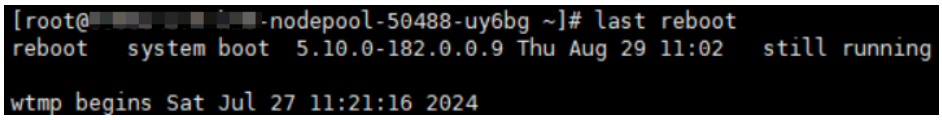
During the restart, the node is in the **NotReady** state.

Solution

1. Check the time when the node was restarted:

```
last reboot
```

The expected output is shown in the figure below.



2. View the node monitoring data and locate the abnormal resource based on the restart time of the node. For details, see **Checking the Node Monitoring Data**.
3. Check the kernel logs and locate the fault based on the restart time.

Abnormal Node Network

Possible Cause

The node is experiencing abnormal running status, incorrect security group configuration, or excessive network load.

Symptom

- The node cannot be logged in to.
- The node is in the **Unknown** state.

Solution

- If you cannot log in to the node, take the following steps to locate the fault:
  - Check whether the node is running on the ECS console.
  - Check whether the fault is caused by the execution failure of Cloud-Init of the ECS. For details, see **Verifying Whether the ECS Can Be Logged In**.
  - Check the security group configuration of the node. For details, see **Checking the Node Security Group**.
- If the network load of the node is too high, perform the following operations:
  - View the node networking through the node monitoring data and check whether the pods on the node are consuming excessive network bandwidth.
  - Use network policies to control network traffic of the pods on the node. For details, see **Configuring Network Policies to Restrict Pod Access**.

Abnormal PLEG

Possible Cause

The pod lifecycle event generator (PLEG) records different events in the lifecycle of a pod, such as the pod startup and termination. The error "PLEG is not healthy" is usually due to abnormal runtime processes on the node or issues with the systemd version on the node.

Symptom

- The node is in the **NotReady** state.
- You can see the following information in the kubelet logs:
  ```
  skipping pod synchronization - PLEG is not healthy: pleg was last seen active 3m17.028393648s ago; threshold is 3m0s.
  ```

Solution

- Restart Docker or containerd and kubelet in sequence and then check whether the node is restored.
- If the node is not restored after the restart of the key components, restart the node. (**Be cautious when restarting the node because it may cause interruptions to your services.**)

Node Overloaded

Possible Cause

The node resources are not enough for pod scheduling.

Symptom

If there are not enough scheduling resources available on the node, pod scheduling will fail and the following error information will be displayed: (Only errors related to common resources are listed.)

- Insufficient CPUs in a cluster: 0/2 nodes are available: 2 Insufficient cpu
- Insufficient memory in a cluster: 0/2 nodes are available: 2 Insufficient memory
- Insufficient temporary storage space in a cluster: 0/2 nodes are available: 2 Insufficient ephemeral-storage

The scheduler determines that node resources are insufficient using the following calculation methods:

- Whether the CPUs of a node are insufficient: Total CPUs requested by a pod > (Total allocatable CPUs on the node - Total CPUs that have been allocated to the pods on the node)
- Whether the memory of a node is insufficient: Total memory requested by a > (Total allocatable memory on the node - Total memory that has been allocated to the pods on the node)
- Whether the temporary storage space of a node is insufficient: Temporary storage space requested by a pod > (Total allocatable temporary storage space on the node - Total temporary storage space that has been allocated to the pods on the node)

If the total resources requested by the pod exceed the allocatable resources on the node (after subtracting the allocated resources to the pods on the node), the pod will not be scheduled on that node.

Check the resource allocation details on the node:

```
kubectl describe node $nodeName
```

Pay attention to the resource allocation in the command output:

```
Allocatable:
  cpu:                1930m
  ephemeral-storage:  94576560382
  hugepages-1Gi:      0
  hugepages-2Mi:      0
  localssd:           0
  localvolume:        0
  memory:             2511096Ki
pods:                 20
Allocated resources:
  (Total limits may be over 100 percent, i.e., overcommitted.)
  Resource           Requests      Limits
  --------           --------      ------
  cpu                1255m (65%)   4600m (238%)
  memory             1945Mi (79%)  3876Mi (158%)
  ephemeral-storage  0 (0%)        0 (0%)
  hugepages-1Gi      0 (0%)        0 (0%)
  hugepages-2Mi      0 (0%)        0 (0%)
  localssd           0             0
  localvolume        0             0
```

Specifically:

- **Allocatable**: specifies the total number of allocatable resources like CPUs, memory, and temporary storage on a node.
- **Allocated resources**: specifies the total number of resources like CPUs, memory, and temporary storage that have been allocated to the pods on a node.

Solution

If the resources on a node are not enough for pod scheduling, reduce the node load through either of the following ways:

- Delete unnecessary pods.
- Restrict the resource configurations of pods based on service requirements.
- Add more nodes to the cluster.

Restricted Node Scheduling with the node.kubernetes.io/route-unreachable Taint

Possible Cause

The network infrastructure's route tables allow the container networks in a cluster that uses the VPC network model to be accessible. A newly created CCE node has network isolation support. If the node network is not functioning properly, the system will automatically add the **node.kubernetes.io/route-unreachable** taint to the node and remove it once the network is ready. If the **node.kubernetes.io/route-unreachable** taint remains for an extended period, it indicates abnormal network connectivity for the node.

Symptom

A newly created node is **restricted for scheduling** for a long time.

Solution

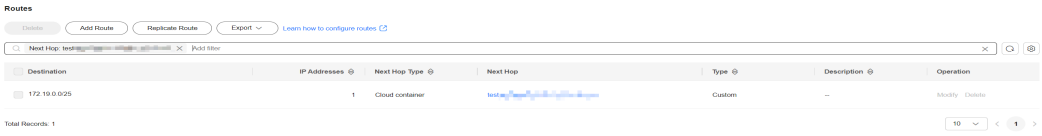
Step 1 If there are other normal nodes in the cluster, run the ping command to check the network connectivity between containers on different nodes.

Create a container for testing. In the following example, {node_ip} indicates the IP address of the abnormal node.

```
kind: Pod
apiVersion: v1
metadata:
  name: nginx
  namespace: default
spec:
  containers:
    - name: container-1
      image: nginx:latest
      imagePullPolicy: IfNotPresent
  imagePullSecrets:
    - name: default-secret
  affinity:
    nodeAffinity:
      requiredDuringSchedulingIgnoredDuringExecution:
        nodeSelectorTerms:
          - matchExpressions:
              - key: kubernetes.io/hostname
                operator: In
                values:
                  - {node_ip}
  schedulerName: default-scheduler
  tolerations:
    - key: node.kubernetes.io/route-unreachable
      operator: Exists
      effect: NoSchedule
```

After the container is started, log in to another normal node and ping the IP address of the container. If the communication is abnormal, go to the next step.

Step 2 On the **Route Tables** page of the VPC console, check whether the node route has been added, whether the next hop type is cloud container, and whether the next hop is the node name. If the node route is present in the route table but the node is still unable to connect to the network, it suggests a potential problem with the underlying network. In such situations, submit a service ticket to the networking team for assistance.



Step 3 If the fault persists, submit a service ticket to CCE for troubleshooting.

----End

Node Unavailable Due to OOM

Possible Cause

Many containers are scheduled onto a node, using up all its resources and causing an OOM issue. This issue is primarily seen on nodes running the Docker container engine.

Symptom

If a node in the cluster is assigned too many containers, the node OS might crash, rendering the node unavailable. You may see the information similar to the following after logging in to the node with VNC.

```
[13513.736557] Out of memory: Killed process 929925 (runc:[2:INIT]) total-vm:482040kB, anon-rss:10624kB, file-rss:0kB, shmem-rss
:3740kB
[13750.207476] Out of memory: Killed process 946406 (icagent) total-vm:2406152kB, anon-rss:188804kB, file-rss:0kB, shmem-rss:0kB
[13975.200701] Out of memory: Killed process 1007445 (icagent) total-vm:2540280kB, anon-rss:200724kB, file-rss:0kB, shmem-rss:0k
B
[13989.696601] Out of memory: Killed process 1044681 (runc:[2:INIT]) total-vm:556572kB, anon-rss:12060kB, file-rss:0kB, shmem-rs
s:3740kB
[13991.505970] Out of memory: Killed process 1043529 (runc:[2:INIT]) total-vm:556316kB, anon-rss:10144kB, file-rss:0kB, shmem-rs
s:3768kB
[14004.576616] Out of memory: Killed process 1043096 (runc:[2:INIT]) total-vm:556316kB, anon-rss:10140kB, file-rss:0kB, shmem-rs
s:3748kB
[14021.768607] Out of memory: Killed process 1043017 (runc:[2:INIT]) total-vm:556572kB, anon-rss:10128kB, file-rss:0kB, shmem-rs
s:3736kB
[14037.518461] Out of memory: Killed process 1043508 (runc:[2:INIT]) total-vm:556316kB, anon-rss:10128kB, file-rss:0kB, shmem-rs
s:3712kB
[14043.899817] Out of memory: Killed process 1044436 (runc:[2:INIT]) total-vm:556316kB, anon-rss:10044kB, file-rss:0kB, shmem-rs
s:3732kB
[14071.578923] Out of memory: Killed process 1045349 (runc:[2:INIT]) total-vm:556316kB, anon-rss:10028kB, file-rss:0kB, shmem-rs
s:3748kB
[14082.340704] Out of memory: Killed process 1044358 (runc:[2:INIT]) total-vm:556572kB, anon-rss:10028kB, file-rss:0kB, shmem-rs
s:3692kB
[14115.023905] Dead loop on virtual device gw_11cbf51a, fix it urgently!
[14116.069973] Dead loop on virtual device gw_11cbf51a, fix it urgently!
[14116.572981] Dead loop on virtual device gw_11cbf51a, fix it urgently!
[14117.077956] Dead loop on virtual device gw_11cbf51a, fix it urgently!
[14117.580965] Dead loop on virtual device gw_11cbf51a, fix it urgently!
[14256.444963] Out of memory: Killed process 1065096 (icagent) total-vm:2051040kB, anon-rss:126944kB, file-rss:0kB, shmem-rss:0k
B
[14492.203423] Out of memory: Killed process 1150151 (runc:[2:INIT]) total-vm:556572kB, anon-rss:12924kB, file-rss:4kB, shmem-rs
s:3740kB
[14496.262777] Out of memory: Killed process 1151198 (runc:[2:INIT]) total-vm:556316kB, anon-rss:12144kB, file-rss:0kB, shmem-rs
```

Check the node events:

```
kubectl describe node {nodeName}
```

Pay attention to the abnormal node events in the output.

```
Conditions:
  Type                 Status  LastHeartbeatTime                 LastTransitionTime                Reason                       Message
  ----                 ------  -----------------                 ------------------                ------                       -------
  NetworkUnavailable   False   Sat, 22 Feb 2025 16:07:07 +0800   Sat, 22 Feb 2025 16:07:07 +0800   YangtseIsUp                  Yangtse is running on this node
  MemoryPressure       False   Sat, 22 Feb 2025 17:12:15 +0800   Sat, 22 Feb 2025 17:11:06 +0800   KubeletHasSufficientMemory   kubelet has sufficient memory available
  DiskPressure         False   Sat, 22 Feb 2025 17:12:15 +0800   Sat, 22 Feb 2025 17:11:06 +0800   KubeletHasNoDiskPressure     kubelet has no disk pressure
  PIDPressure          False   Sat, 22 Feb 2025 17:12:15 +0800   Sat, 22 Feb 2025 17:11:06 +0800   KubeletHasSufficientPID      kubelet has sufficient PID available
  Ready                False   Sat, 22 Feb 2025 17:12:15 +0800   Sat, 22 Feb 2025 17:11:07 +0800   KubeletNotReady              PLEG is not healthy: pleg was last seen active 5m1
1.984571513s ago; threshold is 3m0s
```


Solution

If the resources on a node are not enough for pod scheduling, reduce the node load through either of the following ways:

- Reset the faulty node.
- Delete unnecessary pods.
- Restrict the resource configurations of pods based on service requirements.
- Add more nodes to the cluster.

4.2 Node Creation

4.2.1 How Do I Troubleshoot Problems Occurred When Adding Nodes to a CCE Cluster?

Notes

- 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:

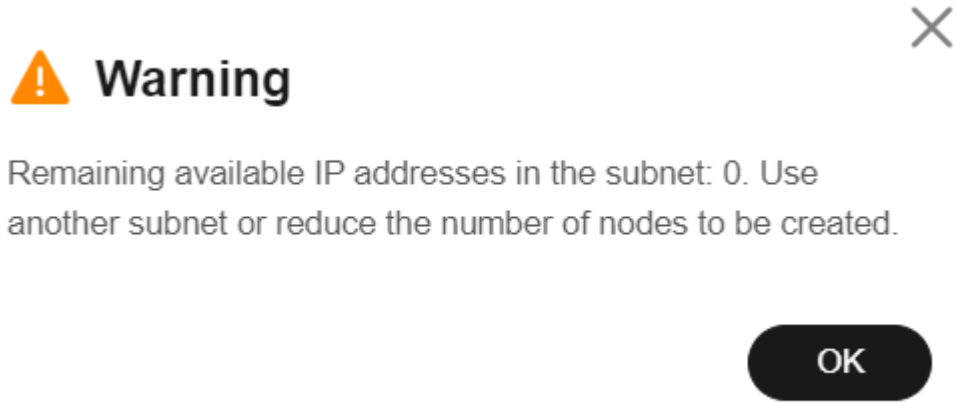
  - /opt/xxxx (excluding **/opt/cloud**)
  - /mnt/xxxx (excluding **/mnt/paas**)
  - /tmp/xxx
  - /var/xxx (excluding key directories such as **/var/lib**, **/var/script**, and **/var/paas**)
  - /xxxx (It cannot conflict with the system directory, such as **bin**, **lib**, **home**, **root**, **boot**, **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: Available IP Addresses in the Subnet

Symptom

New nodes cannot be added to a CCE cluster, and a message is displayed, indicating that the available IP addresses in the subnet are insufficient.



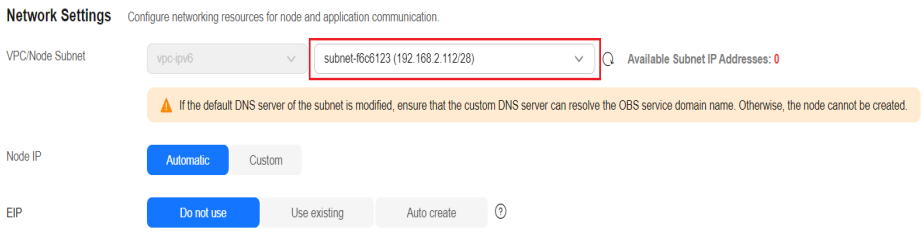
Cause Analysis

The default node subnet CIDR block of the cluster is too small, causing all the available private IP addresses in the subnet to be exhausted. Consequently, it is not possible to allocate any private IP addresses to the nodes.

Solution

- **Scenario 1**: **IP addresses in the VPC CIDR block are not used up.**

  When creating a node, you can select a new node subnet in the network configuration. If no node subnet is available, you can go to the VPC console and create a node subnet. For details, see **Creating a Subnet for an Existing VPC**.

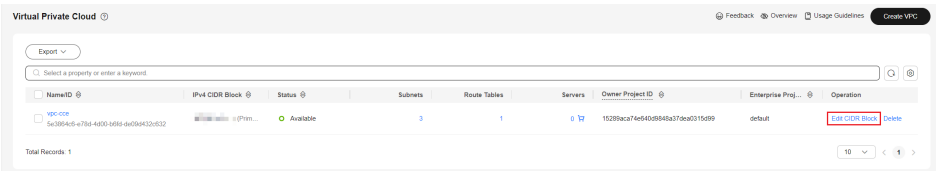


- **Scenario 2: All IP addresses in the VPC CIDR block have been used up.**

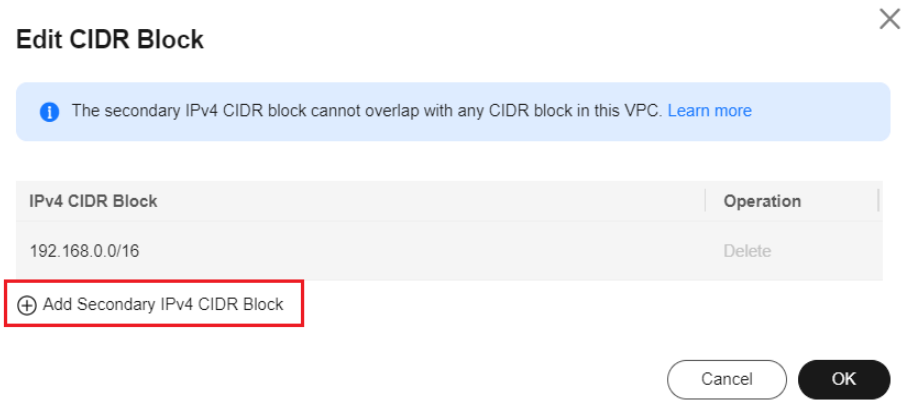
  If all IP addresses in the VPC CIDR block have been used up, you need to expand the VPC CIDR block and create a node subnet.

  a. Log in to the management console. In the service list, choose **Virtual Private Cloud**. In the VPC list, locate the row containing the target VPC and click **Edit CIDR block** in the **Operation** column.

  The following shows an example.

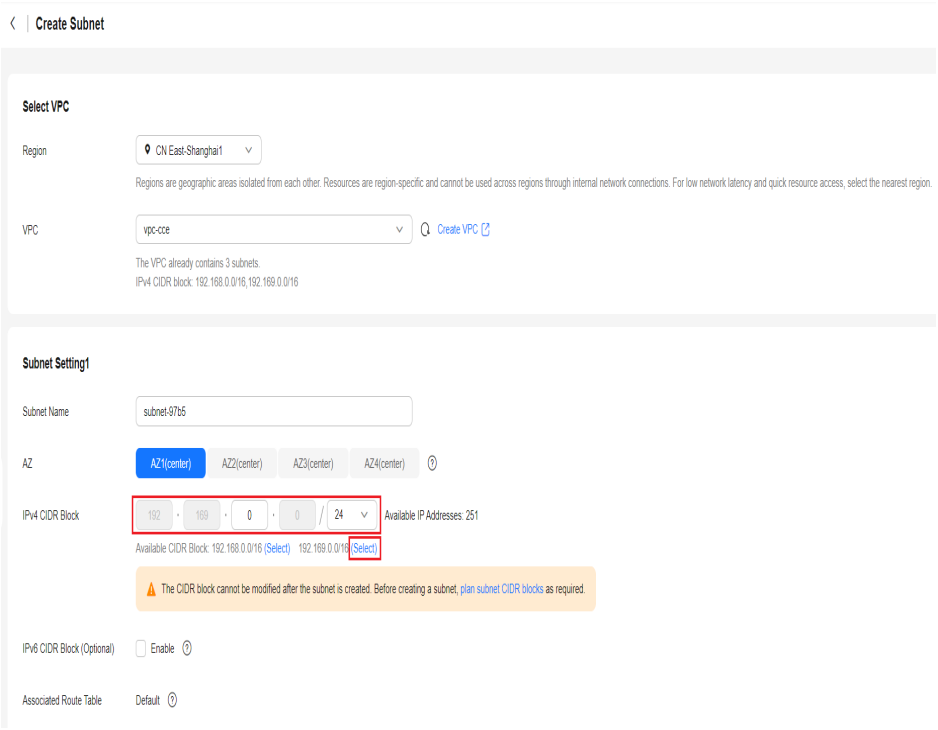


  b. After adding a secondary CIDR block, click **OK**.

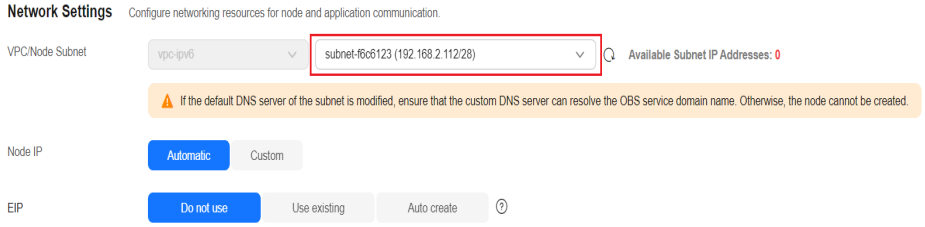


c. In the navigation pane, choose **Subnets** and click **Create Subnet** to create a subnet for the VPC where the cluster resides.

For details, see **Creating a Subnet for an Existing VPC**.



d. Go back 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 CIDR blocks. If the service requirements still cannot be met, you can add more subnets.
2. Subnets in the same VPC can communicate with each other through the private network.

Check Item 2: EIP Quota

Symptom

When a node is added, **EIP** is set to **Auto create**. 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 **Service List** > **Compute** > **Elastic Cloud Server**.
  c. In the ECS list, locate the target ECS and click its name.
  d. On the page displayed, click the **EIPs** tab. In the EIP list, locate the row containing the target EIP, click **Unbind**, and then click **Yes**.

  Figure 4-2 Unbinding an EIP

  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

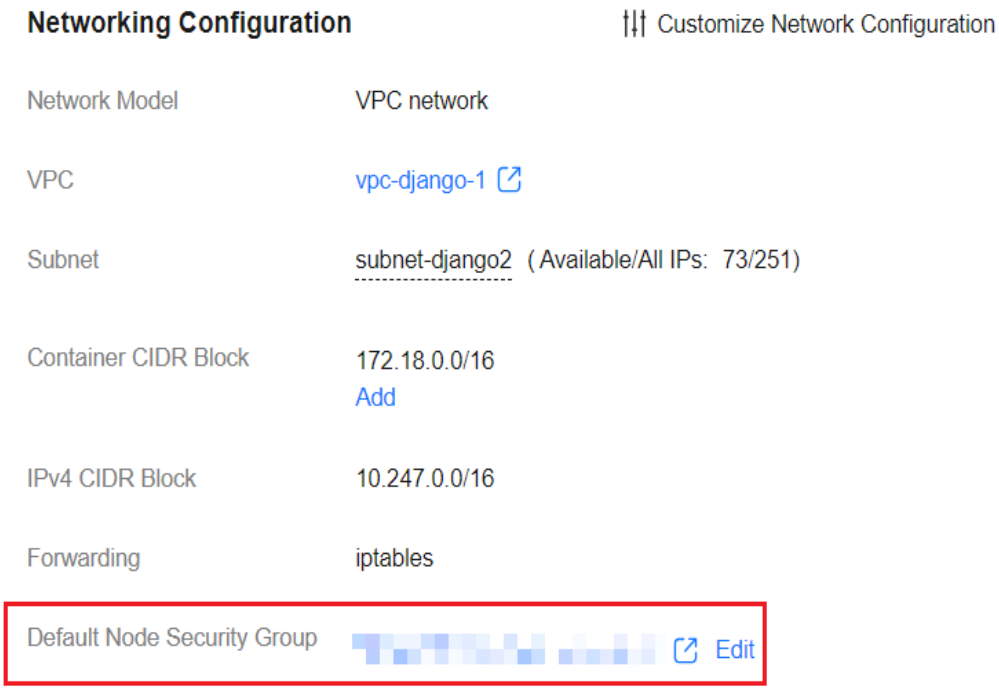
Symptom

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 the value of **Default Node Security Group** to check whether the default security group is deleted and whether the security group rules comply with **How Can I Configure a Security Group Rule for a Cluster?**

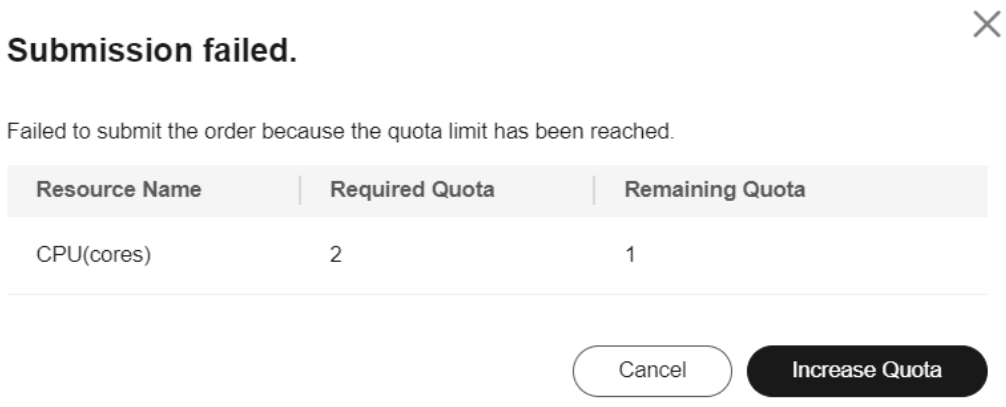
If your account has multiple clusters and you need to manage network security policies of nodes centrally, you can specify custom security groups. For details, see **Changing the Default Security Group of a Node**.



Check Item 4: Resource Quota

Symptom

When a node is added to a CCE cluster, a message is displayed, indicating that the resource quota is insufficient.



Solution

You can click **Increase Quota** to go to the corresponding page and increase the quota.

4.2.2 How Do I Troubleshoot Problems Occurred When Accepting Nodes into a CCE Cluster?

Overview

This section describes how to troubleshoot the problems occurred when you accept or add existing ECSs to a CCE 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.
- The ECS system and data disks will be formatted while the ECS is being accepted into a cluster. Ensure that data in the disks has been backed up.
- During the acceptance of an ECS, do not perform any operation on the ECS through the ECS console.

Notes and Constraints

- BMSs and ECSs, as well as DeHs can be managed.

Prerequisites

The cloud servers to be managed must meet the following requirements:

- The node to be accepted must be in the **Running** state and not used by other clusters. In addition, the node to be accepted does not carry the CCE-Dynamic-Provisioning-Node tag.
- The node to be accepted and the cluster must be in the same VPC. (If the cluster version is earlier than v1.13.10, the node to be accepted and the CCE cluster must be in the same subnet.)
- Data disks must be attached to the nodes to be managed if the system components of these nodes are stored separately. These nodes can be attached with either a local disk (disk-intensive disk) or a data disk of at least 20 GiB. Additionally, any data disks already attached must not be smaller than 10 GiB. For details about how to attach a data disk, see **Adding a Disk to an ECS**.
- The node to be accepted must have at least 2 CPU cores, 4 GiB of memory, and only one network interface.
- If an enterprise project is used, the node to be accepted and the cluster must be in the same enterprise project. Otherwise, resources cannot be identified during management. As a result, the node cannot be accepted.
- Only cloud servers with the same data disk configuration can be accepted in batches for management.
- If IPv6 is enabled for a cluster, only nodes in a subnet with IPv6 enabled can be accepted and managed. If IPv6 is not enabled for the cluster, only nodes in a subnet without IPv6 enabled can be accepted.

- CCE Turbo clusters require that each node supports supplementary network interfaces, or you will need to bind at least 16 network interfaces. For details about the node flavors, see the options provided on the console when you create a node.
- Data disks that have been partitioned will be ignored during node management. Ensure that there is at least one unpartitioned data disk meeting the specifications is attached to the node.

Procedure

View the cluster log information to locate the failure cause and rectify the fault.

Step 1 Log in to the CCE console and 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 accept the node into a cluster again.

----End

Common Issues

If a node fails to be managed, a message will be displayed, indicating that the disk partitioning does not work:

```
Install config-prepare failed: exit status 1, output: [ Mon Jul 17 14:26:10 CST 2023 ] start install config-prepare\nNAME MAJ:MIN RM SIZE RO TYPE MOUNTPOINT\nsda 8:0 0 40G 0 disk \n└─sda1 8:1 0 40G 0 part /\nsdb 8:16 0 100G 0 disk \n└─sdb1 8:17 0 100G 0 part
disk /dev/sda has been partition, will skip this device\nRaw disk /dev/sdb has been partition, will skip this device\nwarning: selector can not match any evs volume
```

To resolve this issue, attach an unpartitioned data disk of 20 GiB or higher to the node. After the node is managed, the unpartitioned data disk is used to store the container engine and kubelet. You can perform operations on the partitioned data disk that does not work as required.

4.2.3 What Should I Do If a Node Cannot Be Managed and an Error Message Appears Saying That the Node Failed to Install?

Symptom

A node fails to be accepted into a cluster.

Possible Cause

Log in to the node and check the **/var/paas/sys/log/baseagent/baseagent.log** installation log. The following error information is displayed:

Check the Logical Volume Manager (LVM) settings of the node. It is found that the LVM logical volume is not created in **/dev/vdb**.

Solution

Run the following command to manually create a logical volume:

```
pvcreate /dev/vdb
vgcreate vgpaas /dev/vdb
```

After the node is reset on the GUI, the node becomes normal.

4.3 Node Running

4.3.1 What Should I Do If a Cluster Is Available But Some Nodes in It Are Unavailable?

If you encountered a fault that a cluster is available but some nodes in it are unavailable, you can rectify this fault by referring to the methods provided in this section.

Mechanism for Detecting Node Unavailability

Kubernetes provides heartbeats to help you detect whether a node is available. For details about the mechanism and detection interval, see **Heartbeats**.

Fault Locating

Possible causes are described here in order of how likely they are to occur.

If the fault persists after you have ruled out a cause, check other 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 Has Been Changed**
- **Check Item 5: Whether the Security Group Rules Contain the One That Allows the Communication Between the Master Nodes and the Worker Nodes**
- **Check Item 6: Whether the Disk Is Abnormal**
- **Check Item 7: Whether Internal Components Are Normal**
- **Check Item 8: Whether the DNS Address Is Properly Configured**
- **Check Item 9: Whether the vdb Disk on the Node Has Been Deleted**
- **Check Item 10: Whether the Docker Service Is Normal**
- **Check Item 11: Whether a Yearly/Monthly Node Is Being Unsubscribed**
- **Check Item 12: Whether the Node Certificate Has Taken Effect**

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**. In the right pane, click the **Nodes** tab, locate the row containing the unavailable node, and click **Monitor**.

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 on the node will result in a high network latency or trigger a system OOM, so the node is displayed as unavailable.

----End

Solution

1. Reduce the number of workloads on the node by migrating the services to other nodes and configure resource limits 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. Restart the node on the ECS console.
6. Add more nodes and deploy memory-intensive containers separately.
7. Reset the nodes. For details, see **Resetting a Node**.

After the nodes become 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 Locate the Fault When a Cluster 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 node.
- If the ECS is faulty, restart it to rectify the fault.
- If the ECS status is **Running**, log in to the ECS and locate the fault by referring 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 is the same as that on the VM and whether the password or key can be used to log in to the node.

Figure 4-3 Checking the displayed node name

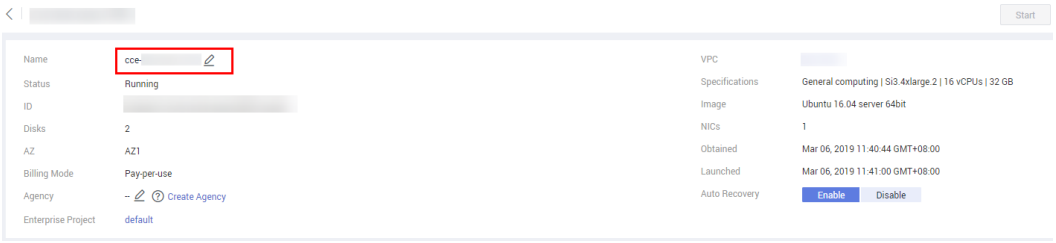
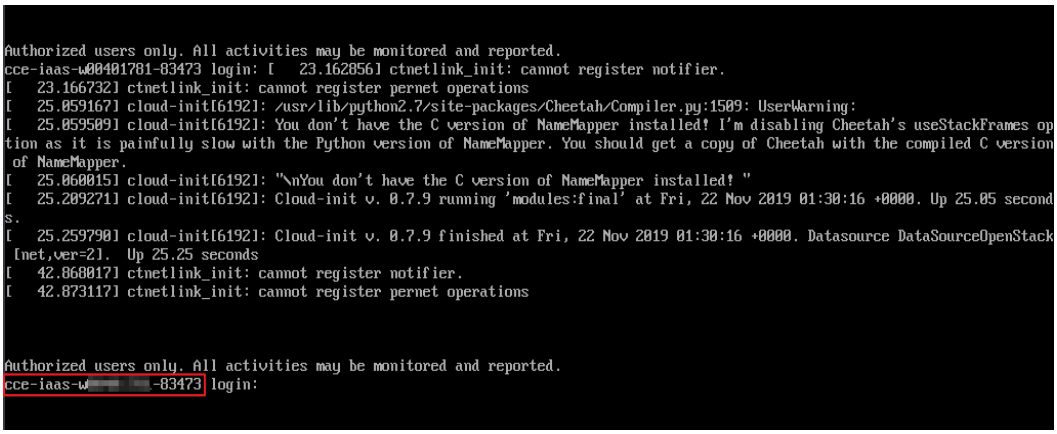


Figure 4-4 Checking the node name on the VM and whether the node can be logged in to



If the node names are inconsistent and the node cannot be logged in to using the password or key, it means that a Cloud-Init problem occurred when the 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 Has Been Changed

Log in to the VPC console. In the navigation pane, choose **Access Control** > **Security Groups** and find the master node security group of the cluster.

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 and then **-cce-control-**.

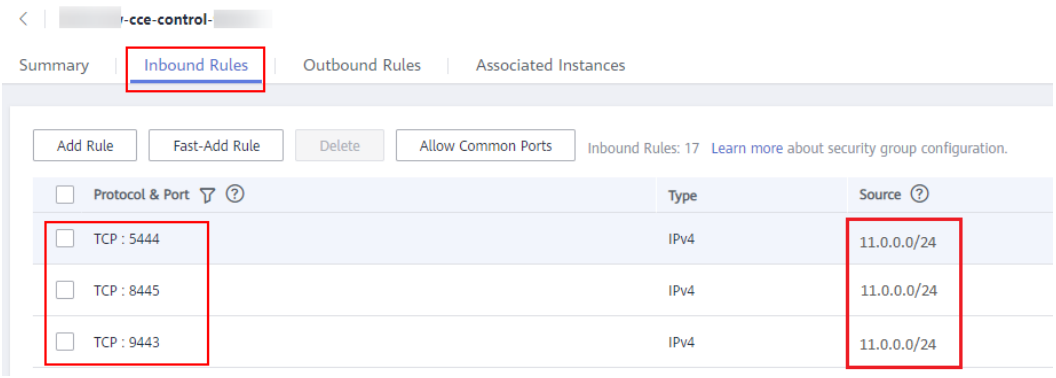
Check whether the security group rules have been changed. For details about security groups, see **How Can I Configure a Security Group Rule for a Cluster?**

Check Item 5: Whether the Security Group Rules Contain the One That Allows the Communication Between the Master Nodes and the Worker Nodes

Check whether such a security group rule exists.

When adding a node to the cluster, add the security group rules in the figure below to the *cluster-name*-**cce-control-***random-ID* security group to ensure the availability of the added node. This is necessary if a secondary CIDR block is added to the VPC of the node subnet and the subnet is in the secondary CIDR block. However, if a secondary CIDR block has already been added to the VPC during cluster creation, this step is not required.

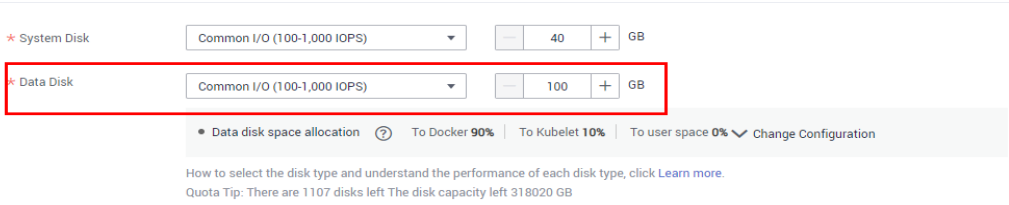
For details about security groups, see **How Can I Configure a Security Group Rule for a Cluster?**



Check Item 6: Whether the Disk Is Abnormal

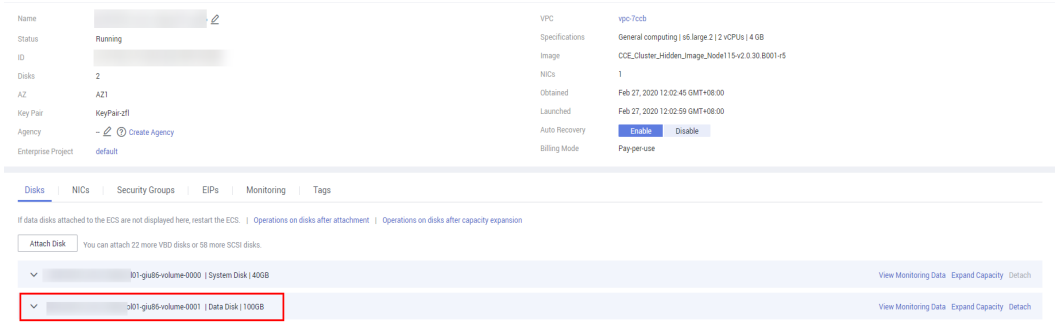
Each new node is equipped with a 100-GiB data disk dedicated for Docker. If this data disk is removed or damaged, the Docker service will be disrupted and the node will become unavailable.

Figure 4-5 The data disk attached to a node during its creation



Click the node name and check whether the data disk attached to the node has been removed. If the disk has been detached from the node, you need to attach another data disk to the node and restart the node. Then the node can be restored.

Figure 4-6 Checking the disk



Check Item 7: Whether Internal Components Are Normal

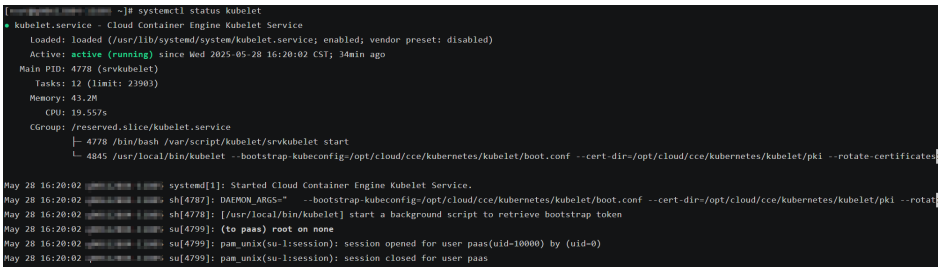
1. Log in to the node and check whether the following key components are running properly:
   - kubelet
   - kube-proxy
   - Network components
     - yangtse: used by clusters that use the VPC or Cloud Native 2.0 networks
     - canal: used by clusters that use the container tunnel networks
   - Runtime: Docker or containerd
   - chronyd

   Check the status of a component. For example, to check the status of kubelet, run the following command:

   ```
   systemctl status kubelet
   ```

   kubelet is a component name. You can replace it as required.

   The expected output is shown in the figure below.

2. If the component is not in the **Active** state, restart it. Specify the restart command based on the faulty component. If yangtse is faulty, run the following command:

   ```
   systemctl restart yangtse
   ```

   Check the component status again.

   ```
   systemctl status yangtse
   ```

3. If the node status is still not restored after the component restart, submit a service ticket and contact customer service.

Check Item 8: Whether the DNS Address Is Properly Configured

Step 1 Log in to the node and 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, it means that there is a domain name resolution failure.

Could not resolve host: test.obs.ap-southeast-1.myhuaweicloud.com; 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.

ping test.obs.ap-southeast-1.myhuaweicloud.com

- If the domain name cannot be pinged, 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 configured incorrectly, leading to the inability to resolve the domain name. To fix this issue, adjust 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

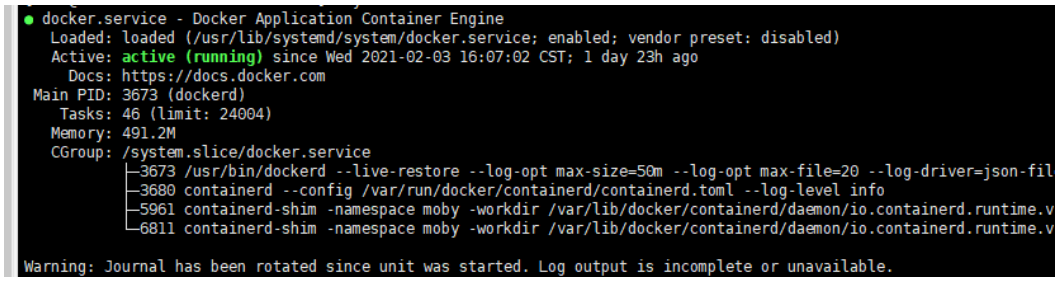
Check Item 9: Whether the vdb Disk on the Node Has Been Deleted

If the vdb disk on a node has been 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
```



If the command fails to be executed 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, takes too long to execute, or if there are over 1000 abnormal containers, you should check if workloads are being repeatedly created

and deleted. If many containers are being created and deleted frequently, it may result in numerous abnormal containers that cannot be cleared promptly.

In this case, stop repeated creation and deletion of workloads or use more nodes to share the load. Typically, the node will be restored after a period of time. If necessary, run the **docker rm** *{container_id}* command to manually clear the abnormal containers.

----End

Check Item 11: Whether a Yearly/Monthly Node Is Being Unsubscribed

Once a node is unsubscribed, it will take some time to process the order, rendering the node unavailable during this period. Typically, the node is expected to be automatically cleared within 5 to 10 minutes.

Check Item 12: Whether the Node Certificate Has Taken Effect

If the region where a cluster is located undergoes a transition between daylight saving time (DST) and standard time (ST), there may be a period of unavailability during the overlapping time. For instance, if you apply to create a node at 02:00 in the morning, the time will shift to 01:00 in the morning when DST changes to ST. This can potentially result in the node being unavailable.

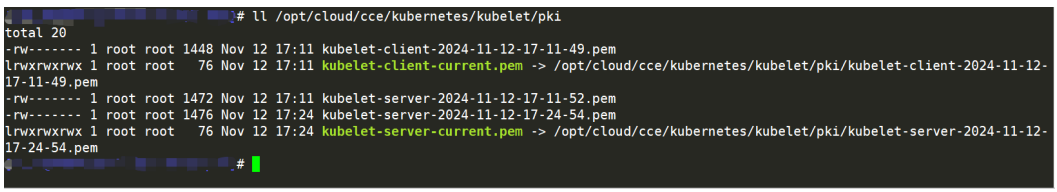
Possible Cause

A node certificate has an effective time set in the future instead of the current time. If the certificate is not yet effective, the node's request to kube-apiserver will be rejected.

Solution

Log in to the node and view the certificate:

```
ll /opt/cloud/cce/kubernetes/kubelet/pki
```



The file name represents the time when the certificate was created. You need to verify if the certificate creation time is after the current time.

If the time when the certificate becomes effective is later than the current time, you can take either of the following actions:

- Delete the node and create a new one.
- Wait until the certificate takes effect, at which point the node will automatically become available.
- Contact O&M personnel and update the node certificate.

4.3.2 How Do I Troubleshoot the Failure to Remotely Log In to a Node in a CCE Cluster?

After creating a node on CCE, you cannot remotely log in to the node using SSH. A message is displayed indicating that the selected key has not been registered on the remote host. In this case, the root user cannot directly log in to the node.

The cause is that the cloud-init is installed on the node on CCE. For the cloud-init, a default Linux user already exists and the user's key is also used for the device running Linux.

Solution

Log in to the device as the **Linux** user, and run the **sudo su** command to switch to the **root** user.

4.3.3 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

4.3.4 How Do I Collect Logs of Nodes in a CCE Cluster?

Paths of Node Logs

The following tables list log files of CCE nodes.

Table 4-3 Node logs

Name	Path
kubelet log	• For clusters of v1.21 or later: **/var/log/cce/kubernetes/kubelet.log** • For clusters of v1.19 or earlier: **/var/paas/sys/log/kubernetes/kubelet.log**
kube-proxy log	• For clusters of v1.21 or later: **/var/log/cce/kubernetes/kube-proxy.log** • For clusters of v1.19 or earlier: **/var/paas/sys/log/kubernetes/kube-proxy.log**
yangtse log (networking)	• For clusters of v1.21 or later: **/var/log/cce/yangtse** • For clusters of v1.19 or earlier: **/var/paas/sys/log/yangtse**
canal log	• For clusters of v1.21 or later: **/var/log/cce/canal** • For clusters of v1.19 or earlier: **/var/paas/sys/log/canal**
System logs	**/var/log/messages**
Container engine Logs	• For Docker nodes: **/var/lib/docker** • For containerd nodes: **/var/log/cce/containerd**

Table 4-4 Add-on logs

Name	Path
everest log	• For v2.1.41 or later: – everest-csi-driver: **/var/log/cce/kubernetes** – everest-csi-controller: **/var/paas/sys/log/kubernetes** • For version earlier than v2.1.41: – everest-csi-driver: **/var/log/cce/everest-csi-driver** – everest-csi-controller: **/var/paas/sys/log/everest-csi-controller**
npd log	• For v1.18.16 or later: **/var/paas/sys/log/kubernetes** • For versions earlier than v1.18.16: **/var/paas/sys/log/cceaddon-npd**
cce-hpa-controller log	• For v1.3.12 or later: **/var/paas/sys/log/kubernetes** • For versions earlier than v1.3.12: **/var/paas/sys/log/ccehpa-controller**

Configuring Log Collection for a Node

CCE allows you to use the Cloud Native Log Collection add-on to collect node logs and report them to LTS, which provides log statistics and analysis. For details, see **Collecting Container Logs Using Cloud Native Log Collection**.

4.3.5 What Can I Do If the Container Network Becomes Unavailable After yum update Is Used to Upgrade the OS?

The CCE console does not support OS upgrades on a node. You are advised not to upgrade the OS using the **yum update** command.

If you upgrade the OS using **yum update**, the container networking will be unavailable.

Perform the following operations to restore the container network:

NOTICE

This restoration method is valid only for EulerOS 2.2.

Step 1 Run the following script as user **root**:

```
#!/bin/bash
function upgrade_kmod()
{
    openvswicth_mod_path=$(rpm -qal openvswitch-kmod)
    rpm_version=$(rpm -qal openvswitch-kmod|grep -w openvswitch|head -1|awk -F "/" '{print $4}')
    sys_version=`cat /boot/grub2/grub.cfg | grep EulerOS|awk 'NR==1{print $3}' | sed 's/[()]//g'`

    if [[ "${rpm_version}" != "${sys_version}" ]];then
        mkdir -p /lib/modules/"${sys_version}"/extra/openvswitch
        for path in ${openvswicth_mod_path[@]};do
            name=$(echo "$path" | awk -F "/" '{print $NF}')
            rm -f /lib/modules/"${sys_version}"/updates/"${name}"
            rm -f /lib/modules/"${sys_version}"/extra/openvswitch/"${name}"
            ln -s "${path}" /lib/modules/"${sys_version}"/extra/openvswitch/"${name}"
        done
    fi
    depmod ${sys_version}
}
upgrade_kmod
```

Step 2 Restart the VM.

----End

Helpful Links

- **High-Risk Operations on Cluster Nodes**

4.3.6 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.

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

4.3.7 Which Ports Are Used to Install kubelet on CCE Cluster Nodes?

The following ports are used:

- **10250 –port**: used by kubelet to communicate with the API server
- **10248 –healthz-port**: used for health checks.
- **10255 –read-only-port**: read-only port, which is used to expose monitoring metrics to external systems

4.3.8 How Do I Configure a Pod to Use the Acceleration Capability of a GPU Node?

Symptom

I have purchased a GPU node, but the operating speed is still slow. How do I configure the pod to use the acceleration capability of the GPU node?

Solution

Solution 1:

You are advised to remove the unschedulable taints from the GPU nodes in the cluster, so that the GPU plug-in driver can be properly installed. In addition, you need to install the GPU driver of a later version.

If a container is not deployed on a GPU node in your cluster, you can configure affinity and anti-affinity policies to prevent the container from being scheduled to the GPU node.

Solution 2:

You are advised to install the GPU driver of a later version and use kubectl to update the GPU plug-in configuration. Add the following configuration:

```
tolerations:
- operator: "Exists"
```

After the configuration is added, the GPU plug-in driver can be properly installed on the GPU node with a taint.

4.3.9 What Should I Do If I/O Suspension Occasionally Occurs When SCSI EVS Disks Are Used?

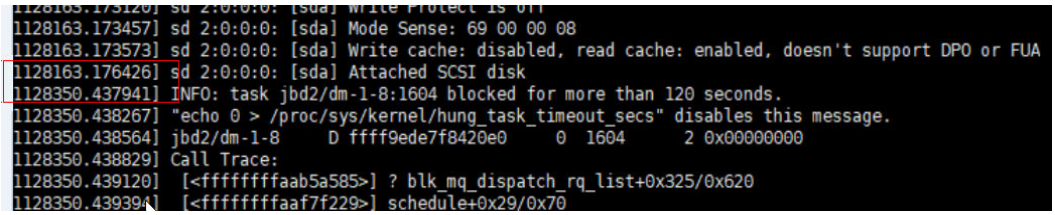
Symptom

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:



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.

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**. For details, see **Resetting a Node**.

4.3.10 What Should I Do If Excessive Docker Audit Logs Affect the Disk I/O?

Symptom

There are a large number of Docker audit logs on existing nodes in some clusters. Due to OS kernel defects, it is slightly possible that I/Os are suspended. You can optimize the audit log rules to avoid this problem.

Impact

Affected cluster versions:

- v1.15.11-r1
- v.1.17.9-r0

NOTICE

- You only need to fix this issue for existing nodes, not for newly created nodes.
- The auditd component needs to be restarted during the upgrade.

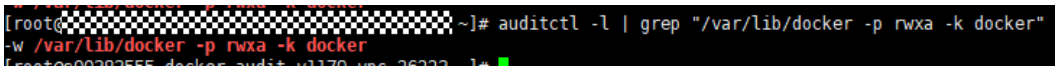
Check Method

Step 1 Log in to the worker node as user **root**.

Step 2 Run the following command to check whether the problem exists on the current node:

auditctl -l | grep "/var/lib/docker -p rwxa -k docker"

If information similar to the following is displayed, the problem exists and needs to be rectified. If no command output is displayed, the node is not affected.



----End

Solution

Step 1 Log in to the worker node as user **root**.

Step 2 Run the following commands:

sed -i "/\/var\/lib\/docker -k docker/d" /etc/audit/rules.d/docker.rules

service auditd restart

----End

Verification Method

Run the following command to check whether the fault is rectified:

auditctl -l | grep "/var/lib/docker -p rwxa -k docker"

If no command output is displayed, the problem has been resolved.

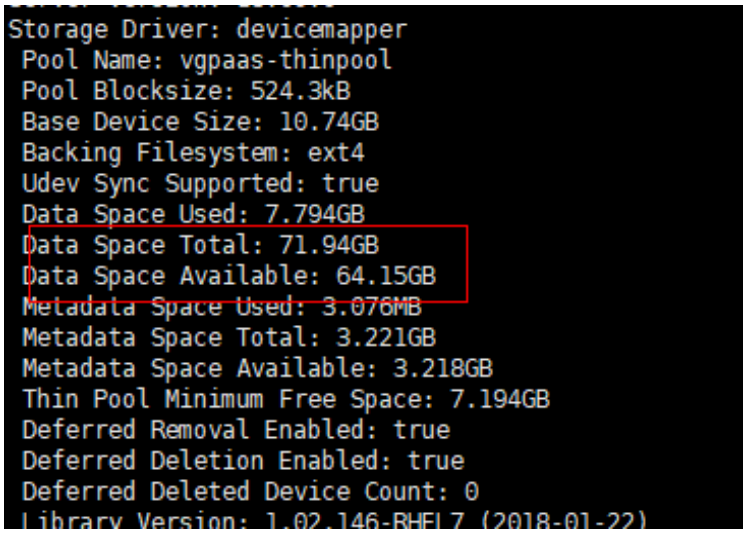
4.3.11 How Do I Fix an Abnormal Container or Node Due to No Thin Pool Disk Space?

Symptom

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.



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 operations:

Step 1 Expand the capacity of a data disk on the EVS console. For details, see **Expanding EVS Disk Capacity**.

Only the storage capacity of EVS disks can be expanded. You need to perform the following operations to expand the capacity of logical volumes and file systems.

Step 2 Log in to the CCE console and click the cluster name to access the cluster console. In the navigation pane, choose **Nodes**. In the right pane, click the **Nodes** tab, locate the row containing the target node, and choose **More** > **Sync Server Data** in the **Operation** column.

Step 3 Log in to the target node.

Step 4 Run **lsblk** to view the block device information of the node.

A data disk is divided depending on the container storage **Rootfs**:

Overlayfs: No independent thin pool is allocated. Image data is stored in **dockersys**.

1. Check the disk and partition space of the device.

```
# 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  150G  0 disk      # The data disk has been expanded to 150 GiB, but 50 GiB space is free.
├─vgpaas-dockersys 253:0    0   90G  0 lvm   /var/lib/containerd
└─vgpaas-kubernetes 253:1    0   10G  0 lvm   /mnt/paas/kubernetes/kubelet
```

2. Expand the disk capacity.

   Add the new disk capacity to the **dockersys** logical volume used by the container engine.

   a. Expand the PV capacity so that LVM can identify the new EVS capacity. */dev/sdb* specifies the physical volume where dockersys is located.

   ```
   pvresize /dev/sdb
   ```

   Information similar to the following is displayed:

   ```
   Physical volume "/dev/sdb" changed
   1 physical volume(s) resized or updated / 0 physical volume(s) not resized
   ```

   b. Expand 100% of the free capacity to the logical volume. *vgpaas/dockersys* specifies the logical volume used by the container engine.

   ```
   lvextend -l+100%FREE -n vgpaas/dockersys
   ```

   Information similar to the following is displayed:


```
Size of logical volume vgpaas/dockersys changed from <90.00 GiB (23039 extents) to 140.00 GiB (35840 extents).
Logical volume vgpaas/dockersys successfully resized.
```

c. Adjust the size of the file system. */dev/vgpaas/dockersys* specifies the file system path of the container engine.

```
resize2fs /dev/vgpaas/dockersys
```

Information similar to the following is displayed:

```
Filesystem at /dev/vgpaas/dockersys is mounted on /var/lib/containerd; on-line resizing required
old_desc_blocks = 12, new_desc_blocks = 18
The filesystem on /dev/vgpaas/dockersys is now 36700160 blocks long.
```

3. Check whether the capacity has been expanded.

```
# 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  150G  0 disk
├─vgpaas-dockersys  253:0    0   140G  0 lvm  /var/lib/containerd
└─vgpaas-kubernetes 253:1    0   10G  0 lvm  /mnt/paas/kubernetes/kubelet
```

Device Mapper: A thin pool is allocated to store image data.

1. Check the disk and partition space of the device.

```
# lsblk
NAME                          MAJ:MIN RM  SIZE RO TYPE MOUNTPOINT
vda                           8:0    0   50G  0 disk
└─vda1                          8:1    0   50G  0 part /
vdb                           8:16   0  200G  0 disk
├─vgpaas-dockersys              253:0    0   18G  0 lvm  /var/lib/docker
├─vgpaas-thinpool_tmeta           253:1    0    3G  0 lvm
│ └─vgpaas-thinpool               253:3    0   67G  0 lvm                # Space used by thin pool
│   ...
├─vgpaas-thinpool_tdata           253:2    0   67G  0 lvm
│ └─vgpaas-thinpool               253:3    0   67G  0 lvm
│   ...
└─vgpaas-kubernetes               253:4    0   10G  0 lvm  /mnt/paas/kubernetes/kubelet
```

2. Expand the disk capacity.

Option 1: Add the new disk capacity to the thin pool.

a. Expand the PV capacity so that LVM can identify the new EVS capacity. */dev/vdb* specifies the physical volume where thin pool is located.

```
pvresize /dev/vdb
```

Information similar to the following is displayed:

```
Physical volume "/dev/vdb" changed
1 physical volume(s) resized or updated / 0 physical volume(s) not resized
```

b. Expand 100% of the free capacity to the logical volume. *vgpaas/thinpool* specifies the logical volume used by the container engine.

```
lvextend -l+100%FREE -n vgpaas/thinpool
```

Information similar to the following is displayed:

```
Size of logical volume vgpaas/thinpool changed from <67.00 GiB (23039 extents) to <167.00 GiB (48639 extents).
Logical volume vgpaas/thinpool successfully resized.
```

c. Do not need to adjust the size of the file system, because the thin pool is not mounted to any devices.

d. Run the **lsblk** command to check the disk and partition space of the device and check whether the capacity has been expanded. If the new disk capacity was added to the thin pool, the capacity has been expanded.


```
# lsblk
NAME                        MAJ:MIN RM  SIZE RO TYPE MOUNTPOINT
vda                         8:0    0   50G  0 disk
└─vda1                        8:1    0   50G  0 part /
vdb                         8:16   0  200G  0 disk
├─vgpaas-dockersys            253:0    0   18G  0 lvm  /var/lib/docker
├─vgpaas-thinpool_tmeta         253:1    0    3G  0 lvm
│ └─vgpaas-thinpool             253:3    0   167G  0 lvm            # Thin pool space after
capacity expansion
│   ...
├─vgpaas-thinpool_tdata         253:2    0   67G  0 lvm
│ └─vgpaas-thinpool             253:3    0   67G  0 lvm
│   ...
└─vgpaas-kubernetes             253:4    0   10G  0 lvm  /mnt/paas/kubernetes/kubelet
```

Option 2: Add the new disk capacity to the **dockersys** disk.

a. Expand the PV capacity so that LVM can identify the new EVS capacity. */dev/vdb* specifies the physical volume where dockersys is located.

```
pvresize /dev/vdb
```

Information similar to the following is displayed:

```
Physical volume "/dev/vdb" changed
1 physical volume(s) resized or updated / 0 physical volume(s) not resized
```

b. Expand 100% of the free capacity to the logical volume. *vgpaas/dockersys* specifies the logical volume used by the container engine.

```
lvextend -l+100%FREE -n vgpaas/dockersys
```

Information similar to the following is displayed:

```
Size of logical volume vgpaas/dockersys changed from <18.00 GiB (4607 extents) to <118.00
GiB (30208 extents).
Logical volume vgpaas/dockersys successfully resized.
```

c. Adjust the size of the file system. */dev/vgpaas/dockersys* specifies the file system path of the container engine.

```
resize2fs /dev/vgpaas/dockersys
```

Information similar to the following is displayed:

```
Filesystem at /dev/vgpaas/dockersys is mounted on /var/lib/docker; on-line resizing required
old_desc_blocks = 3, new_desc_blocks = 15
The filesystem on /dev/vgpaas/dockersys is now 30932992 blocks long.
```

d. Run the **lsblk** command to check the disk and partition space of the device and check whether the capacity has been expanded. If the new disk capacity was added to the dockersys, the capacity has been expanded.

```
# lsblk
NAME                        MAJ:MIN RM  SIZE RO TYPE MOUNTPOINT
vda                         8:0    0   50G  0 disk
└─vda1                        8:1    0   50G  0 part /
vdb                         8:16   0  200G  0 disk
├─vgpaas-dockersys            253:0    0   118G  0 lvm  /var/lib/docker     # dockersys after
capacity expansion
├─vgpaas-thinpool_tmeta         253:1    0    3G  0 lvm
│ └─vgpaas-thinpool             253:3    0   67G  0 lvm
│   ...
├─vgpaas-thinpool_tdata         253:2    0   67G  0 lvm
│ └─vgpaas-thinpool             253:3    0   67G  0 lvm
│   ...
└─vgpaas-kubernetes             253:4    0   10G  0 lvm  /mnt/paas/kubernetes/kubelet
```

----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.

4.3.12 Where Can I Get the Listening Ports of CCE Worker Nodes?

Table 4-5 Listening ports of a worker node

Destination Port	Protocol	Description
10248	TCP	Health check port for kubelet
10250	TCP	Service port of kubelet to provide monitoring information about workloads on nodes and access channels for containers
10255	TCP	Read-only port of kubelet to provide monitoring information about workloads on the node
Dynamic port (related to the range of the host machine, for example, the kernel parameter **net.ipv4.ip_local_port_range**)	TCP	Random port listened by kubelet, which is used to communicate with CRI Shim to obtain the EXEC URL.
10249	TCP	kube-proxy metric port to provide kube-proxy monitoring information
10256	TCP	Health check port for kube-proxy
Dynamic port (32768-65535)	TCP	WebSocket listening port for functions such as docker exec
Dynamic port (32768-65535)	TCP	WebSocket listening port for functions such as containerd exec
28001	TCP	Local listening port of ICAgent to receive syslog logs of the node
28002	TCP	Health check port for ICAgent

Destination Port	Protocol	Description
20101	TCP	Health check port of yangtse-agent/canal-agent (involved when the container tunnel network model is used)
20104	TCP	Metric port of yangtse-agent/ canal-agent to provide component monitoring information (involved when the container tunnel network model is used)
3125	TCP	Health check listening port of everest-csi-driver
3126	TCP	everest-csi-driver pprof port
19900	TCP	Server port for the health check of node-problem-detector
19901	TCP	Port for connecting node-problem-detector to Prometheus to collect monitoring data
4789	UDP	OVS listening port, which is used to transmit VXLAN packets in container networking (involved when the container tunnel network model is used)
4789	UDPv6	OVS listening port, which is used to transmit VXLAN packets in container networking (involved when the container tunnel network model is used)
Dynamic port (30000-32767)	TCP	Listening port of kube-proxy for layer-4 load balancing. Kubernetes allocates a random port to NodePort and LoadBalancer Services. The default port number ranges from 30000 to 32767.
Dynamic port (30000-32767)	UDP	Listening port of kube-proxy for layer-4 load balancing. Kubernetes allocates a random port to NodePort and LoadBalancer Services. The default port number ranges from 30000 to 32767.
123	UDP	Listening port of ntpd used for time synchronization

Destination Port	Protocol	Description
20202	TCP	Listening port of PodLB for layer-7 load balancing, which forwards container image pull requests.

4.3.13 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?

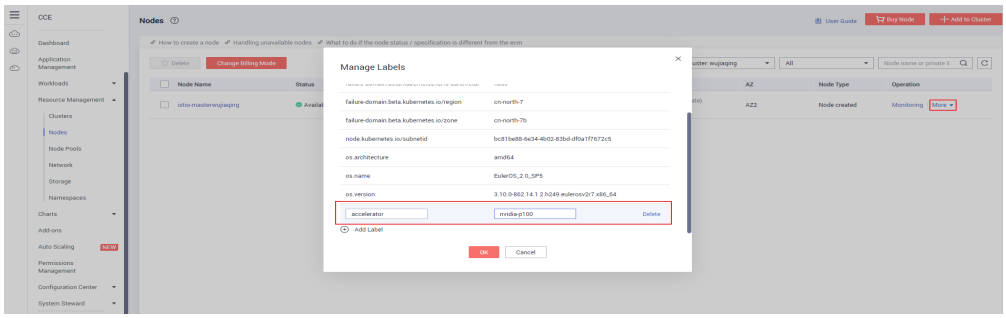
Symptom

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

Fault Locating

1. Check whether the node is attached with NVIDIA label.



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 I Deploy a Service on a GPU Node?

4.3.14 What Can I Do If the Time of CCE Nodes Is Not Synchronized with the NTP Server?

Symptom

In special scenarios, if the ntpd on a node cannot access the NTP server for a long time, the time offset may be too large and cannot be automatically restored.

Problem Detection

You can detect the problem by installing the CCE Node Problem Detector add-on that has the node time synchronization check items. For details, see **CCE Node Problem Detector**.

Possible Cause

This is a known issue on nodes running EulerOS or CentOS. Nodes running other types of OSs are not affected by this issue.

NOTE

This issue has been resolved in clusters of v1.19.16-r7, v1.21.9-r10, and v1.23.7-r10.

Solution

- If your cluster version is v1.19.16-r7, v1.21.9-r10, v1.23.7-r10, or later, the clock of the nodes in such clusters has been configured with chronyd for proper time synchronization. In this case, you can reset the nodes to rectify the fault.
- If your cluster version does not meet the requirements, you are advised to upgrade the cluster to v1.19.16-r7, v1.21.9-r10, v1.23.7-r10, or later, and then reset the nodes.

4.3.15 What Should I Do If the Data Disk Usage Is High Because a Large Volume of Data Is Written Into the Log File?

Symptom

Service containers on nodes that use containerd as the container runtime continuously write a large volume of data into the log file, resulting in full space of the **/var/lib/containerd** directory and slowing down the creation and deletion of containers on the node. This may evict pods and cause problems like high disk usage and abnormal nodes.

Possible Cause

For such service containers, if their logs are generated to the STDOUT, the kubelet will dump the logs. The kubelet also maintains the lifecycle of all containers on the node.

The kubelet will be overloaded if there are too many service containers on a node and they write a large volume of data into the log files. If the load exceeds a certain threshold, kubelet will dump the logs to the disk, which further results in a high disk usage. Operations such as container creation and deletion on the node will be affected.

Solution

Typically, for a node with 8 vCPUs and 16 GiB of memory, and a 100-GiB data disk, the standard log output rate of a single container should be less than or equal to 512 KB/s and the overall standard log output rate of all containers on the node should be less than or equal to 5 MB/s. If a large number of logs are generated, resolve this issue in either of the following ways:

- Do not schedule containers which generate too many logs on the same node. For example, configure anti-affinity policies for pods running such containers or reduce the maximum number of pods on a single node.
- Attach an additional data disk separately. For example, you can attach an extra data disk or mount a dynamically provisioned storage volume when creating a node so that logs can be written to files in it.

4.3.16 Why Does My Node Memory Usage Obtained by Running the kubelet top node Command Exceed 100%?

Symptom

The memory usage of the node obtained on the CCE console is not high, but that obtained by running the **kubelet top node** command exceeded 100%.

```
NAME           CPU(cores)   CPU%   MEMORY(bytes)   MEMORY%
192.168.0.243  79m          4%     2357Mi          109%
```

Possible Cause

kubectl top node calls the kubelet metrics API to obtain data, and the displayed information indicates the total number of used resources on the node divided by all allocatable resources.

For details, see **https://github.com/kubernetes/kubernetes/issues/86499**.

Example Scenarios

To obtain the parameters of a node, run **kubectl describe node**. The following is an example:

```
...
Capacity:
  cpu:                2
  ephemeral-storage:  51286496Ki
```



```
  hugepages-1Gi:      0
  hugepages-2Mi:      0
  localssd:           0
  localvolume:        0
  memory:             4030180Ki
  pods:               40
Allocatable:
  cpu:                1960m
  ephemeral-storage:  47265634636
  hugepages-1Gi:      0
  hugepages-2Mi:      0
  localssd:           0
  localvolume:        0
  memory:             2213604Ki
  pods:               40
...
```

- The **Capacity.memory** field whose value is **4030180Ki** indicates the total memory of the node.
- The **Allocatable.memory** field whose value is **2213604Ki** indicates the allocatable memory of the node.
- The value of the node's used memory in this example is **2413824Ki**. To obtain the value, run the following command:

  ```
  kubectl get --raw /apis/metrics.k8s.io/v1beta1/nodes/
  ```

  Information similar to the following will be displayed:

  ```
  {
     "kind": "NodeMetricsList",
     "apiVersion": "metrics.k8s.io/v1beta1",
     "metadata": {},
     "items": [
        {
           ...
           "timestamp": "2023-08-15T14:09:38Z",
           "window": "1m0.177s",
           "usage": {
              "cpu": "78528126n",
              "memory": "2413824Ki"
           }
        }
     ]
  }
  ```

To check the memory usage of the node, run **kubectl top node**.

Memory usage of a node = Used memory of the node/Allocatable memory of the node = 2413824Ki/2213604Ki = 109%

The actual memory usage of the node is calculated as follows:

Actual memory usage of the node = Used memory of the node/Total memory of the node = 2413824Ki/4030180Ki = 59.9%

4.3.17 What Should I Do If "Failed to reclaim image" Is Displayed in the Node Events?

Symptom

In the event of a node, the alarm "Failed to reclaim image" is repeatedly generated. The following shows an example:


```
wanted to free xx bytes, but freed xx bytes space with errors in image deletion: rpc error: code = Unknown
desc = Error response from daemon: conflict: unable to remove repository reference "imageName:tag"
(must force) - container 966fce58d9b8 is using its referenced image 50a7aa6fa56a
```

In this event, the container with ID **966fce58d9b8** was stopped but not completely deleted.

Possible Cause

kubelet periodically reclaims images that are not in use based on the **imageGCHighThresholdPercent** and **imageGCLowThresholdPercent** parameters. If you run the **docker** or **crictl** command on a node to start a container, the container will be in an exit state but is not fully deleted after being stopped. This means that the container still needs the image. However, kubelet cannot detect whether the image is being used by the container if it does not belong to any pods on the node. If kubelet attempts to delete the container image, the container runtime will stop it because the container still needs the image. As a result, kubelet is unable to regularly reclaim the container image.

Solution

Log in to the node, get the container for which the alarm is generated, and check whether the container is exited. Replace *{containerId}* with the container ID in the alarm.

- To get the container on a node using Docker, run the following command:
  ```
  docker ps -a | grep {containerId}
  ```
- To get the container on a node using containerd, run the following command:
  ```
  crictl ps -a | grep {containerId}
  ```

If the container is no longer used, delete this container. Replace *{containerId}* with the container ID in the alarm.

- To delete the container on a node using Docker, run the following command:
  ```
  docker rm {containerId}
  ```
- To delete the container on a node using containerd, run the following command:
  ```
  crictl rm {containerId}
  ```

After the faulty container is deleted, kubelet can reclaim images normally.

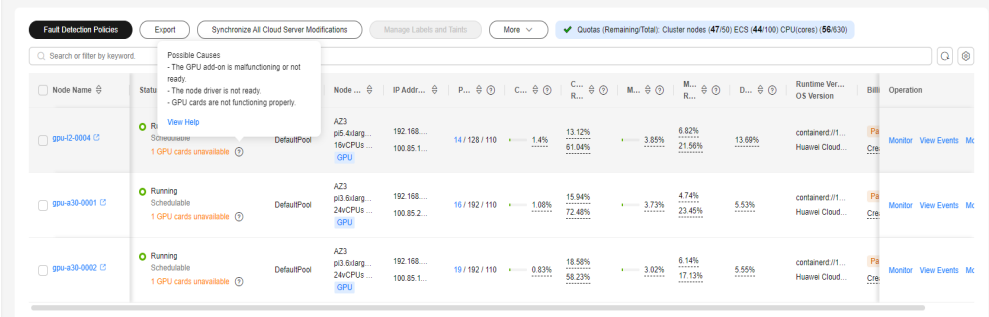
4.3.18 What Can I Do If a GPU Card Is Unavailable on a GPU Node?

Symptom

A GPU card on a GPU node is unavailable. The possible causes include:

- The CCE AI Suite (NVIDIA GPU) add-on is not ready or malfunctioning.
- The node driver is not ready.
- The GPU card is abnormal.

Figure 4-7 An unavailable GPU card



Solution

Check whether the driver is faulty. Then, check the **device-plugin** component of the CCE AI Suite (NVIDIA GPU) add-on. Finally, check the GPU card.

Handling a Driver Fault

Step 1 Check the nvidia-driver-installer pod status.

Log in to the CCE console and click the cluster name to access the cluster **Overview** page. In the navigation pane, choose **Nodes**. In the right pane, click the **Nodes** tab. Locate the row containing the target node, choose **More** > **Pods** in the **Operation** column, and check whether the **nvidia-driver-installer** pod runs on the node. If the **nvidia-driver-installer** pod is present and is:

- In the **Running** state: The pod is functioning properly. Proceed to **Step 2** to verify whether the driver was installed.
- Not in the **Running** state for an extended period: Check the pod events for any abnormalities and troubleshoot based on the reported error information. If the fault persists, **submit a service ticket** and contact technical support.

The name of the **nvidia-driver-installer** pod varies depending on the OS. The details are listed in the table below.

Table 4-6 Names of the nvidia-driver-installer pod

OS	Pod Name
Huawei Cloud EulerOS 2.0	hce20-nvidia-driver-installer
Ubuntu	ubuntu22-nvidia-driver-installer
Others	nvidia-driver-installer

Step 2 Check whether the GPU driver has been installed.

1. In the node list, click the name of the target node. In the dialog box displayed, click **OK**. On the node details page, click **Remote Login** in the upper right corner.

2. Check the driver installation directory.

   a. Check whether the directory exists. If it is present, run the below command to go to the driver installation directory. If it is not present, skip this step and go to **Step 3** to check whether there is an error during the driver installation.

   ```
   cd <Driver installation directory>
   ```

   The driver installation directory varies depending on the CCE AI Suite (NVIDIA GPU) add-on version. The details are as follows:

   - If the CCE AI Suite (NVIDIA GPU) add-on version is later than 2.0.0, the driver installation directory is **/usr/local/nvidia**.
   - If the CCE AI Suite (NVIDIA GPU) add-on version is earlier than 2.0.0, the driver installation directory is **/opt/cloud/cce/nvidia**.

   b. Run the following command in the driver installation directory to view all files in the directory:

   ```
   ls -l
   ```

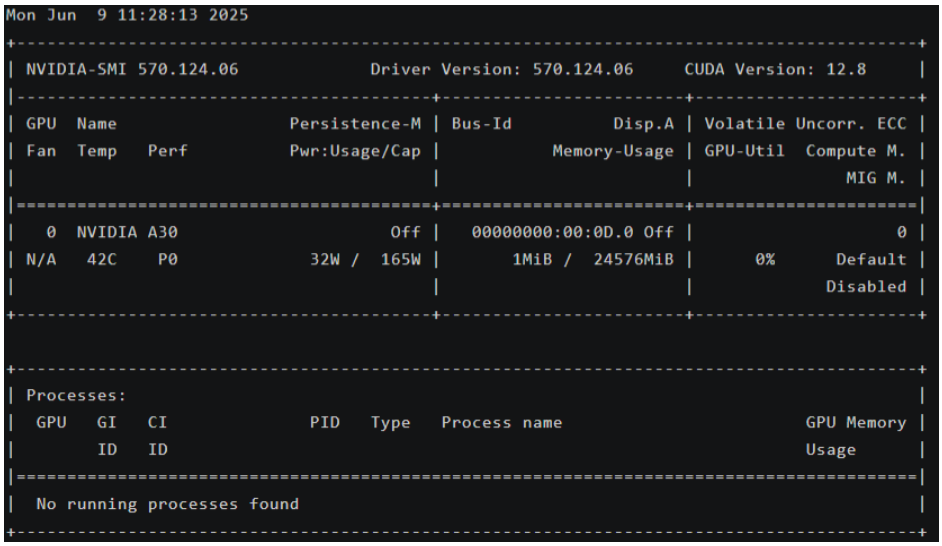
   The figure below shows a typical file directory. **nvidia.run** is the driver installation file. **nvidia-installer.log** is the installation logs generated by the NVIDIA driver. **nvidia-uninstall.log**, if present, is the corresponding uninstallation logs, though it may not always appear in the directory. **If any files are missing, except for nvidia-uninstall.log, go to Step 3 to check whether there is an error during the driver installation.**

   ```
   total 367012
   drwxr-xr-x 2 root root      4096 Jun  5 19:34 bin
   drwxr-xr-x 2 root root      4096 Jun  5 19:34 bin-mount
   drwxr-xr-x 3 root root      4096 Jun  5 19:33 bin-workdir
   drwxr-xr-x 2 root root      4096 Jun  5 19:34 drivers
   drwxr-xr-x 3 root root      4096 Jun  5 19:33 drivers-workdir
   drwxr-xr-x 4 root root      4096 Jun  5 19:34 lib64
   drwxr-xr-x 3 root root      4096 Jun  5 19:33 lib64-workdir
   -rw-r----- 1 root root     46236 Jun  5 19:34 nvidia-installer.log
   -rw-r----- 1 root root 375725620 Jun  5 19:33 nvidia.run
   -rw-r----- 1 root root       887 Jun  5 19:34 nvidia-uninstall.log
   drwxr-xr-x 4 root root      4096 Jun  5 19:05 share
   drwxr-xr-x 2 root root      4096 Jun  6 17:08 xgpu-kmod
   ```

   c. Run the below command to go to the **bin** directory of NVIDIA and check whether **nvidia-smi** is functioning properly. **If the add-on version is earlier than 2.0.0, replace the path with opt/cloud/cce/nvidia/bin.**

   ```
   cd /usr/local/nvidia/bin
   ./nvidia-smi
   ```

   If information similar to that shown in the figure below is not displayed, go to **Step 3** to check whether there is an error during the driver installation.



Step 3 View the node driver installation logs to check whether there is an error during the driver installation.

Run the below command to view the logs of the **nvidia-driver-installer** pod. **If the add-on version is earlier than 2.0.0, replace the path with /opt/cloud/cce/nvidia/nvidia-installer.log.**

```
cat /usr/local/nvidia/nvidia-installer.log
```

If the command output contains the below information, the driver installation completed without error. Otherwise, an error occurred during the installation. If there is an error, **submit a service ticket** and contact technical support.

```
...
> Installation of the NVIDIA Accelerated Graphics Driver for xxx (version: x.x.x) is now complete.
```

----End

Handling a device-plugin Fault

In a CCE cluster, **device-plugin** is responsible for reporting hardware resource statuses. In GPU scenarios, **nvidia-gpu-device-plugin** in the **kube-system** namespace reports the available GPU resources on each node. If the reported GPU resources appear incorrect or if device mounting issues occur, it is advised to first check **device-plugin** for potential anomalies.

Run the following command to **check the device-plugin status**:

```
kubectl get po -A -owide|grep nvidia
```

- If the **device-plugin** pod is not in the **Running** state, **submit a service ticket** and contact technical support.
- If the **device-plugin** pod is in the **Running** state, run the following command to check its logs for errors:

  ```
  kubectl logs -n kube-system nvidia-gpu-device-plugin-9xmhr
  ```

  If "gpu driver wasn't ready. will re-check" is displayed in the command output, go to **Step 2** and check whether the **/usr/local/nvidia/bin/nvidia-smi** or **/opt/cloud/cce/nvidia/bin/nvidia-smi** file exists in the driver installation directory.

  ```
  ...
  I0527 11:29:06.420714 3336959 nvidia_gpu.go:76] device-plugin started
  ```



```
I0527 11:29:06.521884 3336959 nodeinformer.go:124] "nodeInformer started"
I0527 11:29:06.521964 3336959 nvidia_gpu.go:262] "gpu driver wasn't ready. will re-check in %s"
5s="(MISSING)"
I0527 11:29:11.524882 3336959 nvidia_gpu.go:262] "gpu driver wasn't ready. will re-check in %s"
5s="(MISSING)"
...
```

Handling a GPU Fault

Rectify the fault by referring to **GPU Fault Handling**.

4.3.19 What Can I Do If Certain Alarms Are Displayed in the GPU Node Events After the CCE AI Suite (NVIDIA GPU) Add-on Is Upgraded?

Symptom

After the CCE AI Suite (NVIDIA GPU) add-on is upgraded, the following alarms are displayed when you view the GPU node events:

- **Alarm 1**

  Event name: **XGPUKmodNeedUpgrade**

  Kubernetes event: "GPU serverid: xxx, info: XGPU kmod on node xx.xx.xx.xx needs upgrade"
- **Alarm 2**

  Event name: **XGPUKmodAbnormal**

  Kubernetes event: "XGPU kmod on node %s is abnormal"

Possible Cause

- **Alarm 1**: Before the CCE AI Suite (NVIDIA GPU) add-on is upgraded, the GPU virtualization workloads on the GPU nodes were not drained beforehand. As a result, the xGPU kmod upgrade was skipped. This caused a version mismatch between the xGPU kmod and the upgraded add-on.
- **Alarm 2**: The xGPU kmod upgrade failed during the CCE AI Suite (NVIDIA GPU) add-on upgrade.

These alarms do not impact existing services, but they may prevent new features or bug fixes introduced in the upgraded add-on from taking effect. It is advised to address these alarms promptly to ensure full add-on functionality.

Solution

Drain GPU virtualization workloads on the GPU nodes one by one and restart nvidia-gpu-device-plugin on each related node. For details, see **How Can I Drain a GPU Node After Upgrading or Rolling Back the CCE AI Suite (NVIDIA GPU) Add-on?**

4.4 Specification Change

4.4.1 How Do I Change the Node Specifications in a CCE Cluster?

Notes and Constraints

- After the specifications of a node in a node pool are modified on the ECS console, some node pool scaling issues occur. For details, see **What Are the Impacts of Changing the Flavor of a Node in a CCE Node Pool?**
- CCE Turbo cluster nodes of certain specifications can be created only on the CCE console and cannot be modified on the ECS console. You can call the ECS API to modify the specifications. For details, see **Modifying the Specifications of an ECS**.

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 name to access the cluster console. 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.

4.4.2 What Are the Impacts of Changing the Flavor of a Node in a CCE Node Pool?

Context

After you change the flavor of a node in a CCE node pool on the ECS console and then synchronize the ECS status on the CCE console, the node flavor no longer matches the configurations in the node pool.

Impact

When you change the node flavor, it also changes the node parameters such as CPU, memory, and ENI quota (available IP addresses). This can cause the auto scaling settings of the node pool where the node is located to not function as expected.

Assume that the CPU and memory of a node are increased from 2 vCPUs and 4 GiB of memory to 4 vCPUs and 8 GiB of memory.

- During node pool scale-out, the total number of resources in the node pool may exceed the upper limit of the CPU or memory. Expanding a node pool involves calculating resources based on the node template. However, changing the node flavor on the ECS console can cause inconsistencies with the configurations in the node pool, resulting in inaccurate CPU and memory usage for the cluster.
- During node pool scale-in, too many CPU or memory resources may be scaled down. If the node with changed flavor is removed, the actual number of CPUs or memory to be scaled down (4 vCPUs and 8 GiB of memory) may be greater than the expected 2 vCPUs and 4 GiB of memory.

Solution

You are not advised to change the flavor of a node in a node pool. Instead, you can update the node pool and add nodes of other flavors to it. The original node will be removed after services are scheduled to the new nodes.

Step 1 Log in to the CCE console and click the cluster name to access the cluster console. In the navigation pane, choose **Nodes**.

Step 2 Locate the row containing the target node pool and click **Update**.

Step 3 In the **Specifications** area, select new flavors, click **Next: Confirm**, and submit the request.

Step 4 After the node pool configurations are updated, locate the row containing the target node pool and click **Scaling**.

Step 5 In the window that slides out from the right, select the node flavors to be expanded, configure the number of nodes to be added, and click **OK**.

Step 6 Click the **Nodes** tab, locate the row containing the target node, and choose **More** > **Nodal Drainage** to safely evict the service pods on the node.

Step 7 After the service pods are scheduled to a new node, locate the row containing the target node pool, click **Scaling**, select the flavor of the node to be reduced, configure the number of nodes to be removed, and click **OK**.

----End

4.4.3 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

4.4.4 Can I Change the IP Address of a Node in a CCE Cluster?

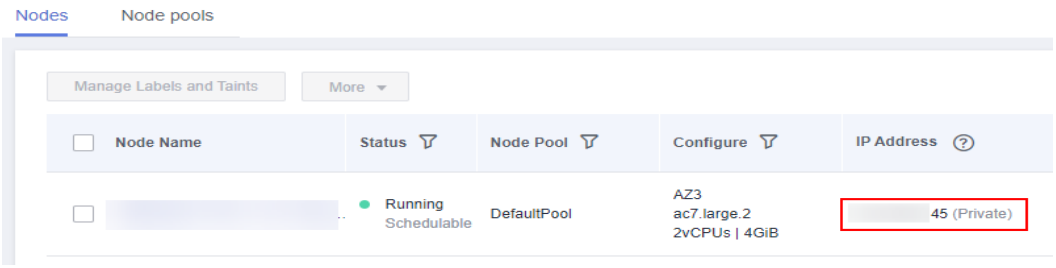
- A private IP address cannot be changed. CCE clusters use the private IP address of a node as the Kubernetes node name, which cannot be changed. Changing the name will make the node unavailable.
- The public IP address of a node can be changed on the ECS console.

How Do I Restore a Node After Its Private IP Is Changed?

After the private IP of a node is changed, the node becomes unavailable. You need to change it back to the original IP address.

Step 1 On the CCE console, view the node details and find the IP address and subnet of the node.

Figure 4-8 Private IP address and subnet of the node



Step 2 Log in to the ECS console, locate and stop the node, go to the node details page, and change the private IP address on the **Network Interfaces** tab page. Note that you need to select the corresponding subnet.

Figure 4-9 Changing the private IP address

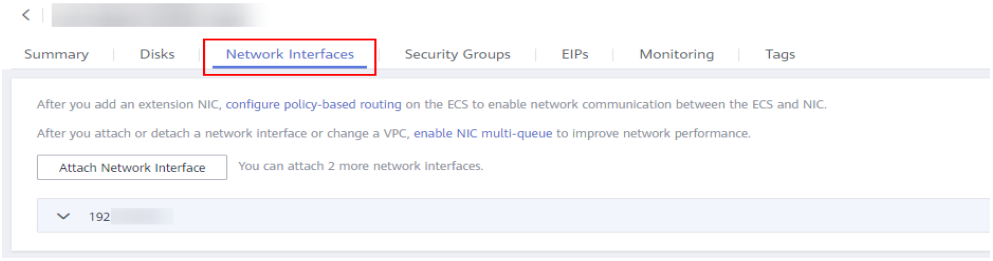
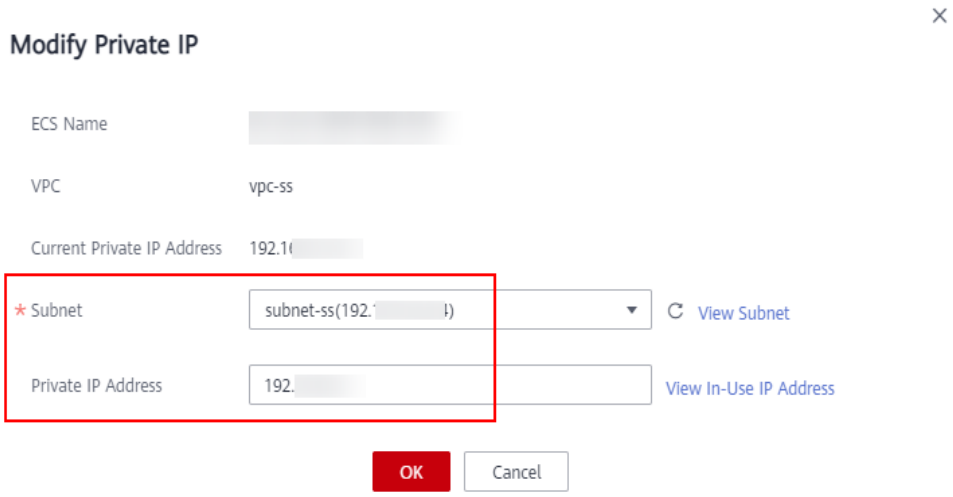


Figure 4-10 Changing the private IP address



Step 3 After the modification is complete, restart the node.

----End

4.5 OSs

4.5.1 What Can I Do If cgroup kmem Leakage Occasionally Occurs When an Application Is Repeatedly Created or Deleted on a Node Running CentOS with an Earlier Kernel Version?

Symptom

When an application is repeatedly created on a node running CentOS 7.6 with a kernel version earlier than 3.10.0-1062.12.1.el7.x86_64, (Such nodes mainly run in clusters 1.17.9.) cgroup kmem leakage occurs. As a result, although there is available memory on the node, new pods still cannot be added to it, and the error message "Cannot allocate memory" displays.

Possible Cause

A temporary memory cgroup is created along with the creation of the application. When the application is deleted, the cgroup (the corresponding **cgroup** directory in **/sys/fs/cgroup/memory**) has already been deleted from the kernel. But in the kernel, cssid is not released, which results in the number of cgroups considered by the kernel is different from the actual number. When the number of residual cgroups exhausts the limit on the node, pods cannot be added to the node.

Solution

- Use the **cgroup.memory=nokmem** parameter globally at the kernel to disable kmem to prevent leakage.
- Clusters of v1.17 are no longer maintained. To resolve this problem, upgrade the cluster to v1.19 or later and reset the OS of the node to the latest version. Ensure that the kernel version is later than 3.10.0-1062.12.1.el7.x86_64.

4.5.2 What Should I Do If There Is a Service Access Failure After a Backend Service Upgrade or a 1-Second Latency When a Service Accesses a CCE Cluster?

Symptom

If the kernel version of a node is earlier than 5.9 and a CCE cluster runs in IPVS forwarding mode, there may be a service access failure after a backend service upgrade or a 1-second latency when a service accesses the CCE cluster. This is caused by a bug in reusing Kubernetes IPVS connections.

IPVS Connection Reuse Parameters

The port reuse policy of IPVS is determined by the kernel parameter **net.ipv4.vs.conn_reuse_mode**.

1. If **net.ipv4.vs.conn_reuse_mode** is set to **0**, IPVS does not reschedule a new connection, but forwards the new connection to the original RS (IPVS backend).
2. If **net.ipv4.vs.conn_reuse_mode** is set to **1**, IPVS reschedules a new connection.

Problems Caused by IPVS Connection Reuse

- **Problem 1**

  If **net.ipv4.vs.conn_reuse_mode** is set to **0**, IPVS does not proactively schedule new connections with port reuse or trigger any connection termination or drop operations. Data packets of the new connections will be directly forwarded to the previously used backend pod. If the backend pod has been deleted or recreated, an exception occurs. However, according to the current implementation logic, in a high-concurrency service access scenario, connection requests for port reuse are continuously forwarded, while kube-proxy did not delete the old ones, resulting in a service access failure.

- **Problem 2**

  If **net.ipv4.vs.conn_reuse_mode** is set to **1** and the source port is the same as that of a previous connection in a high-concurrency scenario, the connection is not reused but rescheduled. According to the processing logic of ip_vs_in(), if **net.ipv4.vs.conntrack** is enabled, the first SYN packet is dropped. As a result, the SYN packet will be retransmitted, leading to a 1-second latency, and the performance deteriorates.

Community Settings and Impact on CCE Clusters

The default value of **net.ipv4.vs.conn_reuse_mode** on a node is **1**. However, the Kubernetes kube-proxy resets this parameter.

Cluster Version	kube-proxy Action	Impact on CCE Cluster
1.17 or earlier	By default, kube-proxy sets **net.ipv4.vs.conn_reuse_mode** to **0**. For details, see **Fix IPVS low throughput issue**.	If CCE clusters of 1.17 or earlier versions use the IPVS service forwarding mode, kube-proxy will set the **net.ipv4.vs.conn_reuse_mode** value of all nodes to **0** by default. This causes **Problem 1**: The RS cannot be removed when the port is reused.

Cluster Version	kube-proxy Action	Impact on CCE Cluster
1.19 or later	kube-proxy sets the value of **net.ipv4.vs.conn_reuse_mode** based on the kernel version. For details, see **ipvs: only attempt setting of sysctlconnreuse on supported kernels**. • If the kernel version is later than 4.1, kube-proxy will set **net.ipv4.vs.conn_reuse_mode** to **0**. • In other cases, the default value **1** will be retained. **NOTE** This issue has been resolved in Linux kernel 5.9. Since Kubernetes 1.22, kube-proxy does not modify the **net.ipv4.vs.conn_reuse_mode** parameter of nodes that use the kernel 5.9 or later. For details, see **Don't set sysctl net.ipv4.vs.conn_reuse_mode for kernels >=5.9**.	If the IPVS service forwarding mode is used in CCE clusters of 1.19.16-r0 or later, the value of **net.ipv4.vs.conn_reuse_mode** varies with the kernel versions of node OSs. • For a node running EulerOS 2.5 or CentOS 7.6, the kernel version is earlier than 4.1. The value of **net.ipv4.vs.conn_reuse_mode** is **1**. This results in **Problem 2**: There is a 1-second latency in the high-concurrency scenarios. • For a node running Ubuntu 18.04, the kernel version is later than 4.1. kube-proxy will set **net.ipv4.vs.conn_reuse_mode** to **0**. This causes **Problem 1**: The RS cannot be removed when the port is reused. • For a node running EulerOS 2.9, the kernel version is too early. kube-proxy will set **net.ipv4.vs.conn_reuse_mode** to **0**. This results in **Problem 1**. To resolve this problem, upgrade the kernel version. For details, see **Rectification Plan**. • For a node running Huawei Cloud EulerOS 2.0 or Ubuntu 22.04, the kernel version is later than 5.9. The problem has been resolved.

Suggestions

Evaluate the impact of these problems. If they affect your services, take the following measures:

1. Use an OS that is not affected by the preceding issues, for example, Huawei Cloud EulerOS 2.0 or Ubuntu 22.04. The newly created nodes which run EulerOS 2.9 are not affected by the preceding issues. Upgrade the earlier kernel versions used by existing nodes to the fixed version. For details, see **Rectification Plan**.
2. Use a cluster whose forwarding mode is iptables.

Rectification Plan

If you use a node running EulerOS 2.9, check whether the kernel version meets the requirements. If the kernel version of the node is too early, reset the node or create a new one.

The following kernel versions are recommended:

- x86: 4.18.0-147.5.1.6.h686.eulerosv2r9.x86_64
- Arm: 4.19.90-vhulk2103.1.0.h584.eulerosv2r9.aarch64

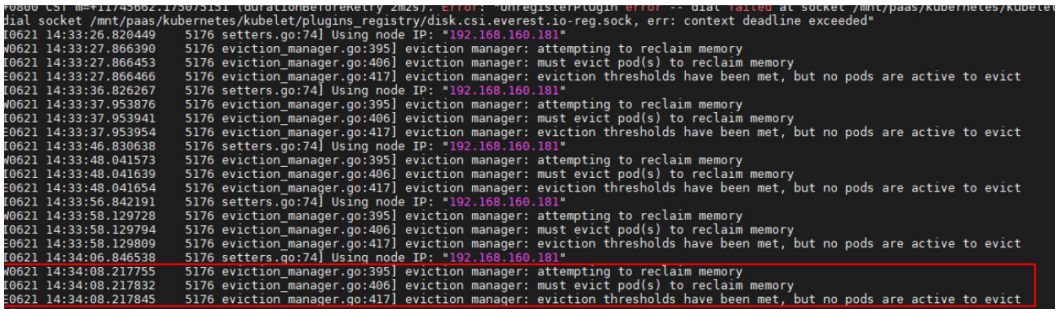
Kubernetes community issue: **https://github.com/kubernetes/kubernetes/issues/81775**

4.5.3 Why Are Pods Evicted by kubelet Due to Abnormal cgroup Statistics?

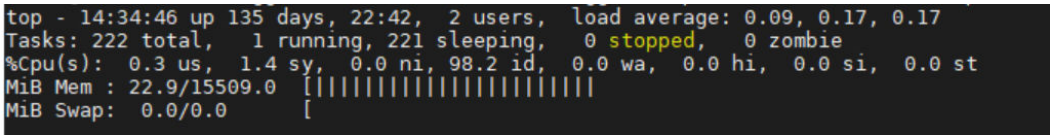
Symptom

On an Arm node, pods are evicted by kubelet due to the abnormal cgroup statistics. As a result, the node runs abnormally.

kubelet keeps evicting pods. After all containers are killed, kubelet still considers that the memory is insufficient.



The resource usage is normal.



The value of **usage_in_bytes** of cgroup in the **/sys/fs/cgroup/memory** directory is abnormal.

```
# cd /sys/fs/cgroup/memory
# cat memory.usage_in_bytes
17618837504
```

Possible Cause

On an Arm node, the kernel of EulerOS 2.8 and 2.9 has a bug, which causes kubelet to evict pods and results in service unavailability.

NOTE

This issue has been resolved in the following versions:

- EulerOS 2.8: kernel-4.19.36-vhulk1907.1.0.h1252.eulerosv2r8.aarch64
- EulerOS 2.9: kernel-4.19.90-vhulk2103.1.0.h819.eulerosv2r9.aarch64

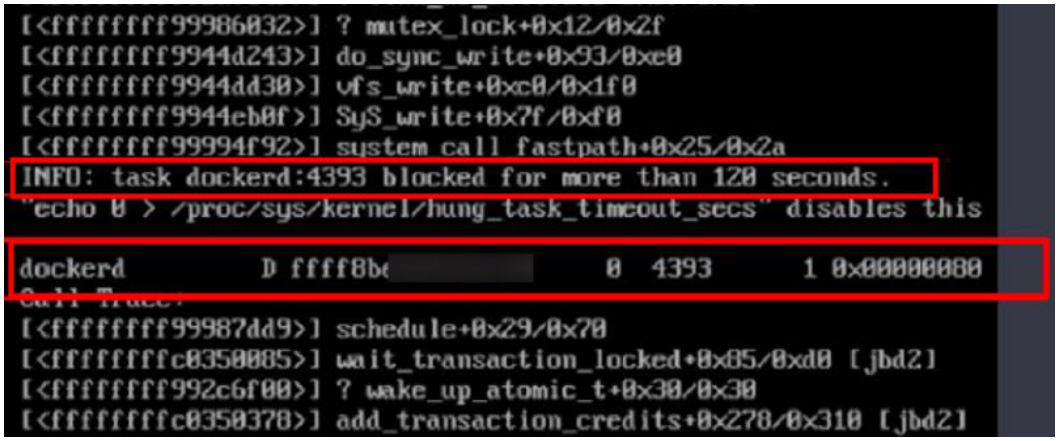
Solution

- If your cluster version is 1.19.16-r0, 1.21.7-r0, 1.23.5-r0, 1.25.1-r0, or later, reset the OS of the node to the latest version.
- If your cluster version does not meet the requirements, upgrade the cluster to the specified version and then reset the node OS to the latest version.

4.5.4 When Container OOM Occurs on the CentOS Node with an Earlier Kernel Version, the Ext4 File System Is Occasionally Suspended

Symptom

If the kernel version of a CentOS 7.6 node is earlier than 3.10.0-1160.66.1.el7.x86_64 and OOM occurs on containers on the node, all containers on the node may fail to be accessed, and processes such as Docker and jdb are in the D state. The fault is rectified after the node is restarted.



Possible Cause

When the memory usage of a service container exceeds its memory limit, cgroup OOM is triggered and the container is terminated by the system kernel. Container cgroup OOM occasionally triggers ext4 file system suspension on CentOS 7, and ext4/jbd2 is permanently suspended due to deadlock. All tasks that perform I/O operation on the file system are affected.

Solution

- Temporary solution: Restart the node to temporarily rectify the fault.
- Long-term solution:
  - If your cluster version is 1.19.16-r0, 1.21.7-r0, 1.23.5-r0, 1.25.1-r0, or later, reset the OS of the node to the latest version.

- If your cluster version does not meet the requirements, upgrade the cluster to the specified version and then reset the node OS to the latest version.

4.5.5 What Should I Do If a DNS Resolution Failure Occurs Due to a Defect in IPVS?

Symptom

In IPVS forwarding mode used in a CCE cluster, packet loss may occur after CoreDNS is upgraded on the node. This results in a Domain Name System (DNS) resolution failure.

Possible Cause

This problem is caused by a defect in IPVS. The community has fixed it in IPVS v5.9-rc1. For details, see **ipvs: queue delayed work to expire no destination connections if expire_nodest_conn=1**

Nodes running Ubuntu 22.04 or Huawei Cloud EulerOS 2.0 are not affected by this problem. Nodes running CentOS, Ubuntu18.04, EulerOS 2.5, EulerOS 2.9 (with earlier kernel version), or Huawei Cloud EulerOS 1.1 are affected by this problem.

Solution

- The impact of the IPVS packet loss can be reduced by using NodeLocal DNSCache. For details, see .
- Use unaffected OSs, such as Huawei Cloud EulerOS 2.0 and Ubuntu 22.04.
- If the OS of your node is EulerOS 2.9, check whether the kernel version of the node meets the following requirements (If the kernel version of the node is too early, reset the node to rectify the fault. If the kernel version of the node meets the requirements, the node is not affected by this issue and no further action is required):
  - x86 node: The kernel version is 4.18.0-147.5.1.6.h998.eulerosv2r9.x86_64 or later.
  - Arm node: The kernel version is 4.19.90-vhulk2103.1.0.h990.eulerosv2r9.aarch64 or later.

4.5.6 What Should I Do If the Number of ARP Entries Exceeds the Upper Limit?

Symptom

The ARP cache exceeds the upper limit, resulting in the abnormal inter-container access, for example, the coredns DNS resolution failure.

Possible Cause

The number of ARP entries cached in the containers on the node exceeds the upper limit.

Fault Locating

- If the OS kernel of a node is later than 4.3, **neighbor table overflow** will display in the dmsg log. For details, see **GitHub**.

```
# dmesg -T
[Tue May 30 18:35:55 2023] neighbour: arp_cache: neighbor table overflow!
[Tue May 30 18:35:55 2023] neighbour: arp_cache: neighbor table overflow!
[Tue May 30 18:35:55 2023] neighbour: arp_cache: neighbor table overflow!
[Tue May 30 18:35:55 2023] neighbour: arp_cache: neighbor table overflow!
[Tue May 30 18:35:55 2023] neighbour: arp_cache: neighbor table overflow!
[Tue May 30 18:35:55 2023] neighbour: arp_cache: neighbor table overflow!
[Tue May 30 18:35:55 2023] neighbour: arp_cache: neighbor table overflow!
[Tue May 30 18:35:55 2023] neighbour: arp_cache: neighbor table overflow!
[Tue May 30 18:35:55 2023] neighbour: arp_cache: neighbor table overflow!
[Tue May 30 18:35:58 2023] print_fib4_table_status: 7 callbacks suppressed
[Tue May 30 18:35:59 2023] print_fib4_table_status: 23 callbacks suppressed
[Tue May 30 18:36:00 2023] print_fib4_table_status: 16 callbacks suppressed
[Tue May 30 18:36:03 2023] print_fib4_table_status: 7 callbacks suppressed
[Tue May 30 18:36:04 2023] print_fib4_table_status: 17 callbacks suppressed
[Tue May 30 18:37:38 2023] net_ratelimit: 7966 callbacks suppressed
[Tue May 30 18:37:38 2023] neighbour: arp_cache: neighbor table overflow!
```

- If the kernel version of the node OS is earlier than 4.3, **neighbor table overflow** will not display. If **callbacks suppressed** is displayed, the number of ARP entries may exceed the upper limit.

```
[Wed Jun 14 21:08:58 2023] net_ratelimit: 198 callbacks suppressed
[Wed Jun 14 21:09:05 2023] net_ratelimit: 11 callbacks suppressed
[Wed Jun 14 21:12:35 2023] nr_pdflush_threads exported in /proc is scheduled for removal
[Wed Jun 14 21:39:03 2023] net_ratelimit: 337 callbacks suppressed
[Wed Jun 14 21:39:10 2023] net_ratelimit: 236 callbacks suppressed
[Wed Jun 14 22:09:18 2023] net_ratelimit: 53 callbacks suppressed
[Wed Jun 14 22:14:04 2023] net_ratelimit: 266 callbacks suppressed
[Wed Jun 14 22:14:10 2023] net_ratelimit: 350 callbacks suppressed
[Wed Jun 14 22:15:28 2023] net_ratelimit: 81 callbacks suppressed
[Wed Jun 14 22:34:12 2023] net_ratelimit: 178 callbacks suppressed
[Wed Jun 14 22:34:19 2023] net_ratelimit: 18 callbacks suppressed
[Wed Jun 14 22:39:17 2023] net_ratelimit: 95 callbacks suppressed
[Wed Jun 14 22:44:24 2023] net_ratelimit: 135 callbacks suppressed
[Wed Jun 14 22:51:43 2023] ip_finish_output2: No header cache and no neighbour!
```

Solution

The maximum number of non-permanent entries allowed by a node is determined by the **net.ipv4.neigh.default.gc_thresh3** parameter of the kernel. This parameter is not isolated by namespace. The node and containers running on the node share the ARP table size. In containers, set this parameter to **163790**.

How to calculate the kernel parameter

- CCE Turbo clusters and clusters using the container tunnel networks

  net.ipv4.neigh.default.gc_thresh3 = Number of containers on a single node x Number of available IP addresses on the container subnet (If there are multiple container subnets in a CCE Turbo cluster, use the maximum number of available IP addresses on a container subnets and the maximum number of containers that can be deployed on a single node.)

  For example, if a container subnet is **192.168.0.1/20**, there will be 4,096 IP addresses available and there can be at most 35 containers deployed on a single node, so you can set **net.ipv4.neigh.default.gc_thresh3** to **143360** (4096 x 35).

- Clusters using the VPC networks

  net.ipv4.neigh.default.gc_thresh3 = Number of containers on a single node squared

For example, if the subnet mask of a node is 25, there will be 128 container IP addresses available, so you can set **net.ipv4.neigh.default.gc_thresh3** to **16384** (128 x 128).

NOTE

The preceding formulas are used in extreme scenarios.

1. All containers on a node proactively access all IP addresses in the container CIDR block. For example, a gateway container needs to access all other containers in the same cluster.

2. All available IP addresses in a container CIDR block are used up.

Step 1 In **88-k8s.conf**, change the value of **net.ipv4.neigh.default.gc_thresh3** to **163790**.

```
vi /etc/sysctl.d/88-k8s.conf
```

NOTE

The **net.ipv4.neigh.default.gc_thresh1** and **net.ipv4.neigh.default.gc_thresh2** parameters cannot be modified.

Step 2 Run the following command to reload the configuration file:

```
sysctl -p /etc/sysctl.d/88-k8s.conf
```

Step 3 Check whether the configuration takes effect.

```
sysctl -a | grep gc_thresh3
```

```
[root@x-turbo-readinessgate-08342-r05vt ~]# sysctl -a | grep gc_thresh3
net.ipv4.neigh.default.gc_thresh3 = 163790
net.ipv6.neigh.default.gc_thresh3 = 1024
```

----End

4.5.7 What Should I Do If a VM Is Suspended Due to an EulerOS 2.9 Kernel Defect?

Symptom

There is a small chance of a deadlock occurring on an EulerOS 2.9 node, which is caused by community issues related to scheduling in the kernel. This can lead to the suspension of the VM.

Impact

- x86 kernel version: 4.18.0-147.5.1.6.h1152.eulerosv2r9.x86_64
- Arm kernel version: 4.19.90-vhulk2103.1.0.h1144.eulerosv2r9.aarch64

Possible Cause

The scheduling in the EulerOS 4.18 kernel has issues related to CPU cgroup usage. When CFS bandwidth control is configured and CPU bandwidth control is triggered, it may result in warn-level alarms being generated. This process holds the rq lock for scheduling. This can cause a deadlock with other processes. Specifically, an ABBA deadlock may occur in x86_64 and an AA deadlock in aarch64.

Solution

You can change the value of **kernel.printk** in the configuration file to rectify the fault. The **kernel.printk** parameter controls how kernel log information is exported and the output level.

Step 1 Check the current configurations of **kernel.printk** in the configuration file.

```
grep "kernel.printk" /etc/sysctl.conf
```

In the command output, the value of **kernel.printk** is **7 4 1 7**.

```
[root@ ~]# grep "kernel.printk" /etc/sysctl.conf
kernel.printk=7       4       1       7
```

Step 2 Delete the **kernel.printk** configuration.

```
sed -i '/^kernel.printk/d' /etc/sysctl.conf
```

Step 3 Run the following command to check whether the configuration file is modified.

No command output is displayed.

```
grep "kernel.printk" /etc/sysctl.conf
```

Step 4 Reconfigure **kernel.printk**.

x86_64 version:

1. Run the following command:
   ```
   sysctl -w kernel.printk="4 4 1 7"
   ```
   ```
   [root@localhost ~]# sysctl -w kernel.printk="4       4       1       7"
   kernel.printk = 4       4       1       7
   ```
2. Run the following command to check whether the modification is successful:
   ```
   sysctl -a | grep kernel.printk
   ```
   Ensure that the value of **kernel.printk** is **4 4 1 7**.
   ```
   [root@localhost ~]# sysctl -a |grep kernel.printk
   kernel.printk = 4       4       1       7
   ```

Arm version:

1. Run the following command:
   ```
   sysctl -w kernel.printk="1 4 1 7"
   ```
   ```
   [root@ ~]# sysctl -w kernel.printk="1       4       1       7"
   kernel.printk = 1       4       1       7
   ```
2. Run the following command to check whether the modification is successful:
   ```
   sysctl -a | grep kernel.printk
   ```
   Ensure that the value of **kernel.printk** is **1 4 1 7**.
   ```
   [root@ ~]# sysctl -a | grep kernel.printk
   kernel.printk = 1       4       1       7
   ```

----End

5 Node Pool

5.1 What Should I Do If a Node Pool Is Abnormal?

Fault Locating

Locate the fault based on the status of the abnormal node pool, as shown in Table 5-1.

Table 5-1 Node pool exceptions

Abnormal Node Pool Status	Description	Solution
Error	The node pool cannot be deleted.	Delete the node pool again. If the node pool still cannot be deleted, submit a service ticket and delete the node pool.
QuotaInsufficient	The node pool cannot be scaled out due to insufficient quota.	Submit a service ticket and increase the quota.
SoldOut	The underlying resources are insufficient.	Update the node pool configuration and select other available resources.

Abnormal Node Pool Status	Description	Solution
ConfigurationInvalid	The ECS group does not exist (**ServerGroupNotExists**). The ECS group to which the node pool belongs is not present. This may be because you manually deleted the ECS group.	1. Log in to the CCE console. In the navigation pane, choose **Nodes**. In the right pane, click the **Node Pools** tab and click the name of the target node pool. Click the **Overview** tab, click **Expand**, and check the ECS group to which the node pool belongs. 2. Log in to the ECS console. In the navigation pane, choose **Elastic Cloud Server** > **ECS Group** and see if the target ECS group is present. 3. If the ECS group is not present, log in to the CCE console. In the navigation pane, choose **Nodes**, click the **Node Pools** tab, locate the row containing the target node pool, and click **Update**. In the **Advanced Settings** area, unbind or change the ECS group.
InstanceAboveServerGroup	The number of ECSs in the ECS group exceeds the upper limit.	1. Log in to the CCE console. In the navigation pane, choose **Nodes**. In the right pane, click the **Node Pools** tab, locate the row containing the target node pool, and click the node pool name. On the page displayed, click the **Overview** tab, click **Expand**, and check the ECS group to which the node pool belongs. 2. Log in to the ECS console, choose **Elastic Cloud Server** > **ECS Group** in the navigation pane, and check the quota of the target ECS group. 3. If the quota is insufficient, log in to the CCE console. In the navigation pane, choose **Nodes**, click the **Node Pools** tab, locate the row containing the target node pool, and click **Update**. In the **Advanced Settings** area, unbind or change the ECS group.
InvalidCapacityReservation	A capacity reservation error occurs.	Update the node pool and select other capacity reservation specifications.

Abnormal Node Pool Status	Description	Solution
MaxNodeCountReached	The number of expected nodes in the current node pool or scaling group has reached the maximum allowed limit.	• Manually add nodes to the node pool or scaling group as needed. If the expected number of nodes in a node pool or scaling group reaches the maximum allowed limit, auto scaling will no longer be triggered for that node pool or scaling group. However, this does not affect manual scale-out operations. • Choose **Nodes** in the navigation pane, click the **Node Pools** tab in the right pane, locate the row containing the target node pool, click **Auto Scaling**, and change the value of **Nodes**. Or simply disable auto scaling of the node pool or scaling group.

5.2 What Should I Do If No Node Creation Record Is Displayed When the Node Pool Is Being Scaled Out?

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 a tenant is in arrears.
- 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 tenant is in arrears, renew the account as soon as possible.
- 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.

5.3 What Should I Do If a Node Pool Scale-Out Fails?

Fault Locating

Locate the fault based on the events of the failure to scale out a node pool, as shown in **Table 5-2**.

Table 5-2 Node pool scale-out failure

Event	Possible Cause	Reference
...call fsp to query keypair fail, error code : Ecs.0314, reason is : the keypair *** does not match the user_id ***...	The possible causes are as follows: • The key pair selected for logging in to the node pool has been deleted. • The key pair selected for logging in to the node pool is a private one which cannot be used by the current user to log in to the node pool and create nodes in the node pool.	**Failed to Obtain the Key Pair Used for Logging In to a Node Pool**
{"error": {"message":"encrypted key id [***] is invalid.","code":"Ecs.0912"}}	The possible causes are as follows: • The KMS key ID entered during node pool creation does not exist. • The KMS key ID entered during node pool creation is the key of another user, but the user has not authorized you.	**Invalid KMS Key ID**
Security group [*****] not found	This issue can arise in the following scenarios: • A custom security group is set up for the node pool but gets deleted, so the node pool scale-out fails. • No custom security group is configured for the node pool and the default security group is deleted, so the node pool scale-out fails.	**The Security Group Specified by the Node Pool Deleted**

Event	Possible Cause	Reference
The Enterprise Project ID [*****] not exist	The enterprise project configured for the node pool has been deleted from EPS.	The Enterprise Project Specified by the Node Pool Deleted

Failed to Obtain the Key Pair Used for Logging In to a Node Pool

If a node pool scale-out fails, the event contains **Ecs.0314**. This error code indicates that the key pair used for logging in to the node pool cannot be obtained, which results in the creation failure of a new ECS.

```
...call fsp to query keypair fail, error code : Ecs.0314, reason is : the keypair *** does not match the user_id ***...
```

The possible causes are as follows:

- The key pair selected for logging in to the node pool has been deleted.
- The key pair selected for logging in to the node pool is a private one which cannot be used by the current user to log in to the node pool and create nodes in the node pool.

Solution

- If the scale-out fails due to the first cause, you can create a key pair and then create a node pool which can be logged in to using this key pair.
- If the scale-out fails due to the second cause, only the user who created the private key pair can scale out the node pool. You can use another key pair when creating a new node pool.

Invalid KMS Key ID

When a node pool fails to be expanded, the reported event contains **Ecs.0912**.

```
{"error":{"message":"encrypted key id [***] is invalid.","code":"Ecs.0912"}}
```

The possible causes are as follows:

- The KMS key ID entered during node pool creation does not exist.
- The KMS key ID entered during node pool creation is the key of another user, but the user has not authorized you.

Solution

- If the scale-out fails due to the first cause, ensure that the entered key ID exists.
- If the scale-out fails due to second cause, use the ID of the shared key that has been authorized to you.

The Security Group Specified by the Node Pool Deleted

When a node pool fails to be expanded, the event contains the following information:

```
Security group [*****] not found
```

This issue can arise in the following scenarios:

- Scenarios 1: A custom security group is set up for the node pool but gets deleted, so the node pool scale-out fails.
- Scenarios 2: No custom security group is configured for the node pool and the default security group is deleted, so the node pool scale-out fails.

Solution

- Scenario 1: Update the security group specified by the **customSecurityGroups** field by calling the API for updating a node pool. For details, see **Updating a Specified Node Pool**.
- Scenario 2: Log in to the CCE console and change the **default node security group** on the **Settings** page of the cluster. The new node security group must meet the communication rules of the cluster ports. For details, see **How Can I Configure a Security Group Rule for a Cluster?**

The Enterprise Project Specified by the Node Pool Deleted

When a node pool fails to be expanded, the event contains the following information:

```
The Enterprise Project ID [*****] not exist
```

The enterprise project configured for the node pool has been deleted from EPS.

Solution

Step 1 Log in to the CCE console.

Step 2 Click the cluster name to access the cluster console. Choose **Nodes** in the navigation pane. In the right pane, click the **Node Pools** tab.

Step 3 Locate the row containing the node pool that fails to be scaled out and click **Update**. In the displayed **Update Node Pool** page, configure the parameters.

Step 4 Select another enterprise project for the node pool.

Step 5 After the configuration is complete, click **OK**.

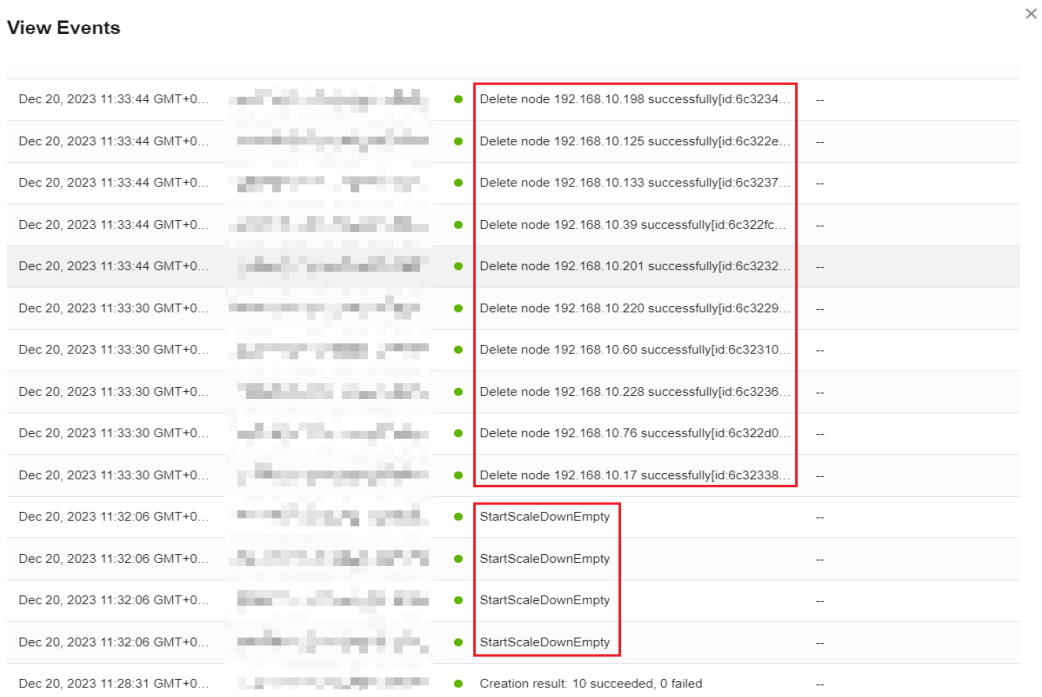
----End

5.4 What Should I Do If Some Kubernetes Events Fail to Display After Nodes Were Added to or Deleted from a Node Pool in Batches?

Symptom

After nodes were scaled in or out in a node pool in batches, some Kubernetes events failed to display.

For instance, if 10 nodes were deleted from a cluster in batches, 10 **Delete node** events are printed by CCE, while only 4 **StartScaleDownEmpty** Kubernetes events were printed.



Possible Cause

Kubernetes limits, aggregates, and counts events before printing to ensure the availability of etcd. Therefore, Kubernetes events are not fully printed, particularly when there are numerous identical events being printed.

This is achieved through the **EventCorrelate** method in Kubernetes source code. For details, see **design proposal** in GitHub.

You do not need to pay attention to this problem, as it is caused by the Kubernetes design mechanism.

5.5 How Do I Modify ECS Configurations When an ECS Can't Be Managed by a Node Pool?

If an ECS cannot be managed by a node pool due to the reasons listed in this section, you can modify the configuration to manage the ECS.

Cause	Solution	Reference
Inconsistent flavors	Change the ECS flavor to that contained in the node pool.	**Modifying the Flavor of an ECS**
Inconsistent VPC and subnet	Change the VPC and subnet where the ECS resides to be the same VPC and subnet as the node pool.	**Changing the VPC and Subnet of an ECS**
Different billing modes	Change the billing mode of the ECS to be the same as that of the node pool.	**Changing the Billing Mode of an ECS**
Different data disk configuration	Change the data disk configuration of the ECS to be the same as that of the node pool.	**Changing Data Disk Configuration of an ECS**
Different enterprise projects	Change the enterprise project of the ECS to be the same as that of the node pool.	**Changing the Enterprise Project of an ECS**
Different ECS groups	Change the ECS group of the ECS to be the same as that of the node pool.	**Changing the ECS Group of an ECS**

Modifying the Flavor of an ECS

NOTE

The flavor of the ECS to be managed must be changed to that contained in the target node pool.

For more operation guides, see **General Operations**.

Step 1 Log in to the ECS console.

Step 2 Click the name of the target ECS. On the page displayed, click **Stop**. After the ECS is stopped, choose **More** > **Modify Specifications** in the **Operation** column.

Step 3 On the **Modify ECS Specifications** page, select the needed flavor and submit the application.

Step 4 Go back to the ECS list page and start the ECS.

----End

Changing the VPC and Subnet of an ECS

NOTE

You need to change the VPC and subnet of the ECS that you want to manage to match those of the target node pool.

For details, see **Changing a VPC**.

Step 1 Log in to the ECS console.

Step 2 Locate the row containing the target ECS and choose **More** > **Manage Network** > **Change VPC** in the **Operation** column.

Step 3 Configure the parameters for changing the VPC.

- **VPC**: Select the target VPC.
- **Subnet**: Select the target subnet.
- **Private IP Address**: Select **Assign new** or **Use existing** as required.

Step 4 Click **OK**.

----End

Changing the Billing Mode of an ECS

NOTE

The billing mode of the ECS to be managed must be the same as that of the target node pool.

From Pay-per-Use to Yearly/Monthly

For details, see **Pay-per-Use to Yearly/Monthly**.

Step 1 Log in to the ECS console.

Step 2 Locate the row containing the target ECS and choose **More** > **Change Billing Mode** in the **Operation** column.

Step 3 Click **OK**. Then you are switched to Billing Center.

Step 4 Select the usage duration, determine whether to enable auto-renewal, confirm the expected expiration date and price, and click **Pay**.

Step 5 Select a payment method and make your payment. Once the order is paid, yearly/monthly billing is applied.

----End

From Yearly/Monthly to Pay-per-Use

For details, see **Yearly/Monthly to Pay-per-Use**.

Step 1 Log in to the ECS console.

Step 2 Locate the row containing the target ECS and choose **More** > **Change to Pay-per-Use** > **Change to Pay-per-Use Immediately** in the **Operation** column.

Step 3 Click **OK**. Then you are switched to Billing Center.

Step 4 Select the resources to be changed to pay-per-use resources following instructions.

Step 5 Confirm the refund information and click **Change to Pay-Per-Use**.

Step 6 Confirm the resources to be changed to pay-per-use resources again and click **OK**.

----End

Changing Data Disk Configuration of an ECS

NOTE

The number, space, and type of data disks of the ECS to be managed must be the same as those of data disks in the node pool.

Data Disk Number

For more operation guides, see **Adding a Disk to an ECS** or **Detaching an EVS Disk from a Running ECS**.

Step 1 Log in to the ECS console.

Step 2 Click the name of the target ECS to access the ECS details page.

Step 3 Click the **Disks** tab.

- If there are fewer data disks on the node to be managed than the number of data disks configured for the target node pool, you need to add more disks.

  Click **Add Disk** and configure parameters for the new disk. For details about how to configure an EVS disk, see **Purchasing an EVS Disk**.

  NOTICE

  The specifications and space of the new disk must be the same as those configured for the target node pool. You need also select **SCSI** for **Advanced Settings**.

- If there are more data disks on the node to be managed than the number of data disks configured for the target node pool, you need to remove some disks.

  Click **Detach** on the right of the EVS disk to be removed.

----End

Data Disk Space

For more operation guides, see **Expanding the Capacity of an EVS Disk**.

Step 1 Log in to the ECS console.

Step 2 Click the name of the target ECS to access the ECS details page.

Step 3 Click the **Disks** tab and click **Expand Capacity** on the right of the EVS disk to be expanded.

Step 4 Configure **New Capacity** following instructions.

Step 5 Click **Next** and submit the order following instructions.

----End

Data Disk Type

For more operation guides, see **Changing the EVS Disk Type (OBT)**.

Step 1 Log in to the ECS console.

Step 2 Click the name of the target ECS to access the ECS details page.

Step 3 Click the **Disks** tab and click **Modify Specifications** on the right of the EVS disk to be expanded.

Step 4 Configure **Disk Type** following instructions.

Step 5 Click **Submit**.

----End

Changing the Enterprise Project of an ECS

NOTE

The enterprise project of the ECS to be managed must be the same as that of the target node pool.

For more operation guides, see **Removing Resources from an Enterprise Project**.

Step 1 Log in to the Huawei Cloud management console.

Step 2 Choose **Enterprise** > **Project Management** in the upper right corner of the page.

Step 3 On the page displayed, select an enterprise project and click **View Resource** in the **Operation** column.

Step 4 Select the resources to be removed and click **Remove**.

Step 5 Select **ECSs and ECS associated resources**. Resources associated with the ECS will be automatically removed simultaneously.

Step 6 Select the target enterprise project and click **OK**.

----End

Changing the ECS Group of an ECS

NOTE

The ECS group of the ECS to be managed must be the same as that of the target node pool.

For more operation guides, see **Managing ECS Groups**.

Step 1 Log in to the ECS console.

Step 2 In the navigation pane, choose **Elastic Cloud Server** > **ECS Group**.

Step 3 Locate the row containing the target ECS group and click **Add ECS** in the **Operation** column.

Step 4 In the dialog box displayed, select the ECS to be added.

Step 5 Click **OK** to add the ECS to the ECS group.

----End

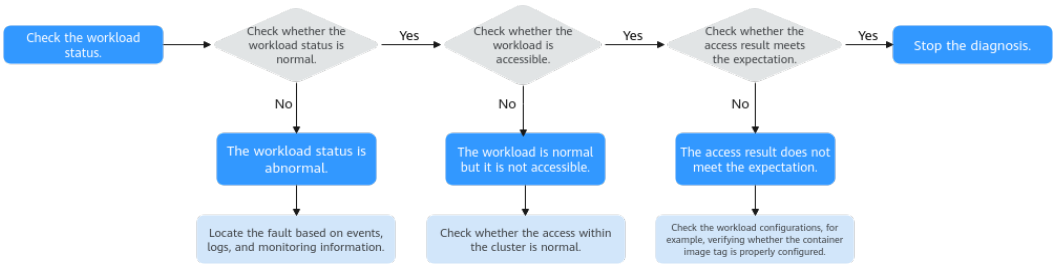
6 Workload

6.1 Workload Exception Troubleshooting

6.1.1 How Can I Locate the Root Cause If a Workload Is Abnormal?

If a workload is abnormal, you can check the pod events first to locate the fault and then rectify the fault.

Fault Locating



To locate the fault of an abnormal workload, take the following steps:

Step 1 Check whether the workload pod is running properly.

1. Log in to the CCE console.
2. Click the cluster name to access the cluster console. In the navigation pane, choose **Workloads**.
3. In the upper left corner of the page, select the namespace, locate the target workload, and view its status.
   - If the workload is not ready, you can view pod events to determine the cause. For details, see **Viewing Pod Events**. You can find the solution to the issue based on the events by referring to **Common Pod Issues**.
   - If the workload is processing, wait patiently.
   - If the workload is running, but it is not accessible, check whether intra-cluster access is normal.

Step 2 Check whether access within the cluster is normal.

Log in to the CCE console or use kubectl to obtain the pod IP address. Then, log in to the node or the pod and run **curl** or use other methods to manually call the APIs and check whether the intra-cluster communication is normal.

If *{Container IP address}*:*{Port number}* is not accessible, log in to the service container, and attempt to access **127.0.0.1**:*{Port number}*.

For details about how to log in to a container, see **Logging In to a Container**.

Step 3 Check whether the expected results are displayed.

If the workload is accessible within the cluster but the expected results are not shown, check the workload configurations, such as verifying if the image tag and environment variables are correctly configured.

----End

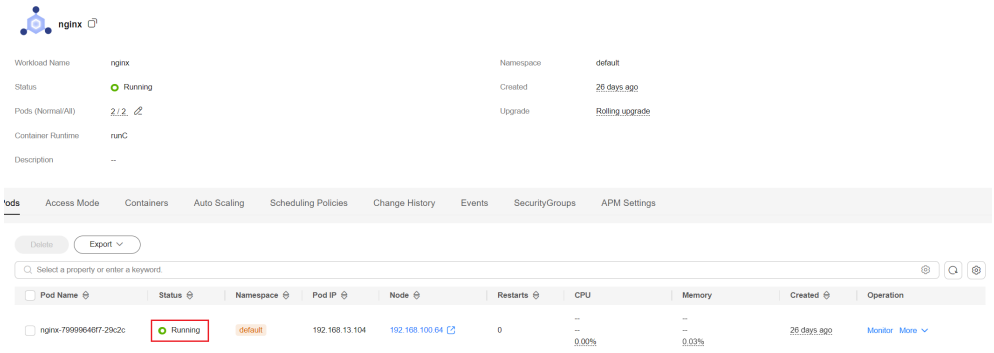
Common Pod Issues

Status	Description	Reference
Pending	The pod scheduling failed.	**What Should I Do If the Scheduling of a Pod Fails?**
Pending	A storage volume fails to be mounted to a pod.	**What Should I Do If a Storage Volume Cannot Be Mounted or the Mounting Times Out?**
Pending	The storage volume mounting failed.	**What Should I Do If a Workload Exception Occurs Due to a Storage Volume Mount Failure?**
FailedPullImage ImagePullBackOff	The image pull failed. The image failed to be pulled again.	**What Should I Do If a Pod Fails to Pull the Image?**
CreateContainerError CrashLoopBackOff	The container startup failed. The container failed to restart.	**What Should I Do If a Pod Startup Fails?**
Evicted	A pod is in the **Evicted** state, and the pod keeps being evicted.	**What Should I Do If a Pod Fails to Be Evicted?**
Creating	A pod is in the **Creating** state.	**What Should I Do If a Workload Remains in the Creating State?**

Status	Description	Reference
Terminating	A pod is in the **Terminating** state.	**What Should I Do If a Pod Remains in the Terminating State?**
Stopped	A pod is in the **Stopped** state.	**What Should I Do If a Workload Is Stopped Caused by Pod Deletion?**

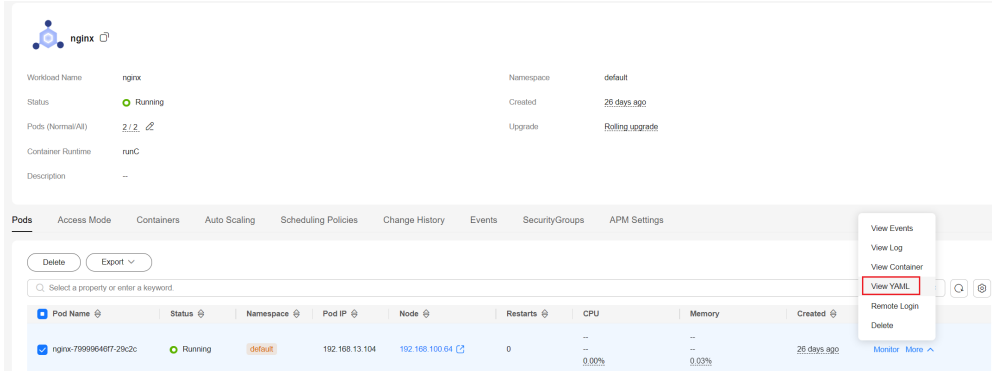
Checking the Pod Status

1. Log in to the CCE console.
2. Click the cluster name to access the cluster console. In the navigation pane, choose **Workloads**.
3. Click the name of the target workload and check the pod statuses.



Checking Pod Configurations

1. Log in to the CCE console.
2. Click the cluster name to access the cluster console. In the navigation pane, choose **Workloads**.
3. Click the name of the target workload. In the workload pod list, locate the row containing the target pod and choose **More** > **View YAML** in the **Operation** column.

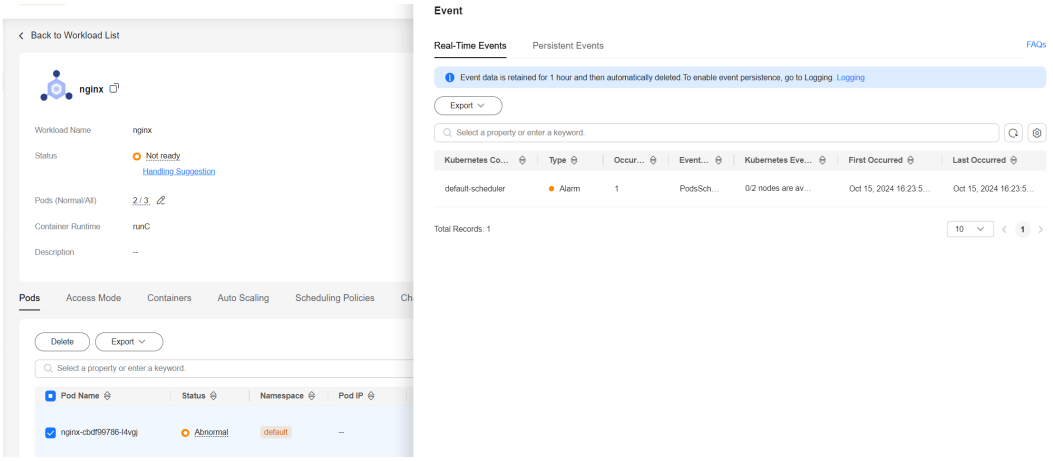


Viewing Pod Events

Method 1

On the CCE console, click the workload name to go to the workload details page, locate the row containing the abnormal pod, and choose **More** > **View Events** in the **Operation** column.

Figure 6-1 Viewing pod events



Method 2

Use the kubectl command:

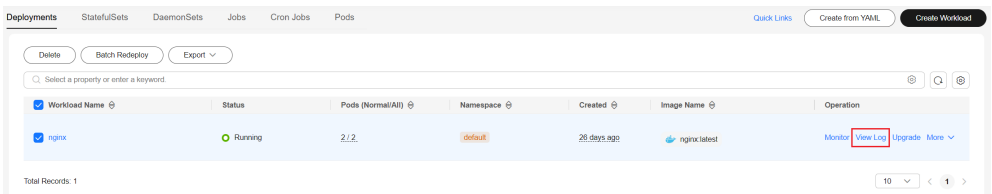
```
kubectl describe pod {pod-name}
```

Information similar to the following is displayed:

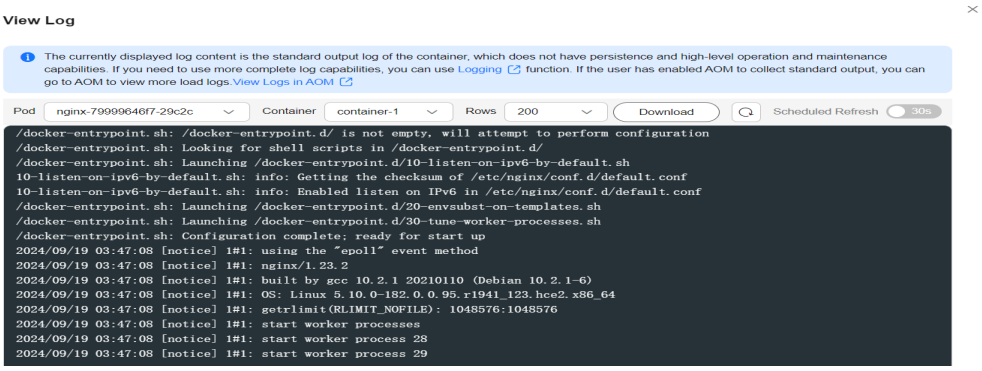
```
...
Events:
  Type     Reason            Age   From               Message
  ----     ------            ----  ----               -------
  Warning  FailedScheduling  49s   default-scheduler  0/2 nodes are available: 2 Insufficient cpu.
  Warning  FailedScheduling  49s   default-scheduler  0/2 nodes are available: 2 Insufficient cpu.
```

Viewing Container Logs

1. Log in to the CCE console.
2. Click the cluster name to access the cluster console. In the navigation pane, choose **Workloads**.
3. Locate the row containing the target workload and click **View Log** in the **Operation** column.



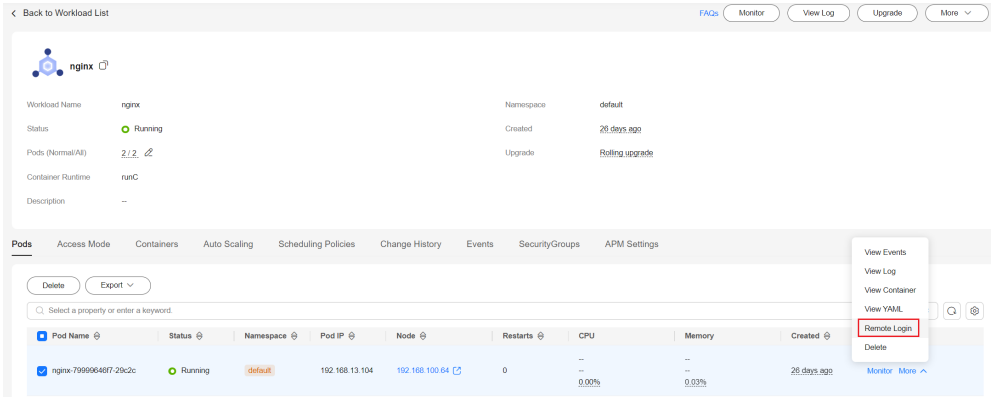
4. Switch between pods and containers above the logs.



Logging In to a Container

You can log in to a container using kubectl. For details, see **Logging In to a Container**.

1. Log in to the CCE console.
2. Click the cluster name to access the cluster console. In the navigation pane, choose **Workloads**.
3. Click the name of the target workload. In the workload pod list, locate the row containing the target pod and choose **More** > **Remote Login** in the **Operation** column.



6.1.2 What Should I Do If the Scheduling of a Pod Fails?

Fault Locating

If a pod is in the **Pending** state and the events contain the information that indicates a pod scheduling failure, you can locate the cause based on the events. For details about how to view events, see **How Can I Locate the Root Cause If a Workload Is Abnormal?**

Troubleshooting

Determine the cause based on the events, as listed in **Table 6-1**.

Table 6-1 Events related to a pod scheduling failure

Event	Cause and Solution
no nodes available to schedule pods.	There are not any available nodes in the cluster. **Check Item 1: Whether a Node Is Available in the Cluster**
0/2 nodes are available: 2 Insufficient cpu. 0/2 nodes are available: 2 Insufficient memory.	The resources (CPU and memory) on the node are insufficient. **Check Item 2: Whether Node Resources (CPU and Memory) Are Sufficient**
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. **Check Item 3: Affinity and Anti-Affinity Configuration of the Workload**
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. **Check Item 4: Whether the Workload's Volume and the Node Are in the Same AZ**
0/1 nodes are available: 1 node(s) had taints that the pod didn't tolerate.	There are some taints on the node, and the pod cannot tolerate these taints. **Check Item 5: Tolerations of the Pod**
0/7 nodes are available: 7 Insufficient ephemeral-storage.	The ephemeral storage space on the node is insufficient. **Check Item 6: Ephemeral Volume Usage**
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. **Check Item 7: Whether the CCE Container Storage (Everest) Add-on Works Properly**
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. **Check Item 8: Whether the Thin Pool Space Is Sufficient**

Event	Cause and Solution
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. **Check Item 9: Whether the Node Has Too Many Pods Scheduled onto It**
UnexpectedAdmissionError Allocate failed due to not enough cpus available to satisfy request, which is unexpected.	The kubelet static CPU pinning is abnormal due to a known community issue. **Check Item 10: Whether the Static CPU Pinning of kubelet Is Abnormal**

Check Item 1: Whether a Node Is Available in the Cluster

You can log in to the CCE console and check whether the node status is **Available**. You can also use 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**, it means that there are no available nodes in the cluster.

Solution

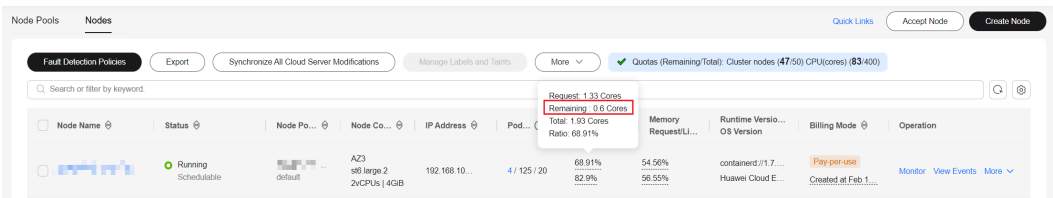
- Add a node. If no affinity rule is configured for the workload, the pod will be automatically scheduled to the new node to ensure proper service operation.
- Locate the unavailable nodes and rectify the faults. For details, see **What Should I Do If a Cluster Is Available But Some Nodes in It Are Unavailable?**
- Reset the unavailable nodes. For details, see **Resetting a Node**.

Check Item 2: Whether Node Resources (CPU and Memory) Are Sufficient

0/2 nodes are available: 2 Insufficient cpu. indicates that the CPUs are insufficient.

0/2 nodes are available: 2 Insufficient memory. indicates that the memory is insufficient.

If the resources requested by the pod exceed the allocatable resources on the node where the pod will run, the pod scheduling onto the node will definitely fail due to insufficient node resources.



If there are fewer allocatable resources on the node than the resources that a pod requests, the pod scheduling will fail.

Solution

Add more 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 the pod scheduling to fail.

For example, an anti-affinity policy is configured for workload 1 and workload 2. They run on node 1 and node 2, respectively.

If you try to configure an affinity policy for workload 3 and workload 2 and then deploy workload 3 on a node different from one hosting workload 2, such as node 1, it will cause a conflict and lead to the 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 and anti-affinity are not met.

Solution

- When configuring workload-workload affinity and workload-node affinity policies, ensure that these policies do not conflict with each other, or the workload deployment will fail.
- For a workload that has a node affinity policy configured, you need to make sure that **supportContainer** in the label of the affinity node is set to **true**. Otherwise, pods cannot be scheduled onto the 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 pod scheduling will fail.

Check Item 4: Whether the Workload's Volume and the Node Are in the Same AZ

0/2 nodes are available: 2 node(s) had volume node affinity conflict. indicates that an affinity conflict occurs between the volume mounted to the pod and the host node. As a result, the pod scheduling fails.

This is because EVS disks cannot be attached to nodes in different AZs from the EVS disks. For example, a workload pod with an EVS volume that is in AZ 1 cannot be scheduled to a node in AZ 2.

The EVS volumes created on CCE have 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:
                - ap-southeast-1a
```

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: Tolerations of the Pod

0/1 nodes are available: 1 node(s) had taints that the pod didn't tolerate. indicates that there are some taints on the node, and the pod cannot tolerate these taints.

In this case, you can check the taints on the node. If information similar to the following is displayed, there are some taints 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 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 is under memory pressure.
- node.kubernetes.io/disk-pressure: The node is under disk pressure. In this case, 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. Follow the instructions in **Changing Process ID Limits (kernel.pid_max)** to handle it.
- node.kubernetes.io/network-unavailable: The node network is unavailable.
- node.kubernetes.io/unschedulable: The node is unschedulable.
- node.cloudprovider.kubernetes.io/uninitialized: When kubelet is started with an external cloud platform driver specified, it adds a taint to the node, marking 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. indicates that there are not enough ephemeral storage space on the node.

In this case, you can check whether the space of the ephemeral volume is limited by the pod. If the ephemeral volume space required by the application exceeds the existing capacity on the node, the application cannot be scheduled to that node. To solve this problem, change the space of the ephemeral volume or expand the disk capacity on 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 volumes on the node, run the **kubectl describe node** command and check the memory request and limit of the allocated temporary volume on 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 the CCE Container Storage (Everest) Add-on Works Properly

0/1 nodes are available: 1 everest driver not found at node. indicates that everest-csi-driver of CCE Container Storage (Everest) is not started properly on the node.

In this case, you can check the daemon named **everest-csi-driver** in the **kube-system** namespace and check whether the pod is started properly. If it is not, delete the pod. The daemon will restart another pod.

Check Item 8: Whether the Thin Pool Space Is Sufficient

A data disk dedicated for kubelet and the container engine will be attached to a new node. For details, see **Data Disk Space Allocation**. If the data disk space is insufficient, the pod cannot be created on the node.

Solution 1: Clearing images

Perform the following operations to clear unused images:

- Nodes that use containerd
  a. Obtain local images on the node.
     ```
     crictl images -v
     ```
  b. Delete the unnecessary images by image ID.
     ```
     crictl rmi {Image ID}
     ```
- Nodes that use Docker
  a. Obtain local images on the node.
     ```
     docker images
     ```
  b. Delete the unnecessary images by image ID.
     ```
     docker rmi {}Image ID}
     ```

NOTE

Do not delete system images such as the **cce-pause** image. Otherwise, the pod creation may fail.

Solution 2: Expanding the disk capacity

To expand a disk capacity, perform the following operations:

Step 1 Expand the capacity of a data disk on the EVS console. For details, see **Expanding EVS Disk Capacity**.

Only the storage capacity of EVS disks can be expanded. You need to perform the following operations to expand the capacity of logical volumes and file systems.

Step 2 Log in to the CCE console and click the cluster name to access the cluster console. In the navigation pane, choose **Nodes**. In the right pane, click the **Nodes** tab, locate the row containing the target node, and choose **More** > **Sync Server Data** in the **Operation** column.

Step 3 Log in to the target node.

Step 4 Run **lsblk** to view the block device information of the node.

A data disk is divided depending on the container storage **Rootfs**:

Overlayfs: No independent thin pool is allocated. Image data is stored in **dockersys**.

1. Check the disk and partition space of the device.

```
# 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  150G  0 disk      # The data disk has been expanded to 150 GiB, but 50 GiB space is free.
├─vgpaas-dockersys  253:0    0   90G  0 lvm   /var/lib/containerd
└─vgpaas-kubernetes 253:1    0   10G  0 lvm   /mnt/paas/kubernetes/kubelet
```

2. Expand the disk capacity.

   Add the new disk capacity to the **dockersys** logical volume used by the container engine.

   a. Expand the PV capacity so that LVM can identify the new EVS capacity. */dev/sdb* specifies the physical volume where dockersys is located.

   ```
   pvresize /dev/sdb
   ```

   Information similar to the following is displayed:

   ```
   Physical volume "/dev/sdb" changed
   1 physical volume(s) resized or updated / 0 physical volume(s) not resized
   ```

   b. Expand 100% of the free capacity to the logical volume. *vgpaas/dockersys* specifies the logical volume used by the container engine.

   ```
   lvextend -l+100%FREE -n vgpaas/dockersys
   ```

   Information similar to the following is displayed:

   ```
   Size of logical volume vgpaas/dockersys changed from <90.00 GiB (23039 extents) to 140.00 GiB (35840 extents).
   Logical volume vgpaas/dockersys successfully resized.
   ```

   c. Adjust the size of the file system. */dev/vgpaas/dockersys* specifies the file system path of the container engine.

   ```
   resize2fs /dev/vgpaas/dockersys
   ```

   Information similar to the following is displayed:

   ```
   Filesystem at /dev/vgpaas/dockersys is mounted on /var/lib/containerd; on-line resizing required
   old_desc_blocks = 12, new_desc_blocks = 18
   The filesystem on /dev/vgpaas/dockersys is now 36700160 blocks long.
   ```

3. Check whether the capacity has been expanded.


```
# 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  150G  0 disk
├─vgpaas-dockersys 253:0    0   140G  0 lvm  /var/lib/containerd
└─vgpaas-kubernetes 253:1    0   10G  0 lvm  /mnt/paas/kubernetes/kubelet
```

Device Mapper: A thin pool is allocated to store image data.

1. Check the disk and partition space of the device.

```
# lsblk
NAME                        MAJ:MIN RM  SIZE RO TYPE MOUNTPOINT
vda                         8:0    0   50G  0 disk
└─vda1                      8:1    0   50G  0 part /
vdb                         8:16   0  200G  0 disk
├─vgpaas-dockersys          253:0    0   18G  0 lvm  /var/lib/docker
├─vgpaas-thinpool_tmeta     253:1    0    3G  0 lvm
│ └─vgpaas-thinpool         253:3    0   67G  0 lvm              # Space used by thin pool
│   ...
├─vgpaas-thinpool_tdata     253:2    0   67G  0 lvm
│ └─vgpaas-thinpool         253:3    0   67G  0 lvm
│   ...
└─vgpaas-kubernetes         253:4    0   10G  0 lvm  /mnt/paas/kubernetes/kubelet
```

2. Expand the disk capacity.

   Option 1: Add the new disk capacity to the thin pool.

   a. Expand the PV capacity so that LVM can identify the new EVS capacity. */dev/vdb* specifies the physical volume where thin pool is located.

   ```
   pvresize /dev/vdb
   ```

   Information similar to the following is displayed:

   ```
   Physical volume "/dev/vdb" changed
   1 physical volume(s) resized or updated / 0 physical volume(s) not resized
   ```

   b. Expand 100% of the free capacity to the logical volume. *vgpaas/thinpool* specifies the logical volume used by the container engine.

   ```
   lvextend -l+100%FREE -n vgpaas/thinpool
   ```

   Information similar to the following is displayed:

   ```
   Size of logical volume vgpaas/thinpool changed from <67.00 GiB (23039 extents) to <167.00 GiB (48639 extents).
   Logical volume vgpaas/thinpool successfully resized.
   ```

   c. Do not need to adjust the size of the file system, because the thin pool is not mounted to any devices.

   d. Run the **lsblk** command to check the disk and partition space of the device and check whether the capacity has been expanded. If the new disk capacity was added to the thin pool, the capacity has been expanded.

   ```
   # lsblk
   NAME                        MAJ:MIN RM  SIZE RO TYPE MOUNTPOINT
   vda                         8:0    0   50G  0 disk
   └─vda1                      8:1    0   50G  0 part /
   vdb                         8:16   0  200G  0 disk
   ├─vgpaas-dockersys          253:0    0   18G  0 lvm  /var/lib/docker
   ├─vgpaas-thinpool_tmeta     253:1    0    3G  0 lvm
   │ └─vgpaas-thinpool         253:3    0  167G  0 lvm            # Thin pool space after capacity expansion
   │   ...
   ├─vgpaas-thinpool_tdata     253:2    0   67G  0 lvm
   │ └─vgpaas-thinpool         253:3    0   67G  0 lvm
   │   ...
   └─vgpaas-kubernetes         253:4    0   10G  0 lvm  /mnt/paas/kubernetes/kubelet
   ```


Option 2: Add the new disk capacity to the **dockersys** disk.

a. Expand the PV capacity so that LVM can identify the new EVS capacity. */dev/vdb* specifies the physical volume where dockersys is located.

```
pvresize /dev/vdb
```

Information similar to the following is displayed:

```
Physical volume "/dev/vdb" changed
1 physical volume(s) resized or updated / 0 physical volume(s) not resized
```

b. Expand 100% of the free capacity to the logical volume. *vgpaas/dockersys* specifies the logical volume used by the container engine.

```
lvextend -l+100%FREE -n vgpaas/dockersys
```

Information similar to the following is displayed:

```
Size of logical volume vgpaas/dockersys changed from <18.00 GiB (4607 extents) to <118.00 GiB (30208 extents).
Logical volume vgpaas/dockersys successfully resized.
```

c. Adjust the size of the file system. */dev/vgpaas/dockersys* specifies the file system path of the container engine.

```
resize2fs /dev/vgpaas/dockersys
```

Information similar to the following is displayed:

```
Filesystem at /dev/vgpaas/dockersys is mounted on /var/lib/docker; on-line resizing required
old_desc_blocks = 3, new_desc_blocks = 15
The filesystem on /dev/vgpaas/dockersys is now 30932992 blocks long.
```

d. Run the **lsblk** command to check the disk and partition space of the device and check whether the capacity has been expanded. If the new disk capacity was added to the dockersys, the capacity has been expanded.

```
# lsblk
NAME                      MAJ:MIN RM  SIZE RO TYPE MOUNTPOINT
vda                       8:0    0   50G  0 disk
└─vda1                      8:1    0   50G  0 part /
vdb                       8:16   0  200G  0 disk
├─vgpaas-dockersys          253:0    0   118G  0 lvm  /var/lib/docker     # dockersys after capacity expansion
├─vgpaas-thinpool_tmeta       253:1    0     3G  0 lvm
│ └─vgpaas-thinpool           253:3    0    67G  0 lvm
│   ...
├─vgpaas-thinpool_tdata       253:2    0    67G  0 lvm
│ └─vgpaas-thinpool           253:3    0    67G  0 lvm
│   ...
└─vgpaas-kubernetes           253:4    0    10G  0 lvm  /mnt/paas/kubernetes/kubelet
```

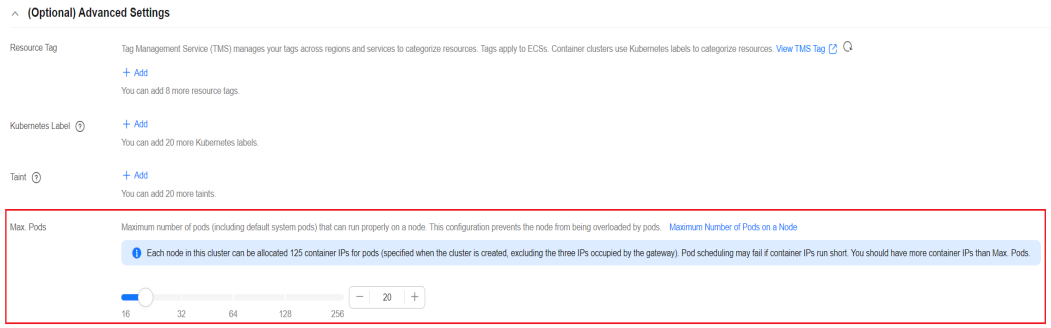
----End

Check Item 9: Whether the Node Has Too Many Pods Scheduled onto It

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 the **Advanced Settings** area 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.

Figure 6-2 Maximum number of pods

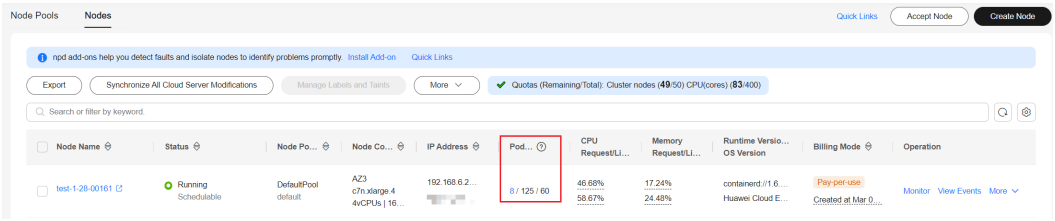


On the **Nodes** page, obtain the **Pods (Allocated/Total Available Addresses/Total)** value of the node, and check whether the number of pods scheduled onto the node has reached the upper limit. If so, add nodes or change the maximum number of pods.

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 custom node pool: Change the value of the node pool parameter **max-pods**. For details, see **Configuring a Node Pool**.

Figure 6-3 Checking the number of pods



Check Item 10: Whether the Static CPU Pinning of kubelet Is Abnormal

If a pod has an init container with a CPU request that is different from the main container settings, and it is assigned a Guaranteed QoS class while the kubelet uses static CPU pinning, the pod scheduling could fail, resulting in the error **UnexpectedAdmissionError**.

Community-related issue: **https://github.com/kubernetes/kubernetes/issues/112228**

Solution

Set the CPU request of the init container to a decimal value that matches the CPU limit and avoid using CPU pinning.

For example: the main container: {"limits":{"cpu":"7","memory":"60G"},"requests":{"cpu":"7","memory":"60G"}}; the init container: {"limits":{"cpu":"6.9","memory":"60G"},"requests":{"cpu":"6.9","memory":"60G"}}

6.1.3 What Should I Do If a Pod Fails to Pull the Image?

Fault Locating

When a workload's status shows "Pod not ready: Back-off pulling image "*xxxxx*", a Kubernetes event of **Failed 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

Determine the cause based on the events, as listed in **Table 6-2**.

Table 6-2 Events related to an image pull failure

Event	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. **Check Item 1: Whether imagePullSecret Is Specified When You Use kubectl to Create a Workload**
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. **Check Item 2: Whether the Image Address Is Correct When a Third-Party Image Is Used** **Check Item 3: Whether an Incorrect Secret Is Used When a Third-Party Image Is Used**
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. **Check Item 4: Whether the Node Disk Space Is Insufficient**
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. **Check Item 5: Whether the Remote Image Repository Uses an Unknown or Insecure Certificate**
Failed to pull image "*xxx*": rpc error: code = Unknown desc = context canceled	The image size is too large. **Check Item 6: Whether the Image Size Is Too Large**

Event	Cause and Solution
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)	**Check Item 7: Whether the Image Repository Can Be Accessed**
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. **Check Item 8: Whether the Number of Public Image Pulls Reaches the Upper Limit**

Check Item 1: Whether imagePullSecret Is Specified When You Use kubectl to Create a Workload

If the workload is abnormal and a Kubernetes event that indicates the pod fails to pull the image is displayed, you can 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 **name** 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. For details about how to create a secret, see **Using kubectl**.

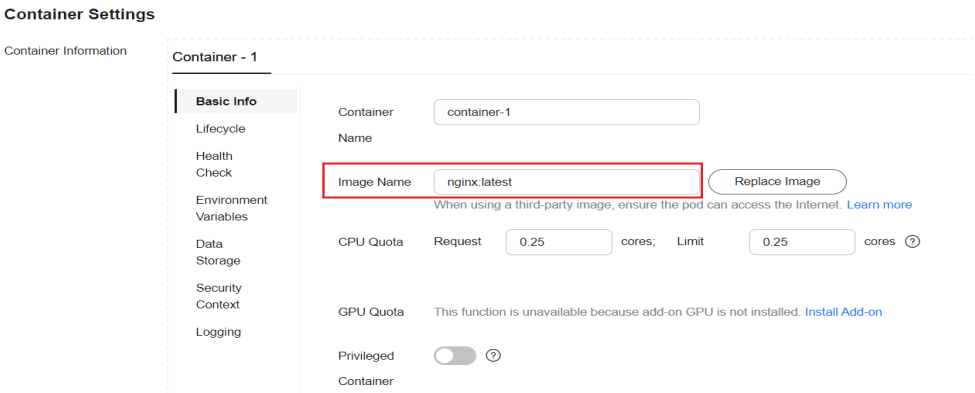
Check Item 2: Whether the Image Address Is Correct When a Third-Party Image Is Used

CCE allows you to create workloads using images pulled from third-party image repositories.

You need to enter the third-party image addresses based on specific requirements. The image address must be in the format of *domainname/organization/imagename:tag*. If no version is specified, **latest** is used by default.

- For a private repository, enter the image address in the format of *domainname/organization/imagename:tag*.
- For an open-source Docker repository, enter the image address in the format of *name:version*, for example, **nginx:latest**.

Figure 6-4 Using a third-party image



When you fail to pull an image due to incorrect image address provided, information similar to the following information is displayed in the Kubernetes events:

```
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

Change the image address by editing your YAML file or log in to the CCE console and 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

Typically, 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, so 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.

For details about how to create a secret, see **Using kubectl**.

Check Item 4: Whether the Node Disk Space Is Insufficient

If information similar to the following is displayed in the Kubernetes events, it means that 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
```

You can obtain the disk space for storing images on a node by running the following command:

```
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 unnecessary images by image ID.
     ```
     crictl rmi {Image ID}
     ```
- Nodes that use Docker
  a. Obtain local images on the node.
     ```
     docker images
     ```
  b. Delete the unnecessary images by image ID.
     ```
     docker rmi {}Image ID}
     ```

NOTE

Do not delete system images such as the **cce-pause** image. Otherwise, the pod creation may fail.

Solution 2: Expanding the disk capacity

To expand a disk capacity, perform the following operations:

Step 1 Expand the capacity of a data disk on the EVS console. For details, see **Expanding EVS Disk Capacity**.

Only the storage capacity of EVS disks can be expanded. You need to perform the following operations to expand the capacity of logical volumes and file systems.

Step 2 Log in to the CCE console and click the cluster name to access the cluster console. In the navigation pane, choose **Nodes**. In the right pane, click the **Nodes** tab, locate the row containing the target node, and choose **More** > **Sync Server Data** in the **Operation** column.

Step 3 Log in to the target node.

Step 4 Run **lsblk** to view the block device information of the node.

A data disk is divided depending on the container storage **Rootfs**:

Overlayfs: No independent thin pool is allocated. Image data is stored in **dockersys**.

1. Check the disk and partition space of the device.

```
# 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  150G  0 disk      # The data disk has been expanded to 150 GiB, but 50 GiB space is free.
├─vgpaas-dockersys 253:0    0   90G  0 lvm  /var/lib/containerd
└─vgpaas-kubernetes 253:1    0   10G  0 lvm  /mnt/paas/kubernetes/kubelet
```

2. Expand the disk capacity.

   Add the new disk capacity to the **dockersys** logical volume used by the container engine.

   a. Expand the PV capacity so that LVM can identify the new EVS capacity. */dev/sdb* specifies the physical volume where dockersys is located.

```
pvresize /dev/sdb
```

   Information similar to the following is displayed:

```
Physical volume "/dev/sdb" changed
1 physical volume(s) resized or updated / 0 physical volume(s) not resized
```

   b. Expand 100% of the free capacity to the logical volume. *vgpaas/dockersys* specifies the logical volume used by the container engine.

```
lvextend -l+100%FREE -n vgpaas/dockersys
```

   Information similar to the following is displayed:

```
Size of logical volume vgpaas/dockersys changed from <90.00 GiB (23039 extents) to 140.00 GiB (35840 extents).
Logical volume vgpaas/dockersys successfully resized.
```

   c. Adjust the size of the file system. */dev/vgpaas/dockersys* specifies the file system path of the container engine.

```
resize2fs /dev/vgpaas/dockersys
```

   Information similar to the following is displayed:

```
Filesystem at /dev/vgpaas/dockersys is mounted on /var/lib/containerd; on-line resizing required
old_desc_blocks = 12, new_desc_blocks = 18
The filesystem on /dev/vgpaas/dockersys is now 36700160 blocks long.
```

3. Check whether the capacity has been expanded.

```
# 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  150G  0 disk
├─vgpaas-dockersys 253:0    0  140G  0 lvm  /var/lib/containerd
└─vgpaas-kubernetes 253:1    0   10G  0 lvm  /mnt/paas/kubernetes/kubelet
```

Device Mapper: A thin pool is allocated to store image data.

1. Check the disk and partition space of the device.

```
# lsblk
NAME                      MAJ:MIN RM  SIZE RO TYPE MOUNTPOINT
vda                       8:0    0   50G  0 disk
└─vda1                      8:1    0   50G  0 part /
vdb                       8:16   0  200G  0 disk
```



```
├─vgpaas-dockersys              253:0    0   18G  0 lvm  /var/lib/docker
├─vgpaas-thinpool_tmeta         253:1    0    3G  0 lvm
│ └─vgpaas-thinpool             253:3    0   67G  0 lvm                # Space used by thin pool
│  ...
├─vgpaas-thinpool_tdata          253:2    0   67G  0 lvm
│ └─vgpaas-thinpool              253:3    0   67G  0 lvm
│  ...
└─vgpaas-kubernetes              253:4    0   10G  0 lvm  /mnt/paas/kubernetes/kubelet
```

2. Expand the disk capacity.

   Option 1: Add the new disk capacity to the thin pool.

   a. Expand the PV capacity so that LVM can identify the new EVS capacity. */dev/vdb* specifies the physical volume where thin pool is located.

   ```
   pvresize /dev/vdb
   ```

   Information similar to the following is displayed:

   ```
   Physical volume "/dev/vdb" changed
   1 physical volume(s) resized or updated / 0 physical volume(s) not resized
   ```

   b. Expand 100% of the free capacity to the logical volume. *vgpaas/thinpool* specifies the logical volume used by the container engine.

   ```
   lvextend -l+100%FREE -n vgpaas/thinpool
   ```

   Information similar to the following is displayed:

   ```
   Size of logical volume vgpaas/thinpool changed from <67.00 GiB (23039 extents) to <167.00 GiB (48639 extents).
   Logical volume vgpaas/thinpool successfully resized.
   ```

   c. Do not need to adjust the size of the file system, because the thin pool is not mounted to any devices.

   d. Run the **lsblk** command to check the disk and partition space of the device and check whether the capacity has been expanded. If the new disk capacity was added to the thin pool, the capacity has been expanded.

   ```
   # lsblk
   NAME                        MAJ:MIN RM  SIZE RO TYPE MOUNTPOINT
   vda                         8:0    0   50G  0 disk
   └─vda1                        8:1    0   50G  0 part /
   vdb                         8:16   0  200G  0 disk
   ├─vgpaas-dockersys            253:0    0   18G  0 lvm  /var/lib/docker
   ├─vgpaas-thinpool_tmeta       253:1    0    3G  0 lvm
   │ └─vgpaas-thinpool           253:3    0   167G  0 lvm            # Thin pool space after capacity expansion
   │   ...
   ├─vgpaas-thinpool_tdata       253:2    0   67G  0 lvm
   │ └─vgpaas-thinpool           253:3    0   67G  0 lvm
   │   ...
   └─vgpaas-kubernetes           253:4    0   10G  0 lvm  /mnt/paas/kubernetes/kubelet
   ```

   Option 2: Add the new disk capacity to the **dockersys** disk.

   a. Expand the PV capacity so that LVM can identify the new EVS capacity. */dev/vdb* specifies the physical volume where dockersys is located.

   ```
   pvresize /dev/vdb
   ```

   Information similar to the following is displayed:

   ```
   Physical volume "/dev/vdb" changed
   1 physical volume(s) resized or updated / 0 physical volume(s) not resized
   ```

   b. Expand 100% of the free capacity to the logical volume. *vgpaas/dockersys* specifies the logical volume used by the container engine.

   ```
   lvextend -l+100%FREE -n vgpaas/dockersys
   ```

   Information similar to the following is displayed:


```
Size of logical volume vgpaas/dockersys changed from <18.00 GiB (4607 extents) to <118.00
GiB (30208 extents).
Logical volume vgpaas/dockersys successfully resized.
```

c. Adjust the size of the file system. */dev/vgpaas/dockersys* specifies the file system path of the container engine.

```
resize2fs /dev/vgpaas/dockersys
```

Information similar to the following is displayed:

```
Filesystem at /dev/vgpaas/dockersys is mounted on /var/lib/docker; on-line resizing required
old_desc_blocks = 3, new_desc_blocks = 15
The filesystem on /dev/vgpaas/dockersys is now 30932992 blocks long.
```

d. Run the **lsblk** command to check the disk and partition space of the device and check whether the capacity has been expanded. If the new disk capacity was added to the dockersys, the capacity has been expanded.

```
# lsblk
NAME                      MAJ:MIN RM  SIZE RO TYPE MOUNTPOINT
vda                       8:0    0   50G  0 disk
└─vda1                      8:1    0   50G  0 part /
vdb                       8:16   0  200G  0 disk
├─vgpaas-dockersys          253:0    0   118G  0 lvm  /var/lib/docker     # dockersys after
capacity expansion
├─vgpaas-thinpool_tmeta       253:1    0    3G  0 lvm
│ └─vgpaas-thinpool           253:3    0   67G  0 lvm
│   ...
├─vgpaas-thinpool_tdata       253:2    0   67G  0 lvm
│ └─vgpaas-thinpool           253:3    0   67G  0 lvm
│   ...
└─vgpaas-kubernetes           253:4    0   10G  0 lvm  /mnt/paas/kubernetes/kubelet
```

----End

Check Item 5: Whether the Remote Image Repository Uses an Unknown or Insecure Certificate

If a pod tries to pull an image from a third-party repository with an unknown or insecure certificate, the image pulling will fail on the node. The pod events contain "Failed to pull the image" with the cause "x509: certificate signed by unknown authority".

NOTE

The security of EulerOS 2.9 images has been improved by removing insecure or expired certificates from the system. While some third-party images on certain nodes may not report any errors, it is common for this type of error to occur in EulerOS 2.9. To fix the issue, you can perform the following operations.

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 "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-
AQbl5RActnudDZV%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 pulled.

Run the following command on an EulerOS or CentOS node with *{server_url}:{server_port}* replaced with the IP address and port number obtained in the previous step, 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 an Ubuntu node:

```
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 events contain "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
```

However, the image can be manually pulled by running the **docker pull** command on the node.

Possible Cause

In Kubernetes clusters, there is a default timeout period for pulling images. If the image pulling progress is not updated within a certain period of time, the pulling will be 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)** Solution 1:
  a. Log in to the node and manually pull the image.
     - Nodes that use containerd:
       ```
       crictl pull <image-address>
       ```
     - Nodes that use Docker:
       ```
       docker pull <image-address>
       ```
  b. When creating a workload, ensure that **imagePullPolicy** is set to **IfNotPresent** (the default configuration). In this case, the workload can use the image that has been pulled to the local host.
- (For clusters of v1.25 or later) Solution 2: Modify the configuration parameters of the node pool. The configuration parameters for nodes in the **DefaultPool** node pool cannot be modified.
  a. Log in to the CCE console.
  b. Click the cluster name to access the cluster console. In the navigation pane, choose **Nodes**. In the right pane, click the **Node Pools** tab.

c. Locate the row containing the target node pool and click **Manage**.
d. In the window that slides out from the right, modify the **image-pull-progress-timeout** parameter under **Docker/containerd**. This **image-pull-progress-timeout** parameter specifies the timeout interval for pulling an image.
e. Click **OK**.

Check Item 7: Whether the Image Repository Can Be Accessed

Symptom

An error message similar to the following is displayed during workload creation:

```
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)
```

Possible Cause

The image repository cannot be accessed due to the network problems. SWR allows you to pull images only from the official Docker repository. To pull images from other repositories, you need to access the Internet.

Solution

- Bind a public IP address to the node to which the image needs to be pulled.
- Push images to SWR and then pull them from SWR to your nodes.

Check Item 8: Whether the Number of Public Image Pulls Reaches the Upper Limit

Symptom

An error message similar to the following is displayed during workload creation:

```
ERROR: toomanyrequests: Too Many Requests.
```

Or

```
you have reached your pull rate limit, you may increase the limit by authenticating an upgrading: https://www.docker.com/increase-rate-limits.
```

Possible Cause

Docker Hub sets limits for the maximum number of container image pull requests. For details, see **Docker Hub pull usage and limits**.

Solution

Push the frequently used images to SWR and then pull them from SWR to your nodes.

6.1.4 What Should I Do If a Pod Startup Fails?

Fault Locating

On the details page of a workload, if an event is displayed indicating that the pod fails to be started, perform the following operations 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.

If the node uses Docker, run the following command:

```
docker ps -a | grep $podName
```

If the node uses containerd, run the following command:

```
crictl ps -a | grep $podName
```

Step 3 View the container logs.

If the node uses Docker, run the following command:

```
docker logs $containerID
```

If the node uses containerd, run the following command:

```
crictl logs $containerID
```

Rectify the fault of the workload based on logs.

Step 4 Check the error logs of the OS. For example, check whether the logs contain any OOM errors.

```
cat /var/log/messages | grep $containerID | grep oom
```

Check whether a system OOM is triggered based on the logs.

----End

Troubleshooting

Determine the cause based on the logs or events, as listed in **Table 6-3**.

Table 6-3 Pod startup failure

Log or Event	Possible Cause	Fault Locating and Solution
Pod logs: exit code 0	There is no process in the pod.	Check whether the pod can run properly. For details, see **No Process in the Pod (Exit Code: 0)**.
• Kubernetes event: Liveness probe failed: Get http… • Pod logs: exit code 137	The health check fails.	Check whether the liveness probe for the pod is properly configured. For details, see **Health Check Failed (Exit Code: 137)**.

Log or Event	Possible Cause	Fault Locating and Solution
• Kubernetes event: `Thin Pool has 15991 free data blocks which is 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` • Pod logs: **no left space**	The disk space is insufficient.	Expand the disk space or clear unneeded files. For details, see **Insufficient Disk Space of the Pod**.
Pod logs: **oom**	The pod memory is insufficient.	Check whether the pod has proper resource settings. For details, see **Insufficient Container Resources**.
Pod logs: **Address already in use**	A conflict occurs between container ports in the pod.	Check whether there is a container port conflict in the pod. For details, see **Container Port Conflict in the Pod**.
Kubernetes event: `Error: failed to start container "filebeat": Error response from daemon: OCI runtime create failed: container_linux.go:330: starting container process caused "process_linux.go:381: container init caused \"setenv: invalid argument\"": unknown`	A secret is mounted to the workload, and the value of the secret is not encrypted using Base64.	For details about the solution, see **Improper Value of the Secret Mounted to the Workload**.
Kubernetes event: `the failed container exited with ExitCode: 255`	The x86 container image may run on an Arm node.	For details about the solution, see **Unmatched Container Image Tag with the Node Architecture**.
Kubernetes event: `the failed container exited with ExitCode: 141`	The containerd version is incompatible with the tail version.	For details about the solution, see **Exit of tail -f xx in the Container Startup Command (Exit Code: 141)**.
Kubernetes event: `Created container init-pinpoint`	The Java probe version is incompatible.	For details about the solution, see **Incompatible Java Probe Version with the Container**.
Other pod logs	Locate the fault based on services.	For details about the fault locating, see **Service Setting Checks**.

No Process in the Pod (Exit Code: 0)

Step 1 Log in to the node where the abnormal workload is located.

Step 2 View the pod status.

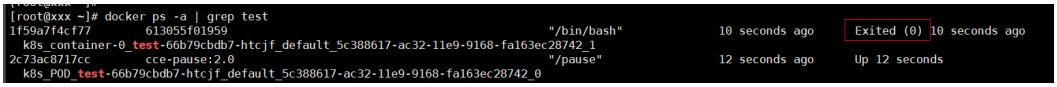
If the node uses Docker, run the following command:

```
docker ps -a | grep $podName
```

If the node uses containerd, run the following command:

```
crictl ps -a | grep $podName
```

Below shows an example.



If there is no process in the pod, the status code **Exited (0)** is displayed.

----End

Health Check Failed (Exit Code: 137)

The health checks configured for a workload are performed on services periodically. If an exception occurs, there will be an event that indicates an unhealthy pod, and the pod restarts will fail.

If a liveness probe is configured for the workload and the number of health check failures exceeds the threshold, 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.

Insufficient Disk Space of the Pod

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 is 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
```

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 unnecessary images by image ID.
     ```
     crictl rmi {Image ID}
     ```
- Nodes that use Docker
  a. Obtain local images on the node.
     ```
     docker images
     ```
  b. Delete the unnecessary images by image ID.
     ```
     docker rmi {}Image ID}
     ```

NOTE

Do not delete system images such as the **cce-pause** image. Otherwise, the pod creation may fail.

Solution 2: Expanding the disk capacity

To expand a disk capacity, perform the following operations:

Step 1 Expand the capacity of a data disk on the EVS console. For details, see **Expanding EVS Disk Capacity**.

Only the storage capacity of EVS disks can be expanded. You need to perform the following operations to expand the capacity of logical volumes and file systems.

Step 2 Log in to the CCE console and click the cluster name to access the cluster console. In the navigation pane, choose **Nodes**. In the right pane, click the **Nodes** tab, locate the row containing the target node, and choose **More** > **Sync Server Data** in the **Operation** column.

Step 3 Log in to the target node.

Step 4 Run **lsblk** to view the block device information of the node.

A data disk is divided depending on the container storage **Rootfs**:

Overlayfs: No independent thin pool is allocated. Image data is stored in **dockersys**.

1. Check the disk and partition space of the device.
   ```
   # 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  150G  0 disk      # The data disk has been expanded to 150 GiB, but 50 GiB space is free.
   ├─vgpaas-dockersys 253:0    0   90G  0 lvm   /var/lib/containerd
   └─vgpaas-kubernetes 253:1    0   10G  0 lvm   /mnt/paas/kubernetes/kubelet
   ```
2. Expand the disk capacity.

   Add the new disk capacity to the **dockersys** logical volume used by the container engine.

   a. Expand the PV capacity so that LVM can identify the new EVS capacity. */dev/sdb* specifies the physical volume where dockersys is located.
      ```
      pvresize /dev/sdb
      ```


Information similar to the following is displayed:

```
Physical volume "/dev/sdb" changed
1 physical volume(s) resized or updated / 0 physical volume(s) not resized
```

b. Expand 100% of the free capacity to the logical volume. *vgpaas/dockersys* specifies the logical volume used by the container engine.

```
lvextend -l+100%FREE -n vgpaas/dockersys
```

Information similar to the following is displayed:

```
Size of logical volume vgpaas/dockersys changed from <90.00 GiB (23039 extents) to 140.00 GiB (35840 extents).
Logical volume vgpaas/dockersys successfully resized.
```

c. Adjust the size of the file system. */dev/vgpaas/dockersys* specifies the file system path of the container engine.

```
resize2fs /dev/vgpaas/dockersys
```

Information similar to the following is displayed:

```
Filesystem at /dev/vgpaas/dockersys is mounted on /var/lib/containerd; on-line resizing required
old_desc_blocks = 12, new_desc_blocks = 18
The filesystem on /dev/vgpaas/dockersys is now 36700160 blocks long.
```

3. Check whether the capacity has been expanded.

```
# 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  150G  0 disk
├─vgpaas-dockersys 253:0    0   140G  0 lvm  /var/lib/containerd
└─vgpaas-kubernetes 253:1    0   10G  0 lvm  /mnt/paas/kubernetes/kubelet
```

Device Mapper: A thin pool is allocated to store image data.

1. Check the disk and partition space of the device.

```
# lsblk
NAME                        MAJ:MIN RM  SIZE RO TYPE MOUNTPOINT
vda                         8:0    0   50G  0 disk
└─vda1                        8:1    0   50G  0 part /
vdb                         8:16   0  200G  0 disk
├─vgpaas-dockersys            253:0    0   18G  0 lvm  /var/lib/docker
├─vgpaas-thinpool_tmeta       253:1    0    3G  0 lvm
│ └─vgpaas-thinpool           253:3    0   67G  0 lvm              # Space used by thin pool
│   ...
├─vgpaas-thinpool_tdata       253:2    0   67G  0 lvm
│ └─vgpaas-thinpool           253:3    0   67G  0 lvm
│   ...
└─vgpaas-kubernetes           253:4    0   10G  0 lvm  /mnt/paas/kubernetes/kubelet
```

2. Expand the disk capacity.

Option 1: Add the new disk capacity to the thin pool.

a. Expand the PV capacity so that LVM can identify the new EVS capacity. */dev/vdb* specifies the physical volume where thin pool is located.

```
pvresize /dev/vdb
```

Information similar to the following is displayed:

```
Physical volume "/dev/vdb" changed
1 physical volume(s) resized or updated / 0 physical volume(s) not resized
```

b. Expand 100% of the free capacity to the logical volume. *vgpaas/thinpool* specifies the logical volume used by the container engine.

```
lvextend -l+100%FREE -n vgpaas/thinpool
```

Information similar to the following is displayed:

```
Size of logical volume vgpaas/thinpool changed from <67.00 GiB (23039 extents) to <167.00 GiB (48639 extents).
Logical volume vgpaas/thinpool successfully resized.
```


c. Do not need to adjust the size of the file system, because the thin pool is not mounted to any devices.

d. Run the **lsblk** command to check the disk and partition space of the device and check whether the capacity has been expanded. If the new disk capacity was added to the thin pool, the capacity has been expanded.

```
# lsblk
NAME                      MAJ:MIN RM  SIZE RO TYPE MOUNTPOINT
vda                       8:0    0   50G  0 disk
└─vda1                      8:1    0   50G  0 part /
vdb                       8:16   0  200G  0 disk
├─vgpaas-dockersys          253:0    0   18G  0 lvm  /var/lib/docker
├─vgpaas-thinpool_tmeta       253:1    0    3G  0 lvm
│ └─vgpaas-thinpool           253:3    0  167G  0 lvm            # Thin pool space after
capacity expansion
│   ...
├─vgpaas-thinpool_tdata       253:2    0   67G  0 lvm
│ └─vgpaas-thinpool           253:3    0   67G  0 lvm
│   ...
└─vgpaas-kubernetes           253:4    0   10G  0 lvm  /mnt/paas/kubernetes/kubelet
```

Option 2: Add the new disk capacity to the **dockersys** disk.

a. Expand the PV capacity so that LVM can identify the new EVS capacity. */dev/vdb* specifies the physical volume where dockersys is located.

```
pvresize /dev/vdb
```

Information similar to the following is displayed:

```
Physical volume "/dev/vdb" changed
1 physical volume(s) resized or updated / 0 physical volume(s) not resized
```

b. Expand 100% of the free capacity to the logical volume. *vgpaas/dockersys* specifies the logical volume used by the container engine.

```
lvextend -l+100%FREE -n vgpaas/dockersys
```

Information similar to the following is displayed:

```
Size of logical volume vgpaas/dockersys changed from <18.00 GiB (4607 extents) to <118.00
GiB (30208 extents).
Logical volume vgpaas/dockersys successfully resized.
```

c. Adjust the size of the file system. */dev/vgpaas/dockersys* specifies the file system path of the container engine.

```
resize2fs /dev/vgpaas/dockersys
```

Information similar to the following is displayed:

```
Filesystem at /dev/vgpaas/dockersys is mounted on /var/lib/docker; on-line resizing required
old_desc_blocks = 3, new_desc_blocks = 15
The filesystem on /dev/vgpaas/dockersys is now 30932992 blocks long.
```

d. Run the **lsblk** command to check the disk and partition space of the device and check whether the capacity has been expanded. If the new disk capacity was added to the dockersys, the capacity has been expanded.

```
# lsblk
NAME                      MAJ:MIN RM  SIZE RO TYPE MOUNTPOINT
vda                       8:0    0   50G  0 disk
└─vda1                      8:1    0   50G  0 part /
vdb                       8:16   0  200G  0 disk
├─vgpaas-dockersys          253:0    0   118G  0 lvm  /var/lib/docker     # dockersys after
capacity expansion
├─vgpaas-thinpool_tmeta       253:1    0    3G  0 lvm
│ └─vgpaas-thinpool           253:3    0   67G  0 lvm
│   ...
├─vgpaas-thinpool_tdata       253:2    0   67G  0 lvm
│ └─vgpaas-thinpool           253:3    0   67G  0 lvm
```



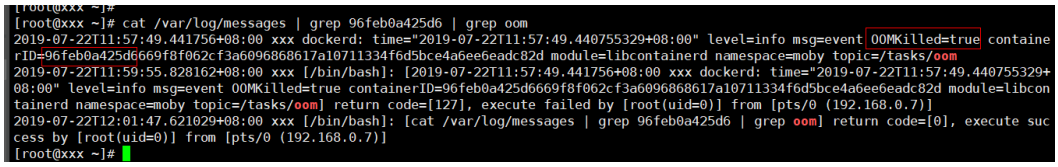
```
...
└─vgpaas-kubernetes              253:4    0   10G  0 lvm  /mnt/paas/kubernetes/kubelet
```

----End

Insufficient Container Resources

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
```



When a workload is created, if the requested resources exceed the configured upper limit, the system OOM is triggered and the container exits unexpectedly.

Container Port Conflict in the Pod

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.

If the node uses Docker, run the following command:

```
docker ps -a | grep $podName
```

If the node uses containerd, run the following command:

```
crictl ps -a | grep $podName
```

Step 3 View the container logs.

If the node uses Docker, run the following command:

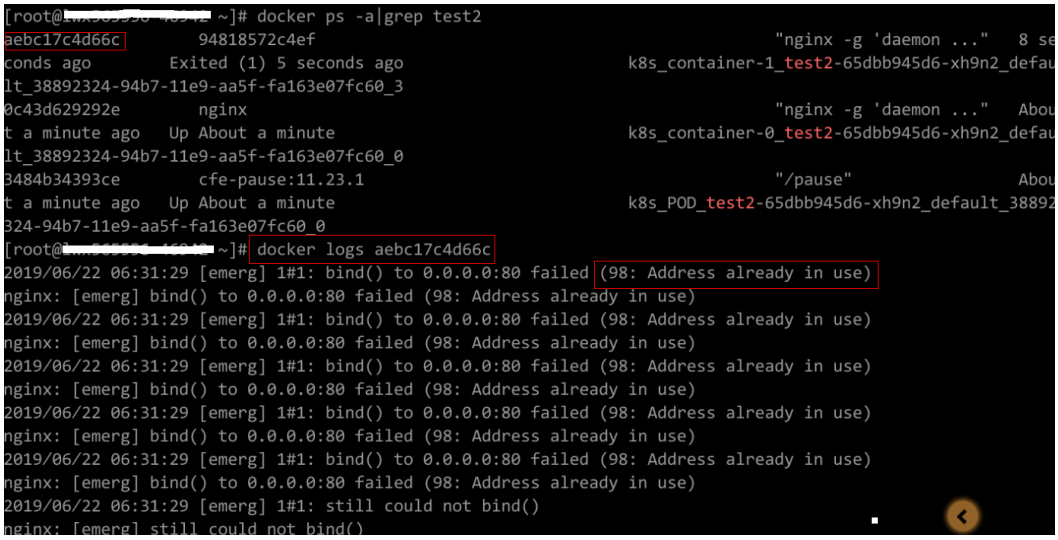
```
docker logs $containerID
```

If the node uses containerd, run the following command:

```
crictl 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 6-5 Pod restart failure due to a container port conflict



----End

Solution

Configure proper container ports that do not conflict with each other. Then, create the workload again.

If a pod uses the host network (with **hostNetwork: true** configured), there may be a container port conflict. This is because containers in the pod share the network interface and port range with the host node. Multiple pods using the same port cannot run on the same node.

Improper Value of the Secret Mounted to the Workload

Information similar to the following is displayed in the event:

```
Error: failed to start container "filebeat": Error response from daemon: OCI runtime create failed: container_linux.go:330: starting container process caused "process_linux.go:381: container init caused \"setenv: invalid argument\"": unknown
```

The root cause is that a secret is mounted to the workload, but the value of the secret is not encrypted using Base64.

Solution

Create a secret on the console. The value of the secret is automatically encrypted using Base64.

If you use YAML to create a secret, you need to manually encrypt its value using Base64.

```
echo -n "Content to be encoded" | base64
```

Unmatched Container Image Tag with the Node Architecture

The proper image tag is not used during the workload creation on an Arm node. To resolve this issue, use the proper image tag.

For details about how to create an image in dual-architecture (x86 and Arm), see **Using Dual-Architecture Images (x86 and Arm) in CCE**.

Exit of tail -f *xx* in the Container Startup Command (Exit Code: 141)

The Kubernetes event is as follows:

```
the failed container exited with ExitCode: 141
```

Possible Cause

The legacy version of containerd is incompatible with the tail version (≥ 8.28) in the container image. As a result, executing the **tail -f** command leads to an unexpected exit, returning exit code 141.

Temporary Workaround

Change **tail -f** *xx* in the startup parameters to **sleep 2 && tail -f** *xx* and create a workload again.

Solution

- Upgrade the cluster to v1.25.16-r20, v1.27.16-r20, v1.28.15-r10, v1.29.10-r10, v1.30.6-r10, v1.31.4-r0, or later.
- Reset the node and use the Docker container engine instead.

Incompatible Java Probe Version with the Container

The Kubernetes event is as follows:

```
Created container init-pinpoint
```

Solution

1. When creating a workload, select the specific latest Java probe version (for example, **1.0.36**, not the **latest** option) on the **APM Settings** tab in the **Advanced Settings** area.
2. If you selected **latest** for the Java probe during workload creation, you can upgrade the workload and change it to the specific latest version (for example, **1.0.36**).

Service Setting Checks

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.

If the node uses Docker, run the following command:

```
docker ps -a | grep $podName
```

If the node uses containerd, run the following command:

```
crictl ps -a | grep $podName
```

Step 3 View the container logs.

If the node uses Docker, run the following command:

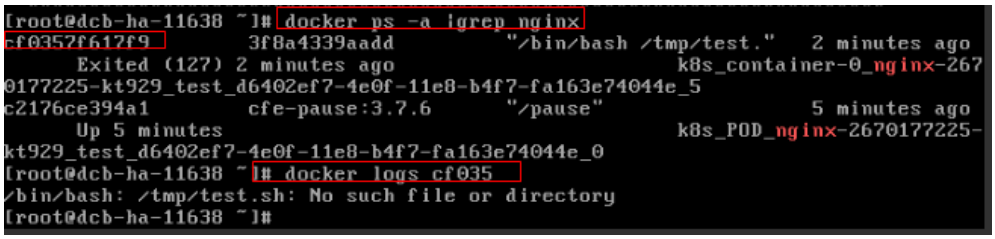
```
docker logs $containerID
```

If the node uses containerd, run the following command:


```
crictl logs $containerID
```

Note: In the preceding command, *containerID* indicates the ID of the container that has exited.

Figure 6-6 Incorrect startup command of the container



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.

6.1.5 What Should I Do If a Pod Fails to Be Evicted?

Principle of Eviction

When a node is abnormal, Kubernetes will evict some pods on the node to ensure 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 node controller. Node controller periodically checks the status of all nodes. If a node is in the **NotReady** state for a period of time, all pods on the node will be evicted.

  kube-controller-manager supports the following startup parameters:

  - **pod-eviction-timeout**: indicates an interval when a node is down, after which pods on that node are evicted. The default interval is 5 minutes.
  - **node-eviction-rate**: indicates the number of nodes to be evicted per second. The default value is **0.1**, indicating that pods are evicted from one node every 10 seconds.
  - **secondary-node-eviction-rate**: specifies a rate at which nodes are evicted in the second grade. If a large number of nodes are down in the cluster, the eviction rate will be reduced to **secondary-node-eviction-rate**. The default value is **0.01**.
  - **unhealthy-zone-threshold**: specifies a threshold for an AZ to be considered unhealthy. The default value is **0.55**, meaning that if the percentage of faulty nodes in an AZ exceeds 55%, the AZ will be considered unhealthy.
  - **large-cluster-size-threshold**: specifies a threshold for a cluster to be considered large. The parameter defaults to **50**. If there are more nodes

than this threshold, the cluster is considered as a large one. If there are more than 55% faulty nodes in a cluster, the eviction rate is reduced to 0.01. If the cluster is a small one, the eviction rate is reduced to 0, which means, pods running on the nodes in the cluster will not be evicted.

- **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 faulty node, while kubelet evicts some pods on a faulty node. kubelet periodically checks the memory and disk resources of nodes. If the resources are insufficient, it will evict some pods based on the priority. For details about the pod eviction priority, see **Pod selection for kubelet eviction**.

  There are soft eviction thresholds and hard eviction thresholds.

  – **Soft eviction thresholds**: A grace period is configured for node resources. kubelet will reclaim node resources associated with these thresholds if that grace period elapses. If the node resource usage reaches these thresholds but falls below them before the grace period elapses, kubelet will not evict pods on the node.

    You can configure soft eviction thresholds using the following parameters:

    ▪ **eviction-soft**: indicates a soft eviction threshold. If a node's **eviction signal** reaches a certain threshold, for example, **memory.available<1.5Gi**, kubelet will not immediately evict some pods on the node but wait for a grace period configured by **eviction-soft-grace-period**. If the threshold is reached after the grace period elapses, kubelet will evict some pods on the node.

    ▪ **eviction-soft-grace-period**: indicates an eviction grace period. If a pod reaches the soft eviction threshold, it will be terminated after the configured grace period elapses. This parameter indicates the time difference for a terminating pod to respond to the threshold being met.

    ▪ **eviction-max-pod-grace-period**: indicates the maximum allowed grace period to use when terminating pods in response to a soft eviction threshold being met.

  – **Hard eviction thresholds**: Pods are immediately evicted once these thresholds are reached.

    You can configure hard eviction thresholds using the following parameters:

    eviction-hard: indicates a hard eviction threshold. When the **eviction signal** of a node reaches a certain threshold, for example, **memory.available<1Gi**, which means, when the available memory of the node is less than 1 GiB, a pod eviction will be triggered immediately.

    kubelet supports the following default hard eviction thresholds:

    ▪ **memory.available<100Mi**

    ▪ **nodefs.available<10%**

- **imagefs.available<15%**
- **nodefs.inodesFree<5%** (for Linux nodes)

kubelet also supports other parameters:

- **eviction-pressure-transition-period**: indicates a period for which the 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 **DiskPressure** or **MemoryPressure**. Then some pods running on the node will be evicted. This parameter can prevent mistaken eviction decisions when a node is oscillating above and below a soft eviction threshold in some cases.
- **eviction-minimum-reclaim**: indicates the minimum number of resources that must be reclaimed in each eviction. This parameter can prevent kubelet from repeatedly evicting pods because only a small number of resources are reclaimed during pod evictions in some cases.

Fault Locating

If the pods are not evicted when the node is faulty, perform the following operations to locate the fault:

After the following command is executed, 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/log/cce/kubernetes/kubelet.log | grep -i Evicted -C3
```

Troubleshooting

The issues here are described in order of how likely they are to occur.

Check these causes one by one until you find the cause of the fault.

- **Check Item 1: Whether the Node Is Under Resource Pressure**
- **Check Item 2: Whether Tolerations Have Been Configured for the Workload**
- **Check Item 3: Whether the Conditions for Stopping Pod Eviction Are Met**
- **Check Item 4: Whether the Allocated Resources of the Pod Are the Same as Those of the Node**
- **Check Item 5: Whether the Workload Pod Fails Continuously and Is Redeployed**

Check Item 1: Whether the Node Is Under Resource Pressure

If a node suffers resource pressure, kubelet will change the **node status** and add taints to the node. Perform the following operations to check whether the corresponding taint exists on the node:

```
$ kubectl describe node 192.168.0.37
Name:           192.168.0.37
...
```



```
Taints:          key1=value1:NoSchedule
...
```

Table 6-4 Statuses of nodes with resource pressure and solutions

Node Status	Taint	Eviction Signal	Description	Solution
MemoryPressure	node.kubernetes.io/memory-pressure	memory.available	The available memory on the node reaches the eviction thresholds.	You can scale out node specifications. For details, see **How Do I Change the Node Specifications in a CCE Cluster?**
DiskPressure	node.kubernetes.io/disk-pressure	nodefs.available, nodefs.inodesFree, imagefs.available or imagefs.inodesFree	The available disk space and inode on the root file system or image file system of the node reach the eviction thresholds.	You can expand the storage space of the node. For details, see **Expanding the Storage Space**.
PIDPressure	node.kubernetes.io/pid-pressure	pid.available	The available process identifier on the node is below the eviction thresholds.	You can modify the upper limit of PIDs on the node. For details, see **Changing Process ID Limits (kernel.pid_max)**.

Check Item 2: Whether Tolerations Have Been Configured for the Workload

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 3: Whether the Conditions for Stopping Pod Eviction Are Met

In a cluster that runs fewer than 50 worker nodes, if 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 4: Whether the Allocated Resources of the Pod Are the Same as Those of the Node

An evicted pod will be frequently scheduled to the original node.

Possible cause

Pods on a node are evicted based on the node resource usage. The evicted pods are 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 5: Whether the Workload Pod Fails Continuously and Is Redeployed

A workload pod 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 -n <namespace> | grep Evicted | awk '{print $1}' | xargs kubectl delete pod -n <namespace>
```

In the preceding command, *<namespace>* indicates the namespace name. Configure it based on your requirements.

References

Kubelet does not delete evicted pods

Submitting a Service Ticket

If the problem persists, **submit a service ticket**.

6.1.6 What Should I Do If a Storage Volume Cannot Be Mounted or the Mounting Times Out?

Fault Locating

- Abnormal EVS Storage Volume Mounting
- Abnormal SFS Turbo Storage Volume Mounting
- Abnormal SFS Storage Volume Mounting
- Storage Volume Mounting Timed Out

Abnormal EVS Storage Volume Mounting

Symptom	Possible Cause	Solution
Mounting an EVS volume to a StatefulSet times out.	The node and the volume are in different AZs, causing a timeout during the mounting process and preventing the volume from being mounted to the workload.	Create a volume in the same AZ as the node and mount the volume to the node.
A pod fails to be created, and an event similar to the following is displayed, indicating that the **volume fails to be mounted** to the pod is reported. Multi-Attach error for volume "pvc-62a7a7d9-9dc8-42a2-8366-0f5ef9db5b60" **Volume is already used by pod(s)** testttt-7b774658cb-lc98h	The number of pods of the Deployment that uses an EVS volume is greater than 1. If the Deployment uses an EVS volume, there can only be one Deployment pod. If you specify more than two pods for the Deployment, it will still be created. However, if these pods are scheduled on different nodes, some of them will fail to start because the EVS volume they rely on cannot be mounted to those nodes.	Set the number of pods of the Deployment that uses an EVS volume to 1 or use other types of volumes.

Symptom	Possible Cause	Solution
A pod fails to be created, and information similar to the following is displayed: 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**	The disk file system has been corrupted.	Back up the disk in EVS and restore the file system: fsck -y *{drive letter}*

Abnormal SFS Turbo Storage Volume Mounting

Symptom	Possible Cause	Solution

• In a common container scenario, the pod is in the **Processing** state, and the pod events include the following information.
`MountVolume.SetUp failed for volume {pv name}...`
• In a secure container scenario, the pod is in the **Abnormal** state, and the pod events include the following information:
`mount {SFS-Turbo-shared-address} to xxx failed`
	1. The shared address in the PV is incorrect.
	2. The network connection between the node where the pod runs and the SFS Turbo file system to be mounted is disconnected.
		1. Check whether the shared address in the PV is correct.
		Obtain the YAML file of the PV and check the value of the **everest.io/share-export-location** field in **spec.csi.volumeAttributes**. (The correct shared address is the share path of the specified SFS Turbo file system.)
		`kubectl get pv {pv name} -ojsonpath='{.spec.csi.volumeAttributes.everest\.io\/share-export-location}{"\n"}'`
		If a sub-path is specified, it must be a valid existing subdirectory in the correct format, for example, **192.168.135.24:/a/b/c**.
		2. Verify the network connectivity between the node where the pod runs and the SFS Turbo file system to be mounted.
		Check whether the SFS Turbo file system can be mounted to a workload:
		`mount -t nfs -o vers=3,nolock,noresvport {SFS-Turbo-shared-address} /tmp`

Abnormal SFS Storage Volume Mounting

Symptom	Possible Cause	Solution
The pod is in the **Processing** state and the pod events contain an alarm that indicates the PV mounting failed. The events are as follows: MountVolume.SetUp failed for volume {pv name}...**please check whether the volumeAttributes of related PersistentVolume of the volume is correct and whether it can be mounted.**	The workload uses a general purpose file system (SFS 3.0 Capacity-Oriented), but the VPC endpoint needed by the file system has not been created. As a result, the file system is inaccessible.	Create a VPC endpoint in the VPC where the cluster is located. For details, see **Configure a VPC Endpoint**.

Storage Volume Mounting Timed Out

If the volume to be mounted stores too much 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.

6.1.7 What Should I Do If a Workload Remains in the Creating State?

Symptom

The workload remains in the creating state.

Troubleshooting

Possible causes are described here in order of how likely they are to occur.

If the fault persists after you have ruled out a cause, check other 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 Cause

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 -xzvf /opt/cloud/cce/package/node-package/pause-*.tgz
```

Step 3 Import the image.

- Nodes that use Docker:
  ```
  docker load -i ./pause/package/image/cce-pause-*.tar
  ```
- Nodes that use containerd:
  ```
  ctr -n k8s.io images import --all-platforms ./pause/package/image/cce-pause-*.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

6.1.8 What Should I Do If a Pod Remains in the Terminating State?

Symptom

When obtaining workloads in a namespace, you may come across pods that are in the **Terminating** state.

For example, if you use the command below to obtain pods in the **aos** namespace, you may notice that some pods are 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-789bf5b497-6rwrg    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
```

Possible Cause

The following lists some possible causes:

- **The node that runs a terminating pod is abnormal.** When a node is unavailable, CCE migrates 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 will be automatically deleted.
- **A container is unresponsive.** If a container within a pod fails to respond to the SIGTERM signal while being terminated, the pod can become stuck in the **Terminating** state.
- **There are unfinished requests or resource occupation within a pod.** If a process within a pod continues to run for an extended period, it may prevent the pod from being terminated and cause it to enter the **Terminating** state.
- **A pod has finalizers specified.** Finalizers are used to clean up resources before they are deleted. If a pod has finalizers specified and the cleanup process is paused or unresponsive, the pod will remain in the **Terminating** state.
- **A pod has terminationGracePeriodSeconds configured.** Once a graceful exit time is set for a pod, it will enter the **Terminating** state upon termination and be automatically deleted after exiting gracefully.

Solution

NOTE

Before forcibly deleting a pod, it is important to consider the potential risks to your services. This is especially true for StatefulSet pods, because there is a higher chance of data inconsistency and abnormal container exits. Take the time to evaluate these risks before proceeding with the operation. For details, see **Force Delete StatefulSet Pods**.

Run the following command to forcibly delete the pods created in any ways:


```
kubectl delete pod <pod> -n <namespace> --grace-period=0 --force
```

Run the following command to delete the terminating pod described at the very beginning of this section:

```
kubectl delete pod aos-apiserver-5f8f5b5585-s9l92 -n aos --grace-period=0 --force
```

6.1.9 What Should I Do If a Workload Is Stopped Caused by Pod Deletion?

Symptom

A workload is in **Stopped** state.

Possible 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**.

6.1.10 What Should I Do If an Error Occurs When I Deploy a Service on a 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 obtained.

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 CCE AI Suite (NVIDIA GPU) add-on has an outdated driver version. After a new driver is downloaded and installed, the fault is rectified.
2. You did not specify the requirement for GPUs in workloads.

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, run the following command:
    ```
    cd /usr/local/nvidia/bin && ./nvidia-smi
    ```
- Container:
  - If the cluster version is v1.27 or earlier, run the following command:
    ```
    cd /usr/local/nvidia/bin && ./nvidia-smi
    ```
  - If the cluster version is v1.28 or later, run the following command:
    ```
    cd /usr/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?**
- **Installing the gpu add-on**

6.1.11 What Should I Do If a Workload Exception Occurs Due to a Storage Volume Mount Failure?

Symptom

A workload is always in the creating state, and an alarm indicating that a storage volume fails to be mounted is generated. The event is as follows:

AttachVolume.Attach failed for volume "pvc-***" : rpc error: code = Internal desc = [***][disk.csi.everest.io] attaching volume *** to node *** failed: failed to send request of attaching disk(id=***) to node(id=***): error statuscode 400 for posting request, response is {"badRequest": {"message": "**Maximum number of scsi disk exceeded**", "code": 400}}, request is {"volumeAttachment": {"volumeId":"***","device":"","id":"","serverId":"","bus":"","pciAddress":"","VolumeWwn":"","VolumeMultiAttach":false,"VolumeMetadata":null}}, url is: .....

Possible Cause

This alarm indicates that the number of EVS disks attached to a node has reached the limit. In this case, if a workload pod with an EVS disk attached is scheduled to this node, the disk attachment will fail. As a result, the workload cannot run properly.

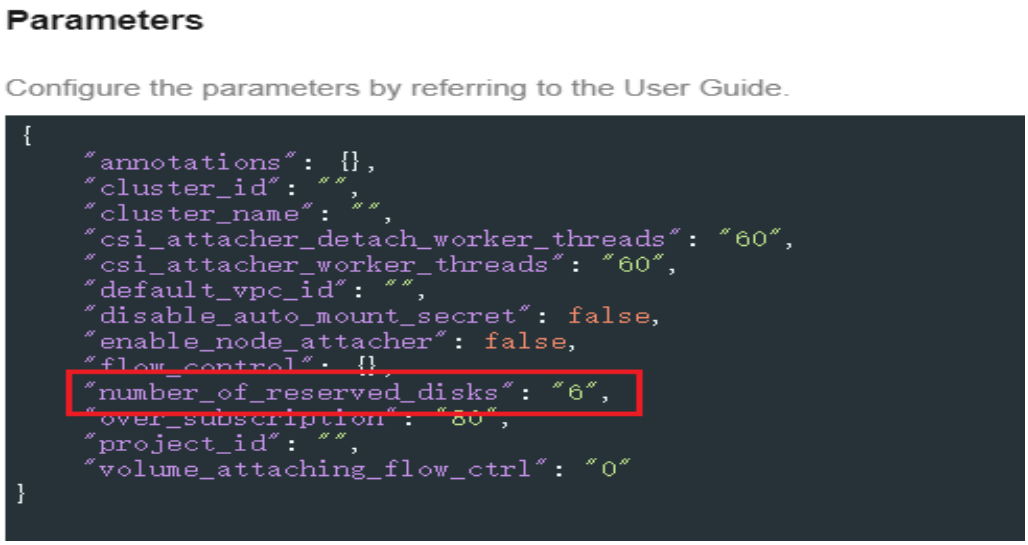
If no more than 20 EVS disks can be attached to a node, the node, already has one system disk and one data disk attached, can only accept up to 18 additional EVS disks. If two raw disks are attached to the node through the ECS console for creating a local storage pool, only 16 additional data disks can be attached to the node. If the node has 18 workload pods scheduled to it, each with one EVS disk attached, two of those pods will encounter the preceding error due to disk quotas being exceeded.

Solution

CCE Container Storage (Everest) 2.3.11 or later supports **number_of_reserved_disks**, which is used to configure the number of disks reserved on a node. By configuring this parameter, you can reserve disk slots for your nodes. Note that the modification of this parameter applies to all nodes in a cluster.

After **number_of_reserved_disks** is configured, the number of the additional EVS disks that can be attached to a node is calculated as follows:

Number of remaining disks attached to a node= Maximum number of EVS disks that can be attached to the node - Value of number_of_reserved_disks



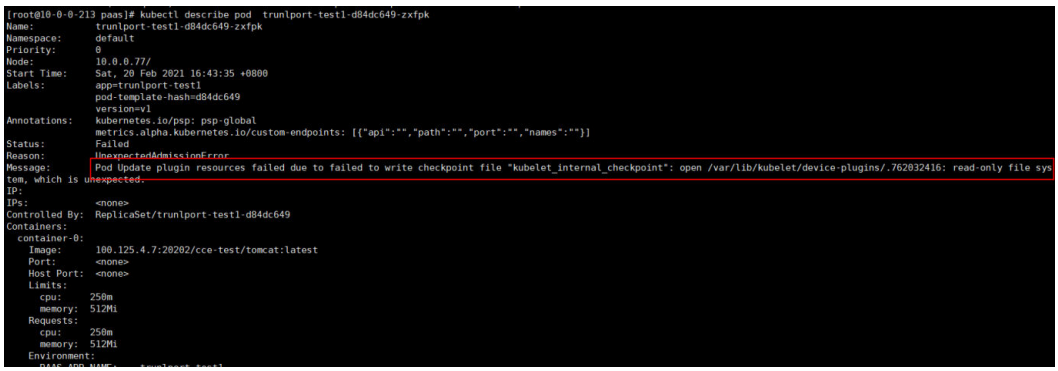
If the maximum number of EVS disks that can be attached to a node is **20** and **number_of_reserved_disks** is set to **6**, the number of the additional EVS disks that can be attached to the node is **14** (20 – 6 = 14) when a workload with EVS disks attached needs to be scheduled. The reserved six disks include one system disk and one data disk that have been attached to the node. You can attach four EVS disks to this node as additional data disks or raw disks for a local storage pool. In this scenario, if 18 workload pods, each with one EVS disk attached, need to be scheduled in the cluster, the node can accept 14 workload pods at most. The remaining four workload pods will be scheduled to other nodes in the cluster. In this way, the problem that the storage volume mount failure will not occur.

6.1.12 Why Does Pod Fail to Write Data?

Pod Events

The file system of the node where the pod is located is damaged. As a result, the newly created pod cannot write data to **/var/lib/kubelet/device-plugins/.xxxxx**. Events similar to the following may occur in the pod:

```
Message: Pod Update Plugin resources failed due to failed to write checkpoint file "kubelet_internal_checkpoint": open /var/lib/kubelet/device-plugins/.xxxxxx: read-only file system, which is unexpected.
```



Such abnormal pods are recorded in error events but do not occupy system resources.

Procedure

There are many causes for file system exceptions, for example, the physical master node is powered on or off unexpectedly. If the file systems are not restored and a large number of pods becomes abnormal (which do not affect services), perform the following operations:

Step 1 Run the **kubectl drain <node-name>** command to mark the node as unschedulable, and evict existing pods to other nodes.

```
kubectl drain <node-name>
```

Step 2 Locate the cause of the file system exception and rectify the fault.

Step 3 Run the following command to make the node schedulable:

```
kubectl uncordon <node-name>
```

----End

Clearing Abnormal Pods

- The garbage collection mechanism of kubelet is the same as that of the community. After the owner (for example, Deployment) of the pod is cleared, the abnormal pod is also cleared.
- You can run the kubelet command to delete the pod recorded as abnormal.

6.1.13 What Should I Do If a Workload Appears to Be Normal But Is Not Functioning Properly?

Symptom

A pod is in the **Running** state, but it is not functioning properly or the access result does not meet the expectations.

Possible Cause

There could be errors in your YAML files, including those related to pods, Deployments, and StatefulSets. Such errors include:

- The image version is not updated. This could be due to using an incorrect image version or having both the old and new images as the latest version. In some cases, even if an image of the old version exists on the node, the new version is not pulled if the **imagePullPolicy** of the workload is set to **IfNotPresent**. As a result, the container continues to run with the old version of the image.
- The environment variables are incorrectly configured. For example, if the word **command** is misspelled as **commnd** in the YAML file, the workload can still be created using that YAML file. However, when the container is running, it executes the default **EntryPoint** command from the image instead of the intended command.

Solution

1. Check the pod configuration and verify if the container configuration within the pod aligns with the desired expectations. For details, see **Checking Pod Configurations**.
2. To check whether a key in an environment variable is misspelled, perform the following operations: (The following takes the example of misspelling the word **command** as **commnd** to show how to identify spelling issues.)
   a. Before running **kubectl apply -f**, add **--validate** to it and then run the **kubectl apply --validate -f** *XXX*.**yaml** command. If you spelled **command** as **commnd**, you will see an error message similar to the following:
   ```
   I0805 10:43:25.129850   46757 schema.go:126] unknown field: commnd
   I0805 10:43:25.129973   46757 schema.go:129] this may be a false alarm, see https://
   github.com/kubernetes/kubernetes/issues/6842/pods/mypod
   ```
   b. Compare the **pod.yaml** file in the command output with the file used for creating the pod:
   ```
   kubectl get pods/$mypod  yaml > mypod.yaml
   ```
   $mypod specifies the name of the abnormal pod. You can run the **kubectl get pods** command to view the pod name.
   - If the **mypod.yaml** file has more lines than the pod file used for creation, then the created pod meets the expected configuration.
   - If some code lines are missing in the **mypod.yaml** file compared with those in the file used for creating the pod, it indicates a spelling error in the YAML file.
3. View the pod logs and locate the fault based on the logs. For details, see **Viewing Container Logs**.
4. Access the container through the terminal and check whether the local files in the container meet the expectation. For details, see **Logging In to a Container**.

6.1.14 Why Is Pod Creation or Deletion Suspended on a Node Where File Storage Is Mounted?

Symptom

On the node to which SFS or SFS Turbo volumes are mounted, pod deletion tasks stay in the **Stopping** state, and pod creation tasks remain **Creating**.

Possible Cause

- The backend file storage is deleted. As a result, the mount point cannot be accessed.
- The network between the node and the file storage is abnormal. As a result, the mount point cannot be accessed.

Solution

Step 1 Log in to the node to which the file storage is mounted and run the following command to find the mount path of the file storage:

findmnt

Example mount path: **/mnt/paas/kubernetes/kubelet/pods/7b88feaf-71d6-4e6f-8965-f5f0766d9f35/volumes/kubernetes.io~csi/sfs-turbo-ls/mount**

Step 2 Run the following command to access the file storage folder:

cd /mnt/paas/kubernetes/kubelet/pods/7b88feaf-71d6-4e6f-8965-f5f0766d9f35/volumes/kubernetes.io~csi/sfs-turbo-ls/mount

If the access fails, the file storage is deleted or the network between the file storage and the node is abnormal.

Step 3 Run the **umount -l** command to unmount the file storage.

umount -l /mnt/paas/kubernetes/kubelet/pods/7b88feaf-71d6-4e6f-8965-f5f0766d9f35/volumes/kubernetes.io~csi/sfs-turbo-ls/mount

Step 4 Restart kubelet.

systemctl restart kubelet

----End

Root Causes

This problem usually occurs when the hard mounts are used for file storage. In this mode, all processes that access the mount point are hung until the access is successful. You can use soft mounts to avoid this issue. For details, see **Setting Mount Options**.

6.1.15 How Can I Locate Faults Using an Exit Code?

When a container fails to be started or terminated, the exit code is recorded by Kubernetes events to report the cause. This section describes how to locate faults using an exit code.

Viewing an Exit Code

You can use kubectl to connect to the cluster and run the following command to check the pod:

```
kubectl describe pod {pod name}
```

In the command output, the **Exit Code** field indicates the status code of the last program exit. If the value is not **0**, the program exits abnormally. You can further analyze the cause through this code.

```
Containers:
  container-1:
    Container ID: ...
    Image: ...
    Image ID: ...
    Ports: ...
    Host Ports: ...
    Args: ...
    State: Running
      Started: Sat, 28 Jan 2023 09:06:53 +0000
    Last State: Terminated
      Reason: Error
```



```
    Exit Code: 255
    Started: Sat, 28 Jan 2023 09:01:33 +0000
    Finished: Sat, 28 Jan 2023 09:05:11 +0000
  Ready: True
  Restart Count: 1
```

Description

The exit code ranges from 0 to 255.

- If the exit code is 0, the container exits normally.
- Generally, if the abnormal exit is caused by the program, the exit code ranges from 1 to 128. In special scenarios, the exit code ranges from 129 to 255.
- When a program exits due to external interrupts, the exit code ranges from 129 to 255. When the operating system sends **an interrupt signal** to the program, the exit code is the interrupt signal value plus 128. For example, if the interrupt signal value of **SIGKILL** is 9, the exit status code is 137 (9 + 128).
- If the exit code is not in the range of 0 to 255, for example, exit(-1), the exit code is automatically converted to a value that is within this range.

  If the exit code is a positive number, the conversion formula is as follows:

  ```
  code % 256
  ```

  If the exit code is a negative number, the conversion formula is as follows:

  ```
  256 - (|code| % 256)
  ```

For details, see **Exit Codes With Special Meanings**.

Common Exit Codes

Table 6-5 Common exit codes

Exit Code	Name	Description
0	Normal exit	The container exits normally. This status code does not always indicate that an exception occurs. When there is no process in the container, it may also be displayed.
1	Common program error	There are many causes for this exception, most of which are caused by the program. You need to further locate the cause through container logs. For example, this error occurs when an x86 image is running on an Arm node.
125	The container is not running.	The possible causes are as follows: • An undefined flag is used in the command, for example, **docker run --abcd**. • The user-defined command in the image has insufficient permissions on the local host. • The container engine is incompatible with the host OS or hardware.

Exit Code	Name	Description
126	Command calling error	The command called in the image cannot be executed. For example, the file permission is insufficient or the file cannot be executed.
127	The file or directory cannot be found.	The file or directory specified in the image cannot be found.
128	Invalid exit parameter	The container exits but no valid exit code is provided. There are multiple possible causes. You need to further locate the cause. For example, an application running on the containerd node attempts to call the docker command.
137	Immediate termination (SIGKILL)	The program is terminated by the **SIGKILL** signal. The common causes are as follows: • The memory usage of the container in the pod reaches the resource limit. For example, OOM causes cgroup to forcibly stop the container. • If OOM occurs, the kernel of the node stops some processes to release the memory. As a result, the container may be terminated. • If the container health check fails, kubelet stops the container. • Other external processes, such as malicious scripts, forcibly stop the container.
139	Segmentation error (SIGSEGV)	The container receives the **SIGSEGV** signal from the OS because the container attempts to access an unauthorized memory location.
143	Graceful termination (SIGTERM)	The container is correctly closed as instructed by the host. Generally, this exit code 143 does not require troubleshooting.
255	The exit code is out of range.	The container exit code is out of range. For example, exit(-1) may be used for abnormal exit, and -1 is automatically converted to 255. Further troubleshooting is required.

Linux Standard Interrupt Signals

You can run the **kill -l** command to view the signals and corresponding values in the Linux OS.

Table 6-6 Common Linux standard interrupt signals

Signal	Value	Action	Commit
SIGHUP	1	Term	Sent when the user terminal connection (normal or abnormal) ends.
SIGINT	2	Term	Program termination signal, which is sent by the terminal by pressing **Ctrl+C**.
SIGQUIT	3	Core	Similar to **SIGINT**, the exit command is sent by the terminal. Generally, the exit command is controlled by pressing **Ctrl+**.
SIGILL	4	Core	Invalid instruction, usually because an error occurs in the executable file.
SIGABRT	6	Core	Signal generated when the abort function is invoked. The process ends abnormally.
SIGFPE	8	Core	A floating-point arithmetic error occurs. Other arithmetic errors such as divisor 0 also occur.
SIGKILL	9	Term	Any process is terminated.
SIGSEGV	11	Core	Attempt to access an unauthorized memory location.
SIGPIPE	13	Term	The pipe is disconnected.
SIGALRM	14	Term	Indicates clock timing.
SIGTERM	15	Term	Process end signal, which is usually the normal exit of the program.
SIGUSR1	10	Term	This is a user-defined signal in applications.
SIGUSR2	12	Term	This is a user-defined signal in applications.
SIGCHLD	17	Ign	This signal is generated when a subprocess ends or is interrupted.
SIGCONT	18	Cont	Resume a stopped process.
SIGSTOP	19	Stop	Suspend the execution of a process.
SIGTSTP	20	Stop	Stop a process.
SIGTTIN	21	Stop	The background process reads the input value from the terminal.
SIGTTOU	22	Stop	The background process reads the output value from the terminal.

6.1.16 What Can I Do If a Large Number of Pods in a Cluster Are in the UnexpectedAdmissionError State?

Symptom

When you obtain pods in a cluster, a large number of pods are in the **UnexpectedAdmissionError** state. For example, when you run the **kubectl get pod -A** command, information similar to the following is displayed:

```
NAME                                READY   STATUS                     RESTARTS   AGE
aos-apiserver-5f8f5b5585-s9l92      0/1     UnexpectedAdmissionError   0          3d1h
aos-cmdbserver-789bf5b497-6rwrg     0/1     UnexpectedAdmissionError   0          3d1h
aos-controller-545d78bs8d-vm6j9     0/1     UnexpectedAdmissionError   3          3d1h
...
```

Possible Cause

After a scheduler assigns a pod to a node but the node lacks sufficient resources, such as CPUs, memory, or heterogeneous resources, to meet the pod requirements, kubelet will reject the pod and mark it as failed.

Solution

Pods in the **UnexpectedAdmissionError** state are not cleared immediately. When the total number of pods in the **Completed** or **UnexpectedAdmissionError** state exceeds 1,000, CCE performs a centralized cleanup. At this stage, resources previously occupied by these pods have already been released, and only their status records remain for troubleshooting and issue diagnosis. If these pods are not needed, you can manually delete them.

6.1.17 What Can I Do If There Is an Abnormal Pod and a Message Stating That the Device Files Can't Be Found?

Symptom

A pod fails to be created, and an error message similar to the following is displayed:

```
Error: failed to generate container "af736..." spec: failed to apply OCI options: lstat /dev/davinci4: no such file or directory
```

Run the following command to check the number of PCIe buses:

```
lspci | grep -i accelerator | wc -l
```

If the number of PCIe buses returned is less than the expected number, there will be a PCIe link down.

Possible Cause

When the PCIe link of an Ascend Snt9 device is down, the device driver cannot report this message. Consequently, tasks continue to be scheduled to the disconnected NPU.

You can run the following command to check whether there is a PCIe link down:

```
npu-smi info -m
```

Information similar to the following is displayed:

```
NPU ID    Chip ID    Chip Logic ID    Chip Name
0         0          0                Ascend xxx
0         1          -                Mcu
1         0          1                Ascend xxx
1         1          -                Mcu
2         0          2                Ascend xxx
2         1          -                Mcu
3         0          3                Ascend xxx
3         1          -                Mcu
5         0          4                Ascend xxx
5         1          -                Mcu
6         0          5                Ascend xxx
6         1          -                Mcu
7         0          6                Ascend xxx
7         1          -                Mcu
```

According to the NPU IDs, the fourth NPU of the Ascend Snt9 device was lost, indicating that there was a PCIe link down. During this process, the device driver rearranged the chip logic IDs of the NPUs, resulting in the loss of chip logic ID 7. When the CCE AI Suite (Ascend NPU) add-on reported information, only the chip logic IDs were updated, while the mapping between the chip logic IDs and NPU IDs remained unchanged. Consequently, Kubernetes marked the seventh NPU (originally corresponding to chip logic ID 7) as faulty and unavailable. The fourth NPU (originally corresponding to chip logic ID 4) was still incorrectly recognized as an available resource by the scheduler.

Solution

Upgrade the Snt9 device driver to version 24.1.rc2 or later. The driver of the new version ensures that the chip logic IDs are no longer rearranged after a PCIe link down, so the accurate information can be reported.

After upgrading the driver, run the following command and verify the conclusion:

```
npu-smi info -m
```

The command output shows that the driver of the new version does not rearrange the chip logic IDs after a PCIe link down.

```
NPU ID    Chip ID    Chip Logic ID    Chip Name
-1        0          -                Mcu
0         0          0                Ascend xxx
0         1          -                Mcu
1         0          1                Ascend xxx
1         1          -                Mcu
2         0          2                Ascend xxx
2         1          -                Mcu
3         0          3                Ascend xxx
3         1          -                Mcu
5         0          5                Ascend xxx
5         1          -                Mcu
6         0          6                Ascend xxx
6         1          -                Mcu
7         0          7                Ascend xxx
7         1          -                Mcu
```


6.2 Container Configuration

6.2.1 When Is Pre-stop Processing Used?

Question

When is pre-stop processing used?

Answer

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.

6.2.2 When Would a Container Need to Be Rebuilt?

Question

When would a container need to be rebuilt?

Possible Cause

Container rebuilding refers to the process of destroying an existing container and creating a new one. There are various reasons that can trigger container rebuilding. The table below illustrates some common scenarios.

Table 6-7 Typical scenarios

Scenario	Description
A container crashes or terminates abnormally.	If a running container encounters software errors, resource exhaustion, or other exceptions, the system will automatically rebuild the container to quickly restore services and ensure uninterrupted service.
A container is manually deleted.	If a running container is deleted manually, the container orchestration tool will automatically reschedule and rebuild the container based on the defined deployment policy, ensuring the desired number of pods is maintained.
There is a priority-based preemption.	When a high-priority pod needs resources, Kubernetes may remove a low-priority pod, which will then be rescheduled and started again.
The configuration is updated.	If the configuration (such as the image version, environment variables, and data storage) of a Deployment or StatefulSet is updated, rolling update will be triggered. As a result, the existing container is gradually destroyed and a new one will be created.

Scenario	Description
Node resources on are insufficient.	When the resources on a node, such as memory and CPUs, are not enough, the cluster may remove certain pods and schedule them on other nodes that have enough resources. This action will then trigger the rebuilding of the containers.
The node is restarted or faulty.	If a node is restarted for any reason, the containers on that node may be destroyed and rebuilt on other available nodes. Similarly, if a node in the cluster is found to be faulty, the cluster will recognize that the node is unavailable and proceed to rebuild the containers on other available nodes.

Using **storage** (such as EVS disks and file systems) can effectively prevent data loss caused by rebuilding of a container. This ensures that important data is persistently stored and can be used after the container is rebuilt.

6.2.3 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.

6.2.4 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.

6.2.5 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
   ```

6.2.6 What Is the Retry Mechanism When CCE Fails to Start a Pod?

CCE is a cloud container engine service built on native Kubernetes. It fully supports native Kubernetes versions, 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.

6.3 Monitoring Log

6.3.1 How Long Are the Events of a Workload Stored?

In a cluster of v1.7.3-r12, 1.9.2-r3, or a later version, the event information of a workload is stored for one hour, after which the data is automatically cleared.

In clusters earlier than 1.7.3-r12, events are stored for 24 hours.

6.3.2 Why Is the Reported Container Memory Usage Inconsistent with the Auto Scaling Action?

Symptom

The memory usage of a container being monitored does not match the auto scaling requirements. For example, the GUI shows that memory usage of a container is around 40%, but the HPA scale-in threshold is set at 70%. The memory usage displayed on the GUI is below the HPA threshold, but no scale-in action has taken place.

Possible Cause

The way the container memory usage is calculated on the GUI differs from the method used for HPA auto scaling.

- The memory usage of a container displayed on the GUI:
  container_memory_rss/Memory limit of the container

  container_memory_rss specifies the resident set size (RSS). It includes some memory parts that may not be actively or effectively used.
- The memory usage of a container calculated for HPA auto scaling:
  container_memory_working_set_bytes/Memory request of the container

  container_memory_working_set_bytes specifies the working set size (WSS) and is calculated by doing as follows:

  Run **cat /sys/fs/cgroup/memory/memory.stat** in the pod to obtain the values of **total_cache** (cache memory), **total_rss** (memory used by the

current application process), and **total_inactive_file** (memory used by inactive files).

WSS = The value of **total_cache** + The value of **total_rss** - The value of **total_inactive_file**

If an application's memory usage displayed on the GUI is below the HPA scale-in threshold, but no scale-in action has been taken, or if the memory usage is not higher than the HPA scale-out threshold, but a scale-out happened anyways, the HPA scale-out behavior may not work as expected. This can occur when:

- The application cache usage is high, which can cause the WSS to be significantly greater than the RSS, resulting in the container memory usage displayed on the GUI being lower than the memory usage calculated by the HPA.
- The difference between the resource limit and request is large. The request may be significantly lower than the limit, causing the container memory usage shown on the GUI to be lower than the memory usage calculated by the HPA.

6.4 Scheduling Policies

6.4.1 How Do I Evenly Distribute Multiple Pods to Each Node?

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**.

You can configure pod anti-affinity policies to evenly distribute pods onto different nodes.

An example is as follows:

```
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:                # Workload anti-affinity
        preferredDuringSchedulingIgnoredDuringExecution:    # Ensure that the following conditions are met:
          - weight: 100 # Priority that can be configured when the best-effort policy is used. The value
ranges from 1 to 100. A larger value indicates a higher priority.
            podAffinityTerm:
              labelSelector:                    # Select the label of the pod, which is anti-affinity with 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
```

6.4.2 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 a pod is killed because of excessive resource usage (while the node resources are still sufficient), the system tends to restart the pod on the same node.
- 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).

6.4.3 Why Are Pods Not Evenly Distributed on Nodes?

Pod Scheduling Principles in Kubernetes

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**.

Possible Causes of Why Pods Are Not Evenly Distributed on Nodes

- Resource configurations may vary between nodes, including differences in CPU and memory size. This can result in the pods' defined requests not being met, even if a node's actual load is low. As a result, the pod cannot be scheduled to the node.
- Custom scheduling policies can be used to schedule pods based on affinity and anti-affinity rules, resulting in uneven distribution of pods across nodes.
- Nodes can have taints that prevent unscheduled pods from being assigned to them if tolerations are not set.
- Certain workloads may have unique distribution constraints. For instance, if an EVS disk is attached to a workload, the workload pods can only be scheduled to nodes within the same AZ as the disk.
- Certain pods may require specific resources, such as GPUs. In such cases, the scheduler can only assign those pods to GPU nodes.
- The health and status of a node can impact scheduling decisions. If a node is unhealthy, it may not be able to accept new pods.

Possible Causes of Why Pod Loads Are Unevenly Distributed on Nodes

When allocating pods, kube-scheduler does not take into account the actual load of applications. This can result in some nodes being heavily loaded while others are lightly loaded, especially if the application load is uneven.

Volcano Scheduler offers CPU and memory load-aware scheduling for pods and preferentially schedules pods to the node with the lightest load to balance node loads. This prevents an application or node failure due to heavy loads on a single node. For details, see **Load-aware Scheduling**.

6.4.4 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 access 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 exists on the node, the message "eerror: unable to drain node "192.168.0.160" due to error: cannot delete DaemonSet-managed Pods..." 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**.

6.4.5 How Do I Check Whether a Pod Uses CPU Binding?

The following takes a node with 4 vCPUs and 8 GiB of memory as an example. A workload whose CPU request is **1** and limit is **2** is deployed in the cluster in advance.

Step 1 Log in to a node in the node pool and view the **/var/lib/kubelet/cpu_manager_state** output.

```
cat /var/lib/kubelet/cpu_manager_state
```

Information similar to the following will be displayed:

```
{"policyName":"static","defaultCpuSet":"0,2-3","entries":{"c1fcd22d-8a83-4aef-a27a-4c037e482b16":
{"container-1":"1"}},"checksum":1500530529}
```

If the value of **policyName** is **static**, the policy has been configured.

Step 2 Check the cgroup setting of **cpuset.preferred_cpus** of the container. The output is the ID of the CPU that is preferentially used.

```
cat /sys/fs/cgroup/cpuset/kubepods/pod {pod uid} / {Container id} /cpuset.cpus
```

- *{pod uid}* indicates the pod UID. It can be obtained by running the following command on the host that has been connected to the cluster using kubectl:

  ```
  kubectl get po {pod name} -n {namespace} -ojsonpath='{.metadata.uid}{"\n"}'
  ```

  In the preceding command, *{pod name}* indicates the pod name and *{namespace}* indicates the namespace to which the pod belongs.

- *{Container id}* must be a complete container ID. To obtain the ID, run the following command on the node where the container is running:

  Docker node pool: In the command, *{pod name}* indicates the pod name.

  ```
  docker ps --no-trunc | grep {pod name} | grep -v cce-pause | awk '{print $1}'
  ```

  containerd node pool: In the command, *{pod name}* indicates the pod name, *{pod id}* indicates the pod ID, and *{container name}* indicates the container name.

  ```
  # Obtain the pod ID.
  crictl pods | grep {pod name} | awk '{print $1}'
  # Obtain the complete container ID.
  crictl ps --no-trunc | grep {pod id} | grep {container name} | awk '{print $1}'
  ```


A complete example is as follows:

```
cat /sys/fs/cgroup/cpuset/kubepods/podc1fcd22d-8a83-4aef-a27a-4c037e482b16/5cb15f55f429e4496172bef05994477caa96e0ca468563208695c1ad5cc141e0/cpuset.cpus
```

The command output shows that CPU 1 is bound.

```
1
```

----End

6.4.6 What Should I Do If Pods Cannot Be Rescheduled After the Node Is Stopped?

Symptom

After a node is stopped, pods on the node are still running. The latest pod event obtained by running **kubectl describe pod** *<pod-name>* is displayed as follows:

```
Warning NodeNotReady 17s node-controller Node is not ready
```

Possible Cause

After a node is stopped, the system automatically adds taints to the node.

- node.kubernetes.io/unreachable:NoExecute
- node.cloudprovider.kubernetes.io/shutdown:NoSchedule
- node.kubernetes.io/unreachable:NoSchedule
- node.kubernetes.io/not-ready:NoExecute

If a pod has tolerations for these taints, it will not be rescheduled. Therefore, check the tolerations of the pod.

Solution

Check the tolerations by viewing the YAML file of the pod or workload. The tolerations of a workload consist of the following fields:

```
tolerations:
- key: "key1"
  operator: "Equal"
  value: "value1"
  effect: "NoSchedule"
```

Or:

```
tolerations:
- key: "key1"
  operator: "Exists"
  effect: "NoSchedule"
```

If the preceding tolerations are incorrectly configured, the scheduling may fail. For example:

```
tolerations:
- operator: "Exists"
```

In this example, the **operator** parameter is set to **Exists**. In this case, the **value** parameter cannot be configured.

- If the **operator** parameter of a toleration is set to **Exists** but the **key** parameter is empty, the toleration can match any key, value, and effect. It can tolerate any taint.
- If the **effect** parameter of a toleration is empty but the **key** parameter is configured, the toleration can match the effects of all keys.

For details, see **Taints and Tolerations**.

Restore the default tolerations configuration by modifying the YAML file of the workload as follows:

```
tolerations:
  - key: node.kubernetes.io/not-ready
    operator: Exists
    effect: NoExecute
    tolerationSeconds: 300
  - key: node.kubernetes.io/unreachable
    operator: Exists
    effect: NoExecute
    tolerationSeconds: 300
```

This default toleration indicates that the pod can run on the node with the preceding taints for 300s and then be evicted.

6.4.7 How Do I Prevent a Non-GPU or Non-NPU Workload from Being Scheduled to a GPU or NPU Node?

Symptom

If there are GPU/NPU nodes and other types of nodes running in your cluster, the non-GPU/NPU workloads may be scheduled to the GPU/NPU nodes. In this case, the GPU/NPU resources cannot be used properly.

Possible Cause

The non-GPU/non-NPU workloads use the vCPUs and memory provided by the GPU/NPU nodes. The scheduler may schedule the non-GPU/NPU workloads to these nodes, even if the workloads do not claim to use the GPU/NPU nodes. This may result in the idle GPU/NPU resources.

Solution

Add taints to the GPU/NPU nodes and configure tolerations to prevent non-GPU/non-NPU workloads from being scheduled to these nodes.

- For the GPU/NPU workloads, add tolerations so that they can be scheduled to the GPU/NPU nodes.
- For the non-GPU/NPU workload, if tolerations are not configured, they cannot be scheduled to the GPU/NPU nodes.

The procedure is as follows:

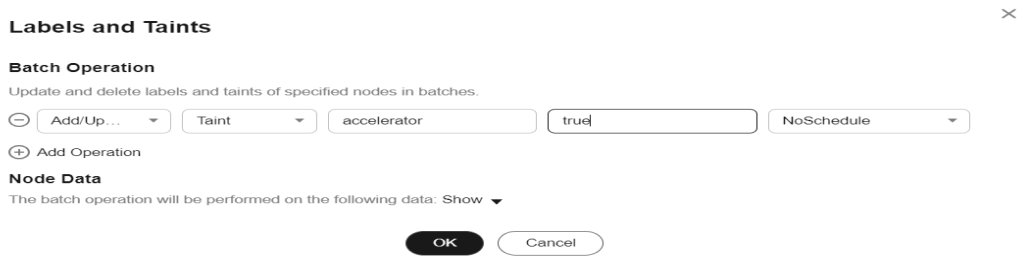
Step 1 Log in to the CCE console and click the cluster name to access the cluster console.

Step 2 In the navigation pane, choose **Nodes**. Click the **Nodes** tab, select a GPU/NPU node, and click **Labels and Taints** above the list.

Step 3 Click **Add Operation** under **Batch Operation** and add a taint to the node.

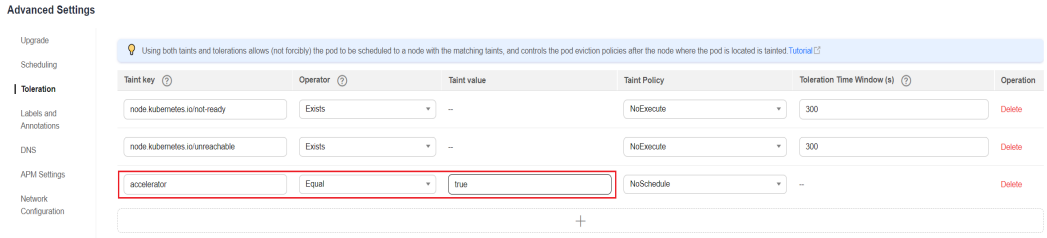
Select **Taint**. Enter the key and value and select the taint effect. The following example shows how to add the **accelerator=true:NoSchedule** taint to the GPU/NPU nodes.

Figure 6-7 Adding a taint



Step 4 When creating a GPU/NPU workload, manually add a toleration in the **Advanced Settings** area.

Figure 6-8 Adding a toleration



Step 5 When creating a non-GPU/NPU workload, do not add any tolerations. This workload will not be scheduled to the GPU/NPU nodes.

----End

6.4.8 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

6.4.9 What Should I Do If the Evicted Pods Are Scheduled Back to the Original Node Due to Changes in the kubelet Parameters?

Symptom

If a node experiences memory, disk, or PID pressure, it will be marked with a system taint. In such cases, if the configuration parameters of the kubelet in the node pool where the node belongs to are changed or if the kubelet of the node is restarted, the taint will be temporarily removed. As a result, the node, which previously had some pods evicted due to resource pressure, may be considered for scheduling again, and the pods will be rescheduled to that node. However, if the resource pressure on the node continues, the eviction process will be triggered once more.

Possible Cause

kubelet reports memory, disk, and PID pressure (heartbeats) based on the detection of eviction manager. This reporting process and detection are carried out by two goroutines simultaneously. Under normal circumstances, if the detection of the eviction manager happens before the heartbeat reporting, kubelet can accurately report the disk status without removing any taint. In abnormal cases where the heartbeat reporting occurs before the eviction manager detection, kubelet will remove the last taint.

Solution

There is no need for you to address this issue. The node will automatically remove the pods after a certain period of time.

6.4.10 How Do I Find the Pod That Is Using a GPU or NPU Based on the GPU or NPU Information?

When a GPU or NPU is used in a CCE cluster, it is not possible to directly obtain the pods that use the GPU or NPU. However, you can use the kubectl commands to obtain pods based on the GPU or NPU information. This allows for timely eviction of pods in case of GPU or NPU malfunction.

Prerequisites

- You have created a CCE cluster and configured kubectl for it. For details, see **Connecting to a Cluster Using kubectl**.
- You have installed CCE AI Suite (NVIDIA GPU) or CCE AI Suite (Ascend NPU) in the cluster. For details, see **CCE AI Suite (NVIDIA GPU)** and **CCE AI Suite (Ascend NPU)**. The NPU driver version must be later than 23.0.

Procedure

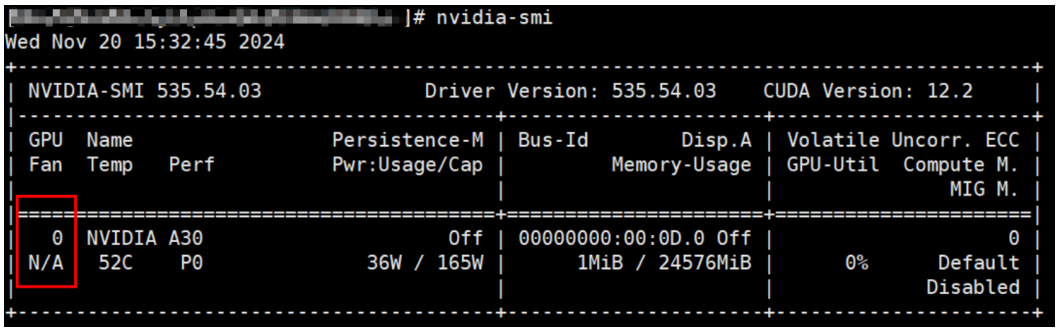
To find the pod that is using a GPU or NPU, obtain the GPU or NPU information on the cluster node and use kubectl to search for the corresponding pod.

Locating the Pod That Uses a GPU

Step 1 Log in to the CCE console and click the cluster name to access the cluster console. In the navigation pane, choose **Nodes**. In the right pane, click the **Nodes** tab and view the IP address of the GPU node. The following uses **192.168.0.106** as an example.

Step 2 Log in to the GPU node and view the GPU information:

```
nvidia-smi
```

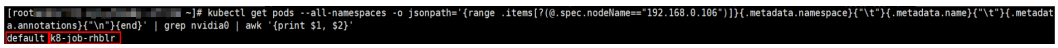


The command output shows that **GPU0** is present. The **GPU0** is used as an example to describe how to locate the pod that uses this GPU.

Step 3 Find the pod that uses the GPU based on the node IP address (192.168.0.106) and device ID (GPU 0).

```
kubectl get pods --all-namespaces -o jsonpath='{range .items[?(@.spec.nodeName=="192.168.0.106")]}{.metadata.namespace}{"\t"}{.metadata.name}{"\t"}{.metadata.annotations}{"\n"}{end}' | grep nvidia0 | awk '{print $1, $2}'
```

This command looks for all pods on the node with the IP address 192.168.0.106 and the pod whose annotation includes **nvidia0** (GPU 0). It displays the namespace and pod name of that specific pod.



Based on the information provided, it can be inferred that the pod named **k8-job-rhblr** in the **default** namespace is using GPU 0 on the node with the IP address 192.168.0.106.

----End

Locating the Pod That Uses an NPU

Step 1 Log in to the CCE console and click the cluster name to access the cluster console. In the navigation pane, choose **Nodes**. In the right pane, click the **Nodes** tab and view the IP address of the NPU node. The following uses **192.168.0.138** as an example.

Step 2 Log in to the NPU node and view the NPU information:

```
npu-smi info
```



```
+------------------------------------------------------------------------------------------------+
| npu-smi 23.0.rc3.6                       Version: 23.0.rc3.6                                   |
+------------------------------+-----------------+----------------------------------------------+
| NPU     Name                 | Health          | Power(W)     Temp(C)           Hugepages-Usage(page) |
| Chip    Device               | Bus-Id          | AICore(%)    Memory-Usage(MB)                        |
+==============================+=================+==============================================+
| 104                          | OK              | 12.8         48                0     / 969          |
| 0       0                    | 0000:00:0D.0    | 0            574  / 7759                            |
+==============================+=================+==============================================+
| 112     310                  | OK              | 12.8         45                0     / 969          |
| 0       1                    | 0000:00:0E.0    | 0            574  / 7759                            |
+==============================+=================+==============================================+
+------------------------------+-----------------+----------------------------------------------+
| NPU     Chip                 | Process id      | Process name             | Process memory(MB)  |
+==============================+=================+==============================================+
| No running processes found in NPU 104                                                          |
+==============================+=================+==============================================+
| No running processes found in NPU 112                                                          |
+==============================+=================+==============================================+
```

The command output shows that **device0** and **device1** are present. The **device0** is used as an example to describe how to locate the pod that uses this NPU.

Step 3 Find the pod that uses the NPU based on the node IP address (192.168.0.138) and device ID (NPU 0).

```
kubectl get pods --all-namespaces -o jsonpath='{range .items[?(@.spec.nodeName=="192.168.0.138")]}{.metadata.namespace}{"\t"}{.metadata.name}{"\t"}{.metadata.annotations}{"\n"}{end}' | grep Ascend310-0 | awk '{print $1, $2}'
```

This command looks for all pods on the node with the IP address 192.168.0.138 and the pod whose annotation includes **Ascend310-0** (NPU 0). It displays the namespace and pod name of that specific pod.

```
[root@ ~]# kubectl get pods --all-namespaces -o jsonpath='{range .items[?(@.spec.nodeName=="192.168.0.138")]}{.metadata.namespace}{"\t"}{.metadata.name}{"\t"}{.metadata.annotations}{"\n"}{end}' | grep Ascend310-0 | awk '{print $1, $2}'
default test-564f996c77-fws6z
```

Based on the information provided, it can be inferred that the pod named **test-564f996c77-fws6z** in the **default** namespace is using NPU 0 on the node with the IP address 192.168.0.138.

NOTE

- If other NPUs are being used, simply replace the name **Ascend 310** in **Ascend 310-0** with the appropriate name of the corresponding NPU.
- The NPU driver version must be 23.0 or later.

----End

6.4.11 How Do I Troubleshoot a Pod Exit Caused by a Node Label Update?

Symptom

After a workload pod is scheduled on a node based on the node labels, any changes to the labels or kubelet restarts due to configuration changes lead to a fault. About 30 seconds after the kubelet restart, you may notice a **Predicate NodeAffinity failed** event in the workload status. Additionally, the pod status changed to **Completed**, and a new pod will be scheduled on an appropriate node. If there are no nodes that match the specified label required by the pod, the pod scheduling will fail.

Possible Cause

Once a workload pod is configured with label affinity, it will only be scheduled to a node with that label. If the node label is changed after the pod is scheduled on

the node, the pod will still run on the node. However, if kubelet is restarted, it will verify if the pod affinity policy matches the node label. If it is not, kubelet will mark the pod as **Completed**. Then, it will create a pod and schedule it.

The main issue here stems from Kubernetes' native mechanism. To maintain pod stability, you are advised to match the label of a pod with the label of the node where the pod runs. This prevents the pod from being terminated during affinity policy checks when kubelet restarts, or a new pod from encountering issues with scheduling due to mismatched affinity policies and node labels after creation.

Solution

- When you only need to add a new Kubernetes label to a node or node pool, avoid removing any existing Kubernetes labels from it. This ensures that the affinity policies remain unchanged so that the existing pods can continue running on the node.
- If you need to change or delete a Kubernetes label on a node or node pool, but the new label does not align with the affinity policies of the pods running on the node, take the following steps to verify if there are pods with affinity scheduling configured on the node: (If no check is done, there may be a problem with pod scheduling when kubelet restarts.)

  a. Check whether an affinity policy has been configured for the workloads on a node.

  - If the node is in the default node pool (**DefaultPool**), run the following command and replace *<node-IP-address>* with the actual one:

    ```
    nodeIP='<node-IP-address>' && kubectl get pod --all-namespaces -o=custom-columns=nodeIP:'status.hostIP',nodeAffinity:'spec.affinity.nodeAffinity.requiredDuringSchedulingIgnoredDuringExecution.nodeSelectorTerms' | sed '1d' | grep $nodeIP | awk '{print substr($0, index($0,$2))}' | grep matchExpressions | uniq
    ```

    - If the command output is empty, it indicates that there is no affinity policy configured for any workload on the node, so the workload pods will not be rescheduled because of a new node label.
    - If the command output is not empty, it means there is an affinity policy configured for some workloads on the node. The following shows an example, and it indicates that there is a **tag1=value1** label configured for some workloads.

      ```
      [root@192-18-0-12 paas]# nodeIP='192.18.0.22' && kubectl get pod --all-namespaces -o=custom-columns=nodeIP:'
      DuringExecution.nodeSelectorTerms' | sed '1d' | grep -E $nodeIP | awk '{print substr($0, index($0,$2))}' | g
      [map[matchExpressions:[map[key:tag1 operator:In values:[value1]]]]]
      ```

  - If the node is in a custom node pool, run the following command and replace *<node-pool-name>* with the actual one:

    ```
    nodeIPs=$(kubectl get nodes -l cce.cloud.com/cce-nodepool=<node-pool-name> -o=custom-columns=IP:'metadata.annotations.alpha\.kubernetes\.io\/provided-node-ip' | sed '1d' | tr '\n' '|' | sed 's/|$//') && kubectl get pod --all-namespaces -o=custom-columns=nodeIP:'status.hostIP',nodeAffinity:'spec.affinity.nodeAffinity.requiredDuringSchedulingIgnoredDuringExecution.nodeSelectorTerms' | sed '1d' | grep -E $nodeIPs | awk '{print substr($0, index($0,$2))}' | grep matchExpressions | uniq
    ```

    - If the command output is empty, it indicates that there is no affinity policy configured for any workload in the node pool, so the workload pods will not be rescheduled because of a new node label.

- If the command output is not empty, it means there is an affinity policy configured for some workloads in the node pool. The following shows an example, and it indicates that there is a **tag1=value1** label configured for some workloads.

```
[root@192-168-0-217 paas]# nodeIPs=$(kubectl get nodes -l cce.cloud.com/cce-nodepool=
-ip' | sed '1d' | tr '\n' '|' | sed 's/|$//') && kubectl get pod --all-namespaces -o=
DuringExecution.nodeSelectorTerms' | sed '1d' | grep -E $nodeIPs | awk '{print substr
[map[matchExpressions:[map[key:tag1 operator:In values:[value1]]]]]
```

b. Get all workloads that are associated with the affinity policy. The value of *affinity-policy* is the one obtained in the previous step.

```
match='affinity-policy' && kubectl get deployment,statefulset,daemonset,job,cronjob --all-namespaces -o=custom-columns=kind:'kind',name:'metadata.namespace',namespace:'metadata.name',nodeAffinity:'spec.template.spec.affinity.nodeAffinity.requiredDuringSchedulingIgnoredDuringExecution.nodeSelectorTerms' | grep -F "$match" | awk '{print "kind:" $1 ",namespace:" $2 ",name:" $3}'
```

As shown in the following figure, the workloads with the **tag1=value1** label are displayed.

```
[root@192-168-0-217 paas]# match='[map[matchExpressions:[map[key:tag1 operator:In values:[value1]]]]]' && kubectl get deployment
ns=kind:'kind',name:'metadata.namespace',namespace:'metadata.name',nodeAffinity:'spec.template.spec.affinity.nodeAffinity.requir
"$match" | awk '{print "kind:" $1 ",namespace:" $2 ",name:" $3}'
kind:Deployment,namespace:default,name:nginx
```

c. Evaluate the impact of the workload restart on services before changing any label on the node or node pool.

You can also modify the affinity policy of a workload to schedule it to other nodes and change the node label after the workload pod runs properly. For details, see **Configuring Node Affinity Scheduling (nodeAffinity)**.

6.4.12 Why Do a Large Number of Pods Fail to Be Executed After a Workload That Uses Even Scheduling on Virtual GPUs Is Created?

Symptom

After a workload that uses even scheduling on virtual GPUs is created, a large number of GPU pods fail to be executed. The following is an example:

```
kubectl get pods
```

Information similar to the following is displayed:

```
NAME                     READY   STATUS                    RESTARTS   AGE
test-586cf9464c-54g4s    0/1     UnexpectedAdmissionError  0          57s
test-586cf9464c-58n6d    0/1     UnexpectedAdmissionError  0          10s
test1-689cf9462f-5bzcv   0/1     UnexpectedAdmissionError  0          58s
...
```

Possible Cause

For even scheduling on virtual GPUs, the cluster version must be compatible with the CCE AI Suite (NVIDIA GPU) add-on version. Compatibility details are as follows:

- If the add-on version is 2.1.41 or later, the cluster version must be v1.27.16-r20, v1.28.15-r10, v1.29.10-r10, v1.30.6-r10, v1.31.4-r0, or later.
- If the add-on version is 2.7.57 or later, the cluster version must be v1.28.15-r10, v1.29.10-r10, v1.30.6-r10, v1.31.4-r0, or later.

When the cluster version is incompatible but the GPU virtualization resources on the nodes are still available, pod scheduling proceeds normally. However, once resources are exhausted, kubelet fails to process even scheduling on virtual GPUs properly. This causes pods that failed to be scheduled to stack, leading to memory leakage and batch pod execution failures.

Solution

- To continue using even scheduling on virtual GPUs, delete the workload that failed to be executed and upgrade the cluster to the required version. For details, see **Cluster Upgrade Overview**. After the cluster is upgraded, create the workload again. Then, the workload can be scheduled properly.
- If you do not need even scheduling on virtual GPUs, restart the kubelet component on the affected nodes to restore performance. The process is as follows:
  a. Check the nodes that have faulty pods and record the node IP addresses. You will need these IP addresses to restore the kubelet component on the nodes.
  ```
  kubectl get pod -l volcano.sh/gpu-num  -owide
  ```
  Information similar to the following is displayed:
  ```
  NAME                      READY   STATUS                   RESTARTS   AGE     IP           NODE
  test-586cf9464c-54g4s     0/1     UnexpectedAdmissionError   0          5m57s   <none>
  11.84.252.4
  test-586cf9464c-58n6d     0/1     UnexpectedAdmissionError   0          5m10s   172.19.0.24
  11.84.252.4
  test-586cf9464c-5bzcv     0/1     UnexpectedAdmissionError   0          5m58s   <none>
  11.84.252.4
  test-586cf9464c-6bb5d     0/1     UnexpectedAdmissionError   0          6m15s   <none>
  11.84.252.4
  test-586cf9464c-6r2bq     0/1     UnexpectedAdmissionError   0          5m11s   <none>
  11.84.252.4
  test-586cf9464c-6rcpl     0/1     UnexpectedAdmissionError   0          6m11s   172.19.0.21
  11.84.252.4
  ...
  ```
  b. Delete the workload that uses even scheduling on virtual GPUs. You need to replace *deployment* with the corresponding workload type and replace *test* with the corresponding workload name in the following command.
  ```
  kubectl delete deployment test  # Delete the test Deployment.
  ```
  Information similar to the following is displayed:
  ```
  deployment.apps/test deleted
  ```
  c. Log in to the nodes involved in **a** one by one and restart kubelet.
  ```
  systemctl restart kubelet
  ```
  Enter the node password in the command output. If no error message is displayed, the node has been restarted. After the restart, pods that do not use even scheduling on virtual GPUs and previously failed to be scheduled should now be scheduled successfully.

6.5 Others

6.5.1 What Should I Do If a Cron Job Cannot Be Restarted After Being Stopped for a Period of Time?

When a cron job is paused mid-execution and later resumed, the controller checks the number of missed scheduling times between the last scheduled time and the

current time. If this number exceeds 100, the controller will not start the job and logs the error. For details, see **CronJob limitations**.

```
Cannot determine if job needs to be started. Too many missed start time (> 100). Set or
decrease .spec.startingDeadlineSeconds or check clock skew.
```

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.

6.5.2 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.

6.5.3 What Should I Do If Error Message "Auth is empty" Is Displayed When a Private Image Is Pulled?

Symptom

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
```


6.5.4 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 **IfNotPreset**:
   a. When the required image does not exist locally, it will be pulled from the remote repository.
   b. When 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.

6.5.5 Why Is the Mount Point of a Docker Container in the Kunpeng Cluster Uninstalled?

Symptom

The mount point of a Docker container in the Kunpeng cluster is uninstalled.

Possible Cause

If the Kunpeng cluster node runs EulerOS 2.8 and the **MountFlags=shared** field is configured in the Docker service file, the container mount point will be uninstalled due to the systemd feature.

Solution

Modify the Docker file, delete the **MountFlags=shared** field, and restart Docker.

Step 1 Log in to the node.

Step 2 Run the following command to delete the **MountFlags=shared** field from the configuration file and save the file:

vi /usr/lib/systemd/system/docker.service


```
[Service]
MountFlags=shared
Type=notify
EnvironmentFile=-/etc/sysconfig/docker
EnvironmentFile=-/etc/sysconfig/docker-storage
EnvironmentFile=-/etc/sysconfig/docker-network
Environment=GOTRACEBACK=crash
```

Step 3 Run the following command to restart Docker:

systemctl restart docker

----End

6.5.6 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-6c69469cf4-nfs7f to 10.14.11.139
  Normal   SuccessfulMountVolume        55s                kubelet                Successfully mounted volumes for pod "prometheus-server-6c69469cf4-nfs7f_cattle-prometheus(48ac202a-649a-429c-91ca-573dbaabcb72)"
  Normal   SuccessfulUpdateSecurityGroup  52s                yangtse-controller     Successfully updated security group to "e0a07f89-6fd6-431a-b901-ed0758b5198c"
  Normal   Pulled                       0s (x6 over 51s)   kubelet                Container image "100.125.0.29:26202/f0a_mve%/e-y/busybox:1.29.2" already present on machine
  Warning  FailedCreate                 7s (x6 over 50s)   kubelet                Error: failed to create containerd container: error unpacking image: failed to extract layer sha256:f9d9e4e6e2f0689cd752390e14ade48b0ec6f2
a488a05af5ab2f9ccaf54c299d: failed to get reader from content store: content digest sha256:8c5a7da1afbc602695fcb2cd6445743cec5ff32053ea589ea9bd8773b7068185: 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 image is not pulled.

Solution

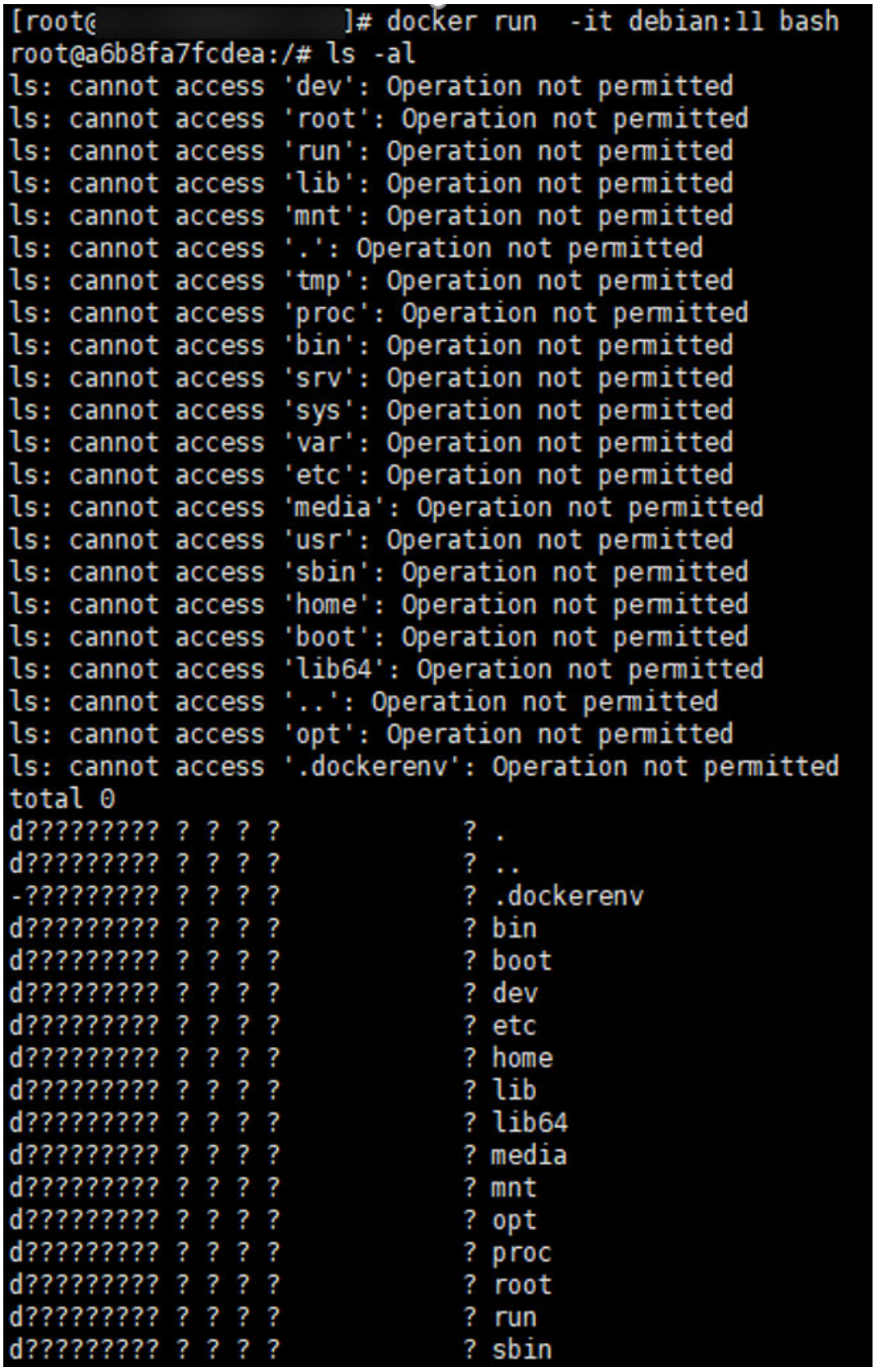
You can use either of the following methods to solve this problem:

- 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.

6.5.7 Why the File Permission and User in the Container Are Question Marks?

Symptom

If the node OS is CentOS 7.6 or EulerOS 2.5 and the Debian GNU/Linux 11 (bullseye) kernel is used as the base image container, exceptions occur on file permissions and users.



Impact

Exceptions occur on file permissions and users in a container.

Solution

CCE provides two solutions:

- Use Debian 9 or 10 as the base image of the service container.
- Use EulerOS 2.9 or Ubuntu 18.04 as the node OS.

7 Networking

7.1 Network Exception Troubleshooting

7.1.1 How Do I Locate a Workload Networking Fault?

Troubleshooting

The issues here are described in order of how likely they are to occur.

If the fault persists after you have ruled out one cause, move on to the next one.

- **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 obtain the pod IP address. Then, log in to the node or the pod and run **curl** to manually call the API and check whether the expected result is returned.

If *{Container IP address}*:*{Port number}* is not accessible, log in to the service container, and attempt to access **127.0.0.1**:*{Port number}*.

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 and 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*}**.
- CCE Turbo cluster:

  The security group name of the node is **{*Cluster name*}-cce-node-{*Random characters*}**.

  The name of the security group associated with the containers is **{*Cluster name*}-cce-eni-{*Random characters*}**.

Check the following:

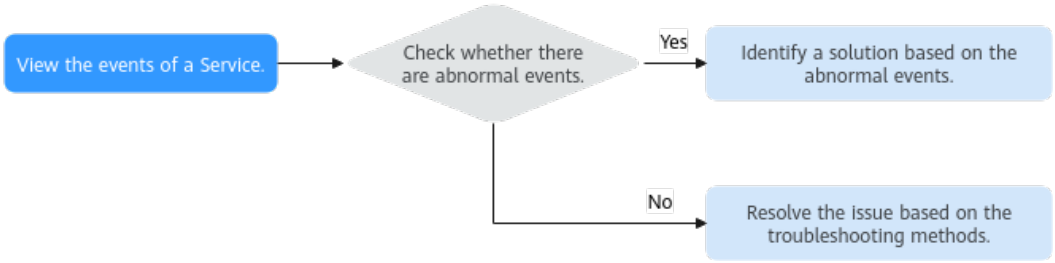
- IP address, port, and protocol of an external request to access the workloads in the cluster. They must be allowed in the inbound rule of the cluster security group.
- IP address, port, and protocol of a request sent by a workload to visit external applications outside the cluster. They must be allowed in the outbound rule of the cluster security group.

For details about security group configuration, see **How Can I Configure a Security Group Rule for a Cluster?**

7.1.2 How Do I Resolve Issues with a LoadBalancer Service?

LoadBalancer Services offer services through load balancers. When you create a LoadBalancer Service, you have the option to choose an existing load balancer or have one created automatically. If you choose an existing load balancer, CCE will assign resources like the load balancer listeners and backend server groups to the Service. If you choose to automatically create a load balancer, CCE will assign resources like the load balancer listeners and backend server groups to the Service and create a load balancer. The following describes how to resolve issues with a LoadBalancer Service.

Troubleshooting Process



1. Log in to the CCE console.
2. Click the cluster name to access the cluster console. In the navigation pane, choose **Services & Ingresses**.
3. Locate the row containing the target Services and click **View Events** in the Operation column to check whether there is any abnormal event.
   - If there is, you can resolve the problem based on the events and **Troubleshooting Based on the Abnormal Events**.
   - If there is not, the problem may be due to access issues or peripheral service configurations. You can resolve the problem by referring to **Troubleshooting Based on the Common Problems**.

Troubleshooting Based on the Abnormal Events

The following table lists the solutions to different abnormal events.

Error Information	Description	Solution
Quota exceeded for resources: loadbalancer	The load balancer fails to be automatically created for the Service due to insufficient quota of load balancers.	• Apply for a higher quota. The default quota is 100. For details about how to obtain the quota details and increase the quota, see **How Do I Increase My Quota?** • Determine if the load balancer can be reused because multiple Services can use the same load balancer.
Quota exceeded for resources: listener	Listeners failed to be created for the load balancer associated with the Service due to insufficient quota of listeners.	Apply for a higher quota. The default quota is 200. For details about how to obtain the quota details and increase the quota, see **How Do I Increase My Quota?**

Error Information	Description	Solution
can't find backend pod with service	The Service does not have any associated backend pods. You need to make sure that there are pods associated with the Service and that they are functioning properly.	• If there is no pod associated with Service, associate some backend pods with it. • If the associated pods are not functioning properly, rectify the pod fault. For details, see **How Can I Locate the Root Cause If a Workload Is Abnormal?**
failed to ensure load balancer: Not Found	The load balancer associated with the Service is not present.	Log in to the ELB console. In the corresponding region, search for the load balancer based on the value of **kubernetes.io/elb.id** in **annotations** of the Service. If the needed load balancer is not present, create a Service as needed and choose another existing load balancer or create a new one automatically for the Service.
Update loadbalancer of service([ServiceName]/[Namespace]) error: listener is empty	The listener of the load balancer associated with the Service may be deleted by mistake.	Log in to the ELB console. In the corresponding region, search for the load balancer based on the value of **kubernetes.io/elb.id** in **annotations** of the service and check whether the listener is present based on the port. If the specified listener of the load balancer is not present, create a Service as needed.

Error Information	Description	Solution
Failed to CreateListener : request failed: {"error_msg":"Load Balancer [ELB id] already has a listener with protocol_port of [port number].","error_code":"ELB.8907","request_id":"xxx"}, status code: 409	A listener port conflict occurs. Another listener is already using that port on ELB.	Update the Service and configure another available port.
Failed to create member : {"error_msg":"Vpc [cluster-VPC-ID] of member's subnet_cidr [subnet-to-which-the-listener-belongs] and vpc [ELB-VPC-ID] of loadbalancer [ELB ID] mismatch","error_code":"ELB.8902","request_id":"xxx"}, status code: 400	When a Service is created, it has been associated with an existing load balancer. However, the load balancer is in a different VPC from the cluster.	Ensure that the selected load balancer is in the same VPC as the cluster.
Failed to CreateListener : request failed: {"error_msg":"Loadbalancer [ELB id] has no flavor of type L7_elastic and cannot create listeners of type L7.","error_code":"ELB.8907","request_id":"xxx"}, status code: 409	Layer-7 listeners cannot be created for the network load balancers.	Select a load balancer of the required specifications. To create an HTTP/HTTPS Layer-7 listener, use a load balancer with application load balancing.
Failed to CreateListener : request failed: {"error_msg":"Loadbalancer [ELB id] has only flavor of type l7 and cannot create listeners of type l4.","error_code":"ELB.8907","request_id":"xxx"}, status code: 409	Layer 4 listeners cannot be created for the application load balancers.	Select a load balancer of the required specifications. To create a TCP/UDP Layer 4 listener, use a load balancer with network load balancing.

Troubleshooting Based on the Common Problems

If there are no abnormal events, resolve the problems by referring to the following table.

Category	Symptom	Reference
ELB access	The load balancer is experiencing an imbalance in its load.	This symptom is due to the improper configuration of the backend routing rules. For details, see **Imbalance Load of a Load Balancer**.
	A load balancer is inaccessible from within a 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). For details, see **Why Can't the ELB Address Be Used to Access Workloads in a Cluster?**
	A load balancer is inaccessible from outside a cluster.	The load balancer or its backend servers are configured improperly. For details, see **An Inaccessible Load Balancer from Outside a Cluster**.
ELB configuration	The configuration of a load balancer is changed.	**Why Was the Configuration of a Load Balancer Changed?**
	The backend of a load balancer is automatically deleted.	**Why Is the Backend Server Group of an ELB Automatically Deleted After a Service Is Published to the ELB?**
ELB deletion issue	The load balancer associated with the Service is not deleted when the Service is deleted.	ELB uses different automatic deletion policies in different scenarios. For details, see **Load Balancer Deletion Policies**.

Imbalance Load of a Load Balancer

Possible Cause

The backend routing rules of the load balancer are not properly configured.

Solution

- For a Service whose **externalTrafficPolicy** is **Local**, set the backend routing rules of the load balancer to the weighted round robin, which means, add the **kubernetes.io/elb.lb-algorithm: ROUND_ROBIN** annotation to the Service.
- For a service that uses a persistent connection, set the backend routing rules of the load balancer to the weighted least connections, which means, add the **kubernetes.io/elb.lb-algorithm: LEAST_CONNECTIONS** annotation to the Service.

An Inaccessible Load Balancer from Outside a Cluster

Possible Cause

The ELB configuration is incorrect or there is an issue with the backend servers.

Procedure

1. View the Service events and handle the abnormal events: (For details, see **Troubleshooting Based on the Abnormal Events**.)
   ```
   kubectl -n {your-namespace} describe svc {your-svc-name}
   ```
2. Check whether the backend server group of the port or URL meets the expectation.

   If there is no backend server group, check whether the service pods are functioning properly. If the associated pods are not functioning properly, rectify the pod fault. For details, see **How Can I Locate the Root Cause If a Workload Is Abnormal?**

Why Was the Configuration of a Load Balancer Changed?

Possible Cause

Under certain conditions, CCE automatically updates the load balancer configuration based on the configuration of the Service with which the load balancer associated. Any changes made to the load balancer configuration on the ELB console may be overwritten.

Solution

Configure load balancers associated with the LoadBalancer Services using annotations. For details about how to configure annotations, see **Configuring LoadBalancer Services Using Annotations**.

NOTICE

Do not manually change any configurations of a load balancer created and managed by CCE clusters on the ELB console. Doing so may result in the loss of configuration and render the Service inaccessible.

Load Balancer Deletion Policies

The automatic deletion policies of the load balancers when their associated Services are deleted are as follows:

- For a load balancer that is automatically created by its associated Service:
  - If the load balancer is not being used by any other Services, it will be automatically deleted when the original Service is deleted.
  - If there are other Services associated with the load balancer, it will not be deleted when the original Service is deleted.
- If an existing load balancer is selected for a Service, the original load balancer will not be deleted when the Service is deleted.

7.1.3 Why Can't the ELB Address Be Used to Access Workloads in a Cluster?

Symptom

In a cluster (on a node or in a container), the ELB address cannot be used to access workloads.

Possible Cause

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.

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.
- In a CCE Turbo cluster that utilizes Cloud Native Network 2.0, node-level affinity is supported only when the Service backend is connected to a hostNetwork pod.
- The table lists only the scenarios where the access may fail. Other scenarios that are not listed in the table indicate that the access is normal.

Service Type Released on the Server	Access Type	Request Initiation Location on the Client	Tunnel Network Cluster (IPVS)	VPC Network Cluster (IPVS)	Tunnel Network Cluster (iptables)	VPC Network Cluster (iptables)
NodePort Service	Public/ Private network	Same node as the service pod	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.

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	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.	The access is successful.	The access is successful.

Service Type Released on the Server	Access Type	Request Initiation Location on the Client	Tunnel Network Cluster (IPVS)	VPC Network Cluster (IPVS)	Tunnel Network Cluster (iptables)	VPC Network Cluster (iptables)
		Other containers on the same node as the service pod	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.	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.	The access failed.

Service Type Released on the Server	Access Type	Request Initiation Location on the Client	Tunnel Network Cluster (IPVS)	VPC Network Cluster (IPVS)	Tunnel Network Cluster (iptables)	VPC Network Cluster (iptables)
		Other containers on different nodes from the service pod	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 shared load balancer	Private network	Same node as the service pod	The access failed.	The access failed.	The access failed.	The access failed.

Service Type Released on the Server	Access Type	Request Initiation Location on the Client	Tunnel Network Cluster (IPVS)	VPC Network Cluster (IPVS)	Tunnel Network Cluster (iptables)	VPC Network Cluster (iptables)
		Other containers on the same node as the service pod	The access failed.	The access failed.	The access failed.	The access failed.
DNAT gateway Service	Public network	Same node as the service pod	The access failed.	The access failed.	The access failed.	The access failed.
		Different nodes from the service pod	The access failed.	The access failed.	The access failed.	The access failed.
		Other containers on the same node as the service pod	The access failed.	The access failed.	The access failed.	The access failed.
		Other containers on different nodes from the service pod	The access failed.	The access failed.	The access failed.	The access failed.

Service Type Released on the Server	Access Type	Request Initiation Location on the Client	Tunnel Network Cluster (IPVS)	VPC Network Cluster (IPVS)	Tunnel Network Cluster (iptables)	VPC Network Cluster (iptables)
LoadBalancer Service using a Dedicated load balancer (Local) for interconnection with NGINX Ingress Controller	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.

Solution

The following methods can be used to solve this problem:

- (**Recommended**) In the cluster, use the ClusterIP Service or service domain name for access.
- Set **externalTrafficPolicy** of the Service to **Cluster**, which means cluster-level service affinity. Note that this affects source address persistence.

```
apiVersion: v1
kind: Service
metadata:
  annotations:
    kubernetes.io/elb.class: union
    kubernetes.io/elb.autocreate: '{"type":"public","bandwidth_name":"cce-
bandwidth","bandwidth_chargemode":"traffic","bandwidth_size":5,"bandwidth_sharetype":"PER","eip_t
ype":"5_bgp","name":"james"}'
  labels:
    app: nginx
  name: nginx
spec:
  externalTrafficPolicy: Cluster
  ports:
  - name: service0
    port: 80
    protocol: TCP
    targetPort: 80
  selector:
    app: nginx
  type: LoadBalancer
```


- Leveraging the pass-through feature of the Service, kube-proxy is bypassed when the ELB address is used for access. The ELB load balancer is accessed first, and then the workload.

  NOTE

  - In a CCE standard cluster, after passthrough networking is configured using a dedicated load balancer, the private IP address of the load balancer cannot be accessed from the node where the workload pod resides or other pods on the same node as the workload.
  - Passthrough networking is not supported for clusters of v1.15 or earlier.
  - In IPVS network mode, the passthrough settings of Services connected to the same load balancer must be the same.
  - If node-level (local) service affinity is used, **kubernetes.io/elb.pass-through** is automatically set to **onlyLocal** to enable pass-through.

```
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":"traffic","bandwidth_size":5,"bandwidth_sharetype":"PER","eip_t
ype":"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
```

7.1.4 Why Can't the Ingress Be Accessed Outside the Cluster?

Ingresses forward requests based on layer-7 HTTP and HTTPS protocols. As an entry of cluster traffic, ingresses use domain names and paths to achieve finer granularities. After an ingress is added to a cluster, the cluster may fail to be accessed. This section describes how to locate the fault when an ingress fails to be added or cannot be accessed. Before rectifying ingress issues, read the following precautions and perform a self-check:

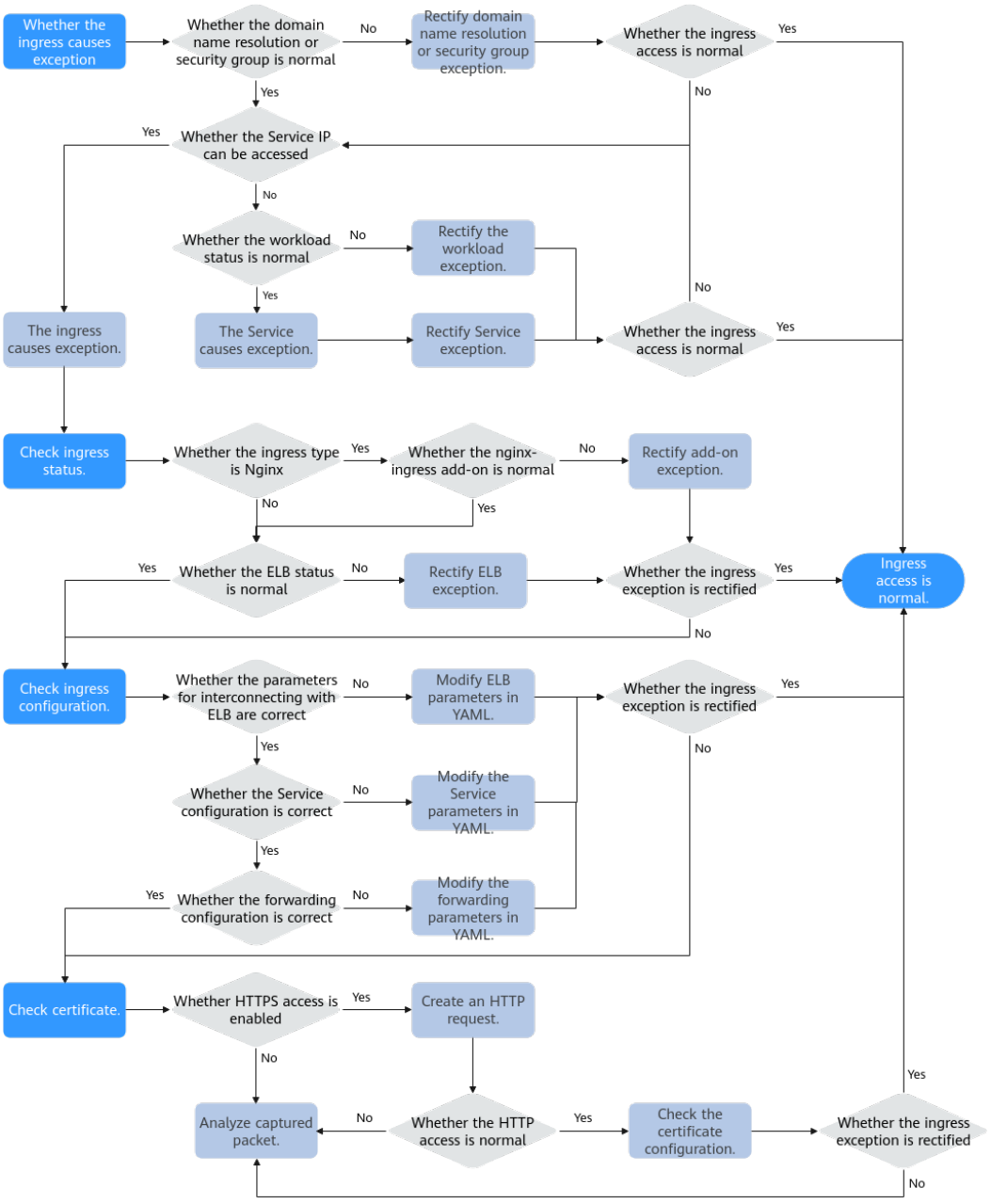
NOTICE

- If the host address is specified in the ingress, the IP address cannot be used for access.
- Check the node security group of the cluster and ensure that the service ports in the range of 30000 to 32767 are accessible to all network segments for inbound traffic.

Fault Locating

This section provides an overview of troubleshooting ingress external access exceptions, as shown in **Figure 7-1**.

Figure 7-1 Fault locating overview



1. **Checking Whether the Exception Is Caused by the Ingress**

   Check whether the problem is caused by the ingress. Ensure that the external domain name resolution is normal, the security group rules are correct, and the service and workload corresponding to the ingress are working properly.

2. **Checking the Ingress Status**

When the service and workload are normal, ensure that the load balancer on which the ingress depends is normal. If the ingress is of the Nginx type, ensure that the NGINX Ingress Controller add-on runs properly.

3. **Checking Whether the Ingress Is Configured Correctly**

   If the preceding check results are normal, the ingress configuration may be incorrect.

   - Check whether the parameters for interconnecting with the load balancer are correct.
   - Check whether the Service configuration is correct.
   - Check whether the forwarding configuration is correct.

4. **Checking Certificate**

   If HTTPS access is enabled on the ingress, you also need to check whether the fault is caused by incorrect certificate configuration. You can use the same load balancer to create an HTTP ingress. If the access is normal, the HTTPS certificate may be faulty.

5. If the fault persists, capture packets for analysis or submit a service ticket for help.

Checking Whether the Exception Is Caused by the Ingress

Check whether the access exception is caused by the ingress. If the domain name resolution exception, security group rule error, service exception, or workload exception occurs, the ingress access may fail.

The following check sequence complies with the rules from outside to inside:

Step 1 Check whether the domain name resolution or security group rules are normal.

1. Run the following command to check whether record sets of the domain name take effect on the authoritative DNS server:
   nslookup -qt= *Type Domain name Authoritative DNS address*
2. Check the security group rules of the cluster nodes and ensure that the service ports in the 30000–32767 range are accessible to all network segments for inbound traffic. For details about how to harden the security group, see **How Can I Configure a Security Group Rule for a Cluster?**

Step 2 Check whether the Service can access services in the container.

You can create a pod in the cluster and use the cluster IP address to access the Service. If the Service type is NodePort, you can also use ***EIP:Port*** to access the service over the Internet.

1. Use kubectl to connect to the cluster and query the Service in the cluster.

```
# kubectl get svc
NAME         TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)        AGE
kubernetes   ClusterIP   10.247.0.1      <none>        443/TCP        34m
nginx        ClusterIP   10.247.138.227  <none>        80/TCP         30m
```

2. Create a pod and log in to the container.

```
kubectl run -i --tty --image nginx:alpine test --rm /bin/sh
```


3. Run the **curl** command to access *ClusterIP address:Port* of the Service to check whether the Service in the cluster is accessible.

```
curl 10.247.138.227:80
```

   If the Service can be accessed, the backend workload status is normal. It can be preliminarily determined that the exception is caused by the ingress. For details, see **Checking the Ingress Status**.

   If the Service access is abnormal, check the workload status to determine the cause.

Step 3 Check whether the workload status is normal.

If the workload is normal but the Service cannot be accessed, the exception may be caused by the Service. Check the Service configuration. For example, check whether the container port is correctly configured to an open service port of the container.

If the workload is normal but the access result is not as expected, check the service code running in the container.

----End

Checking the Ingress Status

CCE supports two types of ingresses. The Ngnix ingress controller is provided by the open-source community and needs to be maintained by installing the NGINX Ingress Controller add-on in the cluster. The LoadBalancer ingress controller runs on the master node and is maintained by a dedicated Huawei Cloud team.

Step 1 If you use an Nginx ingress, you need to install the NGINX Ingress Controller add-on in the cluster. If you use a LoadBalancer ingress, skip this step.

Go to **Add-ons** and check whether the NGINX Ingress Controller add-on runs properly. Ensure that there are enough node resources in the cluster. If not, the add-on pods cannot be scheduled to nodes.

Step 2 Go to the ELB console to check the ELB status.

- LoadBalancer ingress

  The access port can be customized. Check whether the listener and backend server group created on the ELB are not deleted or modified.

  When creating a LoadBalancer ingress, you can enable automatic load balancer creation on the console. Do not modify the automatically created load balancer to prevent ingress exceptions caused by the load balancer.

- Nginx ingress

  The access ports are fixed to 80 and 443. Custom ports are not supported. Installing the NGINX Ingress Controller add-on occupies both ports 80 and 443. Do not delete them. Otherwise, you need to reinstall the add-on.

You can also determine whether the fault is caused by the load balancer based on the error code. If the following error code is displayed, there is a high probability that the fault is caused by the load balancer. In this case, you need to pay special attention to the load balancer.

----End

Checking Whether the Ingress Is Configured Correctly

If the preceding check items are normal, check whether the exception is caused by parameter settings. When using kubectl to create an ingress, a large number of parameters need to be set, which is prone to errors. You are advised to use the console to create ingresses and set parameters as required to automatically filter out load balancers and Services that do not meet requirements. This effectively prevents incorrect formats or missing of key parameters.

Check the ingress configuration according to the following steps:

- **Check whether the parameters for interconnecting with the load balancer are correct.**

  Load balancers are defined by parameters in the **annotations** field. Kubernetes does not verify the parameters in the **annotations** field when creating resources. If key parameters are incorrect or missing, an ingress can be created but cannot be accessed.

  The following problems frequently occur:

  - The interconnected ELB load balancer is not in the **same VPC** as the cluster.
  - Key fields **kubernetes.io/elb.id**, **kubernetes.io/elb.ip**, **kubernetes.io/ingress.class**, and **kubernetes.io/elb.port** in **annotations** are missing when a LoadBalancer ingress is associated with an existing ELB load balancer.
  - When you add an Nginx ingress, the NGINX Ingress Controller add-on is not installed. As a result, the ELB connection is unavailable.
  - When you add an Nginx ingress, the **kubernetes.io/elb.port** field does not support custom ports. If HTTP is used, the value is fixed to **80**. If HTTPS is used, the value is fixed to **443**.

- **Check whether Service is configured correctly.**

  - Check whether the Service type connected to the ingress is correct. For details about the Service types supported by the ingress, see the following table.

Table 7-1 Services supported by LoadBalancer ingresses

Cluster Type	ELB Type	ClusterIP	NodePort
CCE standard cluster	Shared load balancer	Not supported	Supported
	Dedicated load balancer	Not supported	Supported
CCE Turbo cluster	Shared load balancer	Not supported	Supported
	Dedicated load balancer	Supported	Supported

Table 7-2 Services supported by Nginx ingress

Cluster Type	ELB Type	ClusterIP	NodePort
CCE standard cluster	Shared load balancer	Supported	Supported
	Dedicated load balancer	Supported	Supported
CCE Turbo cluster	Shared load balancer	Supported	Supported
	Dedicated load balancer	Supported	Supported

  - Check whether the access port number of the Service is correct. The access port number (**port** field) of the Service must be different from the container port number (**targetPort** field).

- **Check whether the forwarding configuration is correct.**
  - The forwarding URL added 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.

    NOTE

    When using the Nginx Ingress Controller, you can add the **rewrite** comment to the **annotations** field for redirection to rewrite the path that does not exist in the Service to avoid the error that the access path does not exist. For details, see **Rewrite**.
  - If the domain name (host) is specified when an ingress is created, the ingress cannot be accessed using an IP address.

Checking Certificate

The ingress secret certificate type of CCE is **IngressTLS** or **kubernetes.io/tls**. If the certificate type is incorrect, the ingress cannot create a listener on the load balancer. As a result, the ingress access becomes abnormal.

Step 1 Remove HTTPS parameters from YAML and create an HTTP ingress to check whether the ingress can be accessed.

If the HTTP access is normal, check whether the HTTPS secret certificate is correct.

Step 2 Check whether the secret type is correct. Check whether the secret type is **IngressTLS** or **kubernetes.io/tls**.

```
kubectl get secret
```

Information similar to the following is displayed:

```
NAME                  TYPE                                  DATA   AGE
ingress               IngressTLS                            2      36m
```

Step 3 Create test certificates to rectify the certificate fault.

```
openssl req -x509 -nodes -days 365 -newkey rsa:2048 -keyout tls.key -out tls.crt -subj "/
CN={YOUR_HOST}/O={YOUR_HOST}"
```

Step 4 Use the test certificates **tls.key** and **tls.crt** to create a secret and check whether the secret can be accessed normally. The following example shows how to create an IngressTLS secret.

Specifically, the IngressTLS secret is created using kubectl:

```
kind: Secret
apiVersion: v1
type: IngressTLS
metadata:
  name: ingress
  namespace: default
data:
  tls.crt: LS0tLS1CRU*****FURS0tLS0t
  tls.key: LS0tLS1CRU*****VZLS0tLS0=
```

NOTE

In the preceding information, **tls.crt** and **tls.key** are only examples. Replace them with the actual files. The values of **tls.crt** and **tls.key** are the content encrypted using Base64.

----End

7.1.5 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 7-2 404: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 7-3 404:nginx/1.**.*

7.1.6 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.

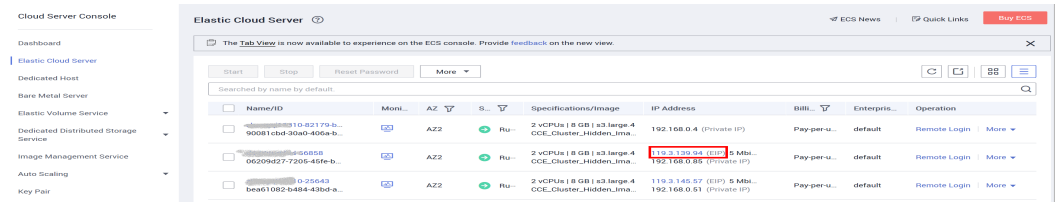
Check Item 1: Whether the Node Can Access the Internet

Step 1 Log in to the ECS console.

Step 2 Check whether an EIP has been bound to the ECS (node) or whether the ECS has a NAT gateway configured.

Figure 7-4 shows that an EIP has been bound. If no EIP is displayed, bind an EIP to the ECS.

Figure 7-4 Node with an EIP bound



----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 **cat /etc/resolv.conf** 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 correctly configured.

7.1.7 What Can I Do If a VPC Subnet Cannot Be Deleted?

A VPC subnet may fail to be deleted if you have used the VPC subnet in the CCE cluster. Therefore, you need to delete the corresponding cluster on the CCE console before deleting the VPC subnet.

NOTICE

- If you delete a cluster, all nodes, applications, and services in the cluster will be deleted. Exercise caution when deleting a cluster.
- You are not advised to delete nodes in a CCE cluster on the ECS page.

7.1.8 How Do I Restore a Faulty Container ENI?

If a container ENI is faulty, the container restarts repeatedly and cannot provide services for external systems. To rectify the fault, perform the following operations:

Procedure

Step 1 Run the following command to delete the pod of the faulty container:

kubectl delete pod {podName} -n {podNamespace}

Where,

- **{podName}**: Enter the name of the pod of the faulty container.
- **{podNamespace}**: Enter the namespace where the pod is located.

Step 2 After the pod of the faulty container is deleted, the system automatically recreates a pod for the container. In this way, the container ENI is restored.

----End

7.1.9 What Should I Do If a Node Fails to Access the Internet?

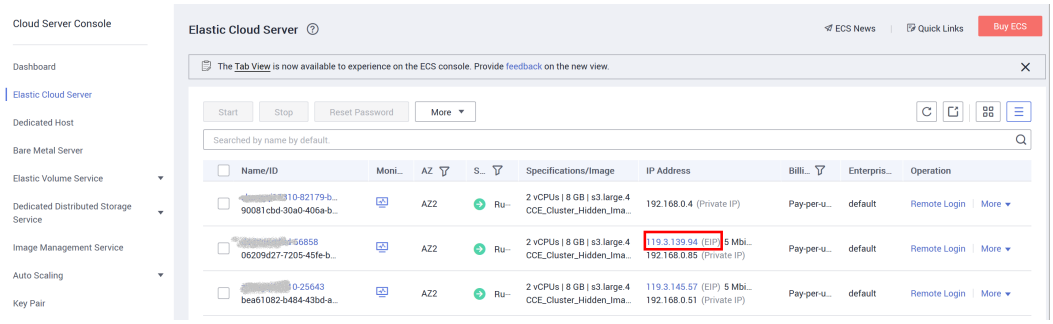
If a node cannot access the Internet, you can check the items described in this section and resolve the issue.

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.

Figure 7-5 Node with an EIP bound



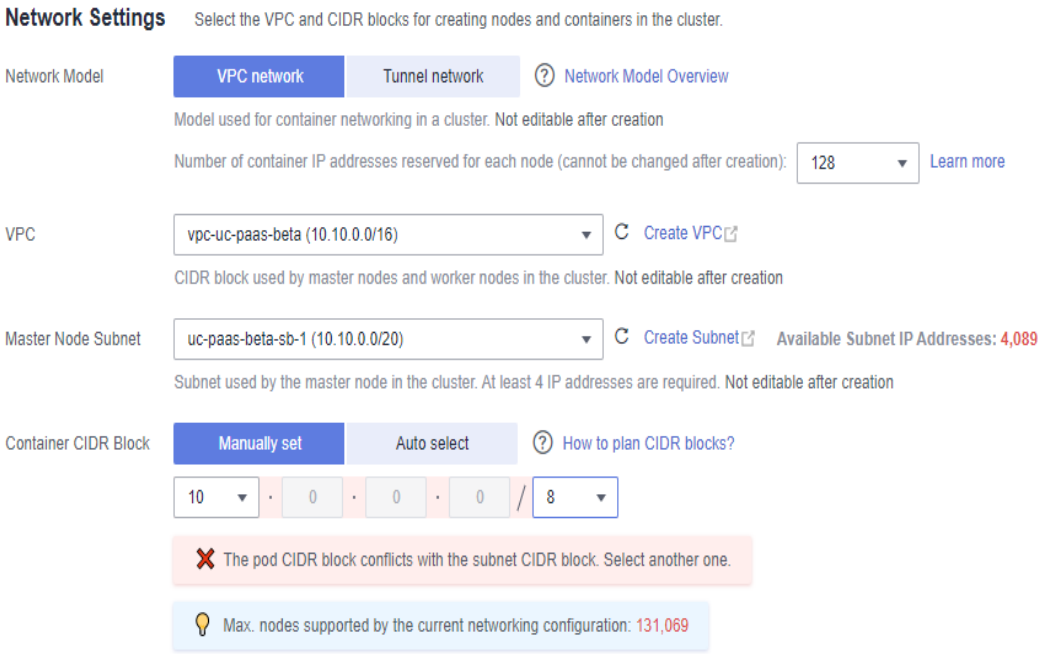
Check Item 2: Whether a Network ACL Has Been Configured for the Node

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.

7.1.10 How Do I Resolve a Conflict Between the VPC CIDR Block and the Container CIDR Block?

When you create a cluster, if the container CIDR block conflicts with the VPC CIDR block, an error message will be displayed. In this case, change the container CIDR block.

Figure 7-6 Conflict error message



7.1.11 What Should I Do If the Java Error "Connection reset by peer" Is Reported During Layer-4 ELB Health Check

Complete Error Information

```
java.io.IOException: Connection reset by peer
at sun.nio.ch.FileDispatcherImpl.read0(Native Method)
at sun.nio.ch.SocketDispatcher.read(SocketDispatcher.java:39)
at sun.nio.ch.IOUtil.readIntoNativeBuffer(IOUtil.java:223)
at sun.nio.ch.IOUtil.read(IOUtil.java:197)
at sun.nio.ch.SocketChannelImpl.read(SocketChannelImpl.java:380)
at com.wanyu.smarthome.gateway.EquipmentSocketServer.handleReadEx(EquipmentSocketServer.java:245)
at com.wanyu.smarthome.gateway.EquipmentSocketServer.run(EquipmentSocketServer.java:115)
```

Analysis Results

A socket server is established using Java Non-blocking I/O (NIO). When the client is shut down unexpectedly rather than sending a specified notification to instruct the server to exit, an error is reported.

TCP Health Check Process

1. The ELB node that performs health checks sends an SYN packet to the backend server (private IP address+health check port) based on the health check configuration.
2. After receiving the packet, the backend server returns an SYN-ACK packet over its port.
3. If the ELB node does not receive the SYN-ACK packet within the timeout duration, the backend server is declared unhealthy. Then, the ELB node sends an RST packet to the backend server to terminate the TCP connection.

4. If the ELB node receives the SYN-ACK packet from the backend server within the timeout duration, it sends an ACK packet to the backend server and declares that the backend server is healthy. Then, the ELB node sends an RST packet to the backend server to terminate the TCP connection.

Note

After a normal TCP three-way handshake, there will be data transfer. However, an RST packet will be sent to terminate the TCP connection during the health check. The applications on the backend server may determine a connection error and reports a message, for example, "Connection reset by peer".

This error is justified and unavoidable. You can ignore it.

7.1.12 How Do I Locate the Service Event Indicating That No Node Is Available for Binding?

Step 1 Log in to the CCE console and click the cluster name to access the cluster console. In the navigation pane, choose **Services & Ingresses**.

Step 2 Check whether the Service has an associated workload, or whether the pods of the associated workload are normal.

----End

7.1.13 Why Does "Dead loop on virtual device gw_11cbf51a, fix it urgently" Intermittently Occur When I Log In to a VM using VNC?

Symptom

In a cluster that uses the VPC network model, the message "Dead loop on virtual device gw_11cbf51a, fix it urgently" is displayed after login to the VM.

```
[7520230.908741] Dead loop on virtual device gw_11cbf51a, fix it urgently!
[7764908.323899] Dead loop on virtual device gw_11cbf51a, fix it urgently!
[7876345.412678] Dead loop on virtual device gw_11cbf51a, fix it urgently!
[7886952.430199] Dead loop on virtual device gw_11cbf51a, fix it urgently!
[8053806.787694] Dead loop on virtual device gw_11cbf51a, fix it urgently!
```

Cause

The VPC network model uses the open source Linux IPvlan module for container networking. In IPvlan L2E mode, layer-2 forwarding is preferentially performed, and then layer-3 forwarding.

Scene reproduction

Assume that there is a service pod A, which provides services externally and is constantly accessed by the node you log in to via the container gateway port through the host Kubernetes Service. Another scenario can be that pods on this node directly access each other. When pod A exits due to upgrade, scale-in, or other reasons, and the corresponding network resources are reclaimed, if the node still attempts to send packets to the IP address of pod A, the IPvlan module in the

kernel first attempts to forward these packets at Layer 2 based on the destination IP address. However, as the NIC to which the pod A IP address belongs can no longer be found, the IPvlan module determines that the packet may be an external packet. Therefore, the module attempts to forward the packet at Layer 3 and matches the gateway port based on the routing rule. After the gateway port receives the packet again, it forwards the packet through the IPvlan module, and this process repeats. The **dev_queue_xmit** function in the kernel detects that the packet is repeatedly sent for 10 times. As a result, the packet is discarded and this log was generated.

After a packet is lost, the access initiator generally performs backoff retries for several times. Therefore, several logs are printed until the ARP in the container of the access initiator ages or the service terminates the access.

For communication between containers on different nodes, the destination and source IP addresses do not belong to the same node-level dedicated subnet (note that this subnet is different from the VPC subnet). Therefore, packets will not be repeatedly sent, and this problem will not occur.

Pods on different nodes in the same cluster can be accessed through a NodePort Service. However, the IP address of the NodePort Service is translated into the IP address of the gateway interface of the accessed container by SNAT, which may generate the logs you see above.

Impact

The normal running of the accessed container is not affected. When a container is destroyed, there is a slight impact that packets are repeatedly sent for 10 times and then discarded. This process is fast in the kernel and has little impact on the performance.

If the ARP ages, the service does not retry, or a new container is started, the container service packets are redirected to the new service through kube-proxy.

Handling in the Open Source Community

Currently, this problem still exists in the community when the IPvlan L2E mode is used. The problem has been reported to the community for a better solution.

Solution

The dead loop problem does not need to be resolved.

However, it is recommended that the service pod gracefully exit. Before the service is terminated, set your pod to the deleting state. After the service processing is complete, the pod exits.

7.1.14 Why Does a Panic Occasionally Occur When I Use Network Policies on a Cluster Node?

Scenario

Cluster version: v1.15.6-r1

Cluster type: CCE cluster

Network model: Container tunnel network

Node operating system: CentOS 7.6

After a network policy is configured for the cluster, the canal-agent network component on the node is incompatible with the CentOS 7.6 kernel. As a result, a kernel panic may occur.

Conditions

If any of the following conditions is not met, this issue will not occur:

- The cluster version is v1.15.6-r1 and the container tunnel network model is used.
- The CentOS 7.6 node uses the canal-agent component whose version is 1.0.RC10.1230.B005 or earlier. (CentOS 7.6 nodes created on or before February 23, 2021 use such component.)
- You plan to use or have used network policies.

Fault Locating

Quick locating (for pay-per-use nodes)

Check whether your CentOS 7.6 node was created after February 24, 2021 on the CCE console.

Accurate locating (General)

If the cluster version is v1.15.6-r1, the network model is container tunnel network, the node OS is CentOS 7.6, and the canal-agent component version is 1.0.RC10.1230.B005.sp1 or later, the problem will not occur. If an earlier version is used (for example, 1.0.RC10.1230.B005, 1.0.RC10.1230.B003, or 1.0.RC10.1230.B002), you are advised to reset or delete the node before configuring network policies.

Perform the following operations to get the version of the network component on the node:

Step 1 Prepare a node where kubectl can be used.

Step 2 Run the following command to query the CentOS node list:

```
for node_item in $(kubectl get nodes --no-headers | awk '{print $1}') ; do kubectl get node ${node_item} -o yaml | grep CentOS >/dev/null; if [[ "$?" == "0" ]];then echo "${node_item} is CentOS node";fi;done
```

The command output is as follows:

```
10.0.50.187 is CentOS node
10.0.50.220 is CentOS node
10.0.50.43 is CentOS node
```

Step 3 Assume that the IP address of the target CentOS node is 10.0.50.187. Run the following command to check the canal-agent version:

```
kubectl get packageversions.version.cce.io 10.0.50.187 -o yaml | grep -A 1 canal-agent
```

The command output is as follows:


```
  - name: canal-agent
    version: 1.0.RC10.1230.B005.sp1
```

----End

Solution

If you still want to use the node, reset the CentOS 7.6 nodes in the cluster to upgrade the networking components to the latest version. For details, see **Resetting a Node**.

If you want to delete the risky node and purchase a new one, see **Deleting a Node** and **Buying a Node**.

7.1.15 Why Are Lots of source ip_type Logs Generated on the VNC?

Scenario

Cluster version: v1.15.6-r1

Cluster type: CCE cluster

Network model: VPC network

Node operating system: CentOS 7.6

When containers on the preceding nodes communicate with each other, the container networking component reports a large number of source ip_types or "not ipvlan but in own host logs" on the VNC. As a result, the VNC page on the node and the container networking performance in high-load scenarios are affected. Symptoms of this problem are as follows:

```
[ 3840.916433] ==========source ip_type 2, ipv4 10.0.0.128, mac fa:16:3e:57:f2:8f
[ 3840.916527] ==========source ip_type 2, ipv4 10.0.0.129, mac fa:16:3e:57:f2:8f
[ 3840.916736] ==========source ip_type 2, ipv4 10.0.0.129, mac fa:16:3e:57:f2:8f
```

```
[16739.000551] =====not ipvlan but in own host, mac_src=fa:16:3e:34:23:93 mac_dst=ff:ff:ff:ff:ff:ff
[16740.000968] =====not ipvlan but in own host, mac_src=fa:16:3e:34:23:93 mac_dst=ff:ff:ff:ff:ff:ff
```

Fault Locating

1. Quick Check

This method applies to pay-per-use nodes. Check the node creation time on the CCE console. CentOS 7.6 nodes created on or after February 24, 2021 do not have this problem.

2. Accurate Check (General)

You can perform the following operations to check whether a node is affected:

Step 1 Log in to each CCE node as user **root**.

Step 2 Run the following command to check whether the node is risky:

```
ETH0_IP=$(ip addr show eth0 | grep "inet " | head -n 1 | awk '{print $2}' | awk -F '/' '{print $1}') ;arping -w 0.2 -c 1 -I gw_11cbf51a 1.1.1.1 >/dev/null 2>&1 ; echo ;dmesg -T | grep -E "==not ipvlan but in own host|
```



```
==source ip_type" 1>/dev/null 2>&1 ; if [[ "$?" == "0" ]];then echo "WARNING, node ${ETH0_IP} is affected"; else echo "node ${ETH0_IP} works well"; fi;
```

NOTE

In this command, *1.1.1.1* is an example IP address, which is used only to trigger ARP packet sending. You can use it or replace it with a valid IP address.

Step 3 If the following information is displayed, the node has potential risks. *10.2.0.35* is the IP address of the eth0 NIC on the node. The actual IP address will be displayed in your practice.

```
WARNING, node 10.2.0.35 is affected
```

If the following information is displayed, the node does not have this problem:

```
node 10.2.0.35 works well
```

----End

Solution

If you still want to use the node, reset the CentOS 7.6 nodes in the cluster to upgrade the networking components to the latest version. For details, see **Resetting a Node**.

If you want to delete the risky node and purchase a new one, see **Deleting a Node** and **Buying a Node**.

7.1.16 What Should I Do If Status Code 308 Is Displayed When the Nginx Ingress Controller Is Accessed Using the Internet Explorer?

Symptom

After the Nginx Ingress Controller is upgraded, existing services cannot be accessed by Using the Internet Explorer, and the status code is 308.

Possible Cause

After the Nginx Ingress Controller is upgraded, the default permanent redirection status code changes from 301 to 308. However, Internet Explorer of some earlier versions does not support this code. As a result, the Nginx Ingress Controller cannot be accessed.

Nginx Ingress Controller community issue: **https://github.com/kubernetes/ingress-nginx/issues/1825**

Solution

When creating an Ingress, you can use the **nginx.ingress.kubernetes.io/permanent-redirect-code** annotation to specify that the permanent redirection status code is 301.

An example is as follows:

```
apiVersion: networking.k8s.io/v1
kind: Ingress
```



```
metadata:
  name: ingress-test
  namespace: default
  annotations:
    nginx.ingress.kubernetes.io/permanent-redirect-code: '301'
...
```

7.1.17 What Should I Do If Nginx Ingress Access in the Cluster Is Abnormal After the NGINX Ingress Controller 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 Controller 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:' -B 1
```

- 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@ + + + +  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@ + + + +  paas]# kubectl logs -nkube-system   cceaddon-nginx-ingress-controller-545db6b4f7-bv74t | grep 'updating Ing
  ress 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.

----End

Solution

Add an annotation to the Nginx ingress 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 Cause

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 Ingress.
- 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: {}
```


7.1.18 What Should I Do If An Error Occurred During a LoadBalancer Update?

Symptom

An error occurs when a LoadBalancer is updated. The error information is as follows:

```
(combined from similar events):Details:Update member of listener/pool(dc9098a3-e004-4e60-ac6c-44a9a04bd8f8/539490e1-51c2-4c09-b4df-10730f77e35f) error: Failed to create member : (error_msg":"Quota exceeded for resources: members_per_pool","error_code":"ELB.8905","request_id":"e064fd46211318ff57f455b29c07c821"},status code: 409
```

Check Item 1: Check Whether the Number of Backend Servers Reaches the Upper Limit

By default, a maximum of 500 backend servers can be added to a backend server group of a load balancer. When a dedicated load balancer is used for creating a Service in a CCE Turbo cluster, a backend server is created on the ELB console for each Service pod. If the number of backend servers exhausts the upper limit, the preceding error occurs.

Solution: Properly plan the backend servers of the load balancer based on service requirements. For details, see the following documents:

Check Item 2: Check Whether the Backend Server Health Check Is Abnormal

To ensure uninterrupted service, a new backend server is added first during the load balancer backend server update. The original backend server will be deleted only after the new backend server is available.

However, if the backend server quota is used up, no more backend servers can be added. As a result, the preceding error occurs, and the existing backend servers will be updated directly. If the health checks of all updated backend servers failed due to incorrect configuration during the Service update, the original normal backend servers will not be deleted to ensure normal services. In this case, the incorrect configurations apply only to some backend servers, while the other backend servers still keep the original configurations.

Solution: If the backend server quota is used up, configure the correct health check protocol and port when updating the Service and then check whether the health check is performed successfully.

7.1.19 How Do I Handle "Invalid Input for Rules" That Occurs with a LoadBalancer Ingress?

Symptom

An alarm is generated when a LoadBalancer ingress is created or updated and information similar to the following is displayed:

```
Update elb(*****) listener(*****)error: status_code: 400, resp_body:{"error_msg":"Invalid input for rules. Reason: the number of condition for per policy must be no larger than 10.","error_code":"ELB.8902"."request_id": *****"}
```


Solution

The error code **ELB.8902** in this alarm indicates that there is an issue with the request parameter. For details, see **Error Codes** You will need to adjust the parameter settings based on the cause of the error.

Symptom	Possible Cause	Solution
The alarm contains the following information: Reason: the number of condition for per policy must be no larger than 10.	Because of the limitations imposed by the ELB API, there are certain restrictions when using advanced forwarding rules. In each forwarding policy, including the domain name, path, HTTP request method, HTTP request header, query string, CIDR block, and cookie forwarding rules, a maximum of 10 conditions can be configured for all types of forwarding rules. Each parameter value in a rule is considered as one condition. For example, if you have one domain name forwarding rule and one path forwarding rule in a forwarding policy, you can add up to eight character strings if you include the query string forwarding rules. For details about the advanced forwarding rules, see **Configuring Advanced Forwarding Actions for a LoadBalancer Ingress**.	Modify the ingress configuration and make sure that the number of rules in the forwarding policy does not exceed the maximum limit.

7.1.20 What Could Cause Access Exceptions After Configuring an HTTPS Certificate for a LoadBalancer Ingress?

If you configure an HTTPS certificate for a LoadBalancer ingress, access may become abnormal if any of the following issues arise. To fix the problem, refer to the causes listed in the table.

Cause	Symptom	Solution
The certificate has expired.	The error similar to the following is displayed when the **curl** command is executed: `SSL certificate problem: certificate has expired`	Replace the certificate in a timely manner.
An unmatched HTTPS certificate chain is used by a client to verify the HTTPS certificate configured for the LoadBalancer ingress.	The error similar to the following is displayed when the **curl** command is executed: `SSL certificate problem: unable to get local issuer certificate`	Ensure that the HTTPS certificate chain on the client matches the certificate configured for the LoadBalancer ingress.
No domain name is specified when a certificate is created.	The error similar to the following is displayed when the **curl** command is executed: `SSL: unable to obtain common name from peer certificate`	Specify a domain name when creating a certificate.
The domain name to be accessed is different from the domain name of the HTTPS certificate.	The error similar to the following is displayed when the **curl** command is executed: `SSL: certificate subject name 'example.com' does not match target host name 'test.com'`	Configure a certificate that matches the domain name for the ingress.

NOTE

You can run the following command to check the certificate information, such as expiration time and domain name. **ca.crt** specifies the certificate path.

```
openssl x509 -in ca.crt -subject -noout -text
```

Updating a Certificate

- To update a TLS certificate, modify the secret where the certificate is imported to on CCE. The TLS certificate is imported to a secret first. CCE then automatically handles the certificate configurations on the ELB console and gives a name to the certificate (started with **k8s_plb_default**). This certificate, which is generated by CCE, **cannot be modified or deleted from the ELB console**.
- To update a certificate created on the ELB console, modify the certificate on the ELB console. There is no need to manually set up the cluster secret.

7.2 Network Planning

7.2.1 What Is the Relationship Between Clusters, VPCs, and Subnets?

A 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 ECSs, load balancers, 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.

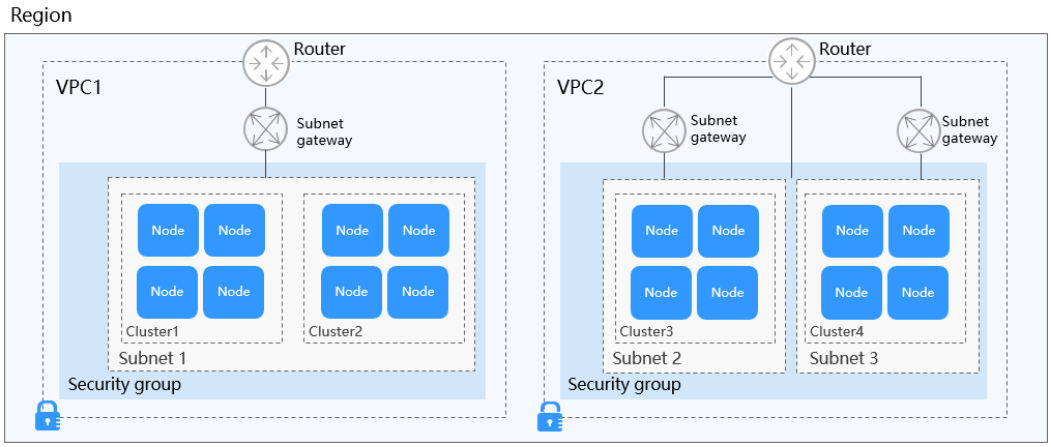
A VPC can be divided into multiple subnets. Security groups are configured to determine whether these subnets can communicate with each other. This ensures that subnets can be isolated from each other, so that you can deploy different services on different subnets.

A cluster is one or a group of ECSs or BMSs (also known as nodes) in the same VPC. It provides computing resource pools for running containers.

As shown in **Figure 7-7**, a region may consist 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 7-7 Relationship between clusters, VPCs, and subnets



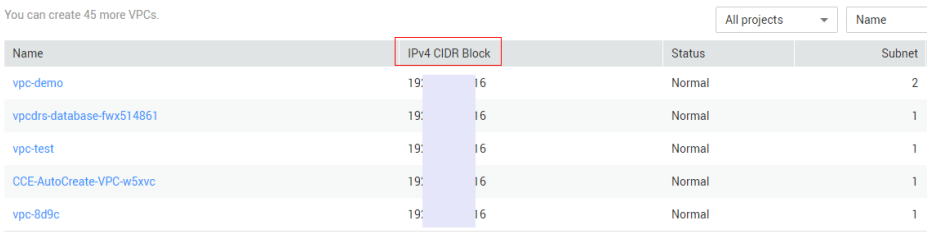
References

Planning CIDR Blocks for a Cluster

7.2.2 How Do I View the VPC CIDR Block?

On the home page of the VPC console, view the **Name/ID** and **CIDR Block** of VPCs. You can modify the CIDR block of a VPC or re-create a VPC.

Figure 7-8 Viewing the CIDR block of VPCs



7.2.3 How Do I Set the VPC CIDR Block and Subnet CIDR Block for a CCE Cluster?

The CIDR block of a VPC cannot be changed after the VPC is created. When creating a VPC, allocate sufficient IP addresses for the VPC and subnets.

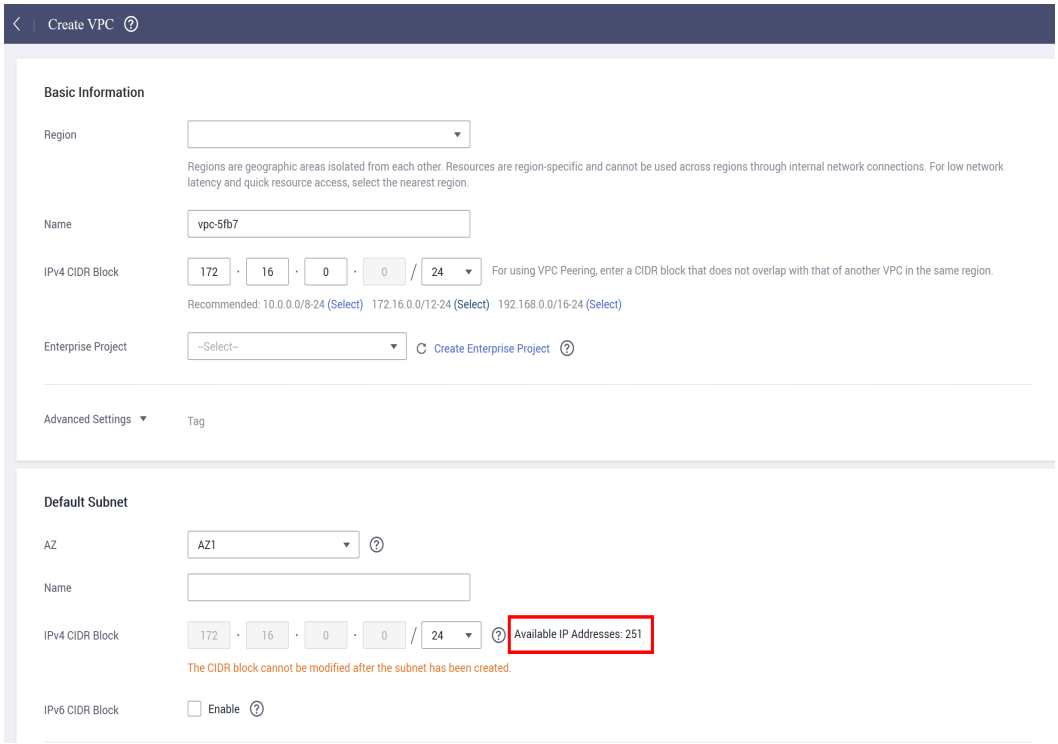
The subnet CIDR block can be set on the VPC console by clicking **Create VPC**. You can view the number of available IP addresses under the CIDR block setting.

If the subnet mask is not set properly, the number of available nodes in the cluster may be insufficient.

Example:

- If the cluster has 1000 nodes, you can set the subnet CIDR block to 192.168.0.0/20, which supports 4090 nodes.
- If VPC CIDR block is set to 192.168.0.0/16 and the subnet CIDR block is set to 192.168.0.0/25, only 122 nodes are supported. If you create a cluster with 200 nodes using this VPC, only 122 nodes (including master nodes) can be added.

Figure 7-9 Viewing the number of available IP addresses



References

Planning CIDR Blocks for a Cluster

7.2.4 How Do I Set a Container CIDR Block for a CCE Cluster?

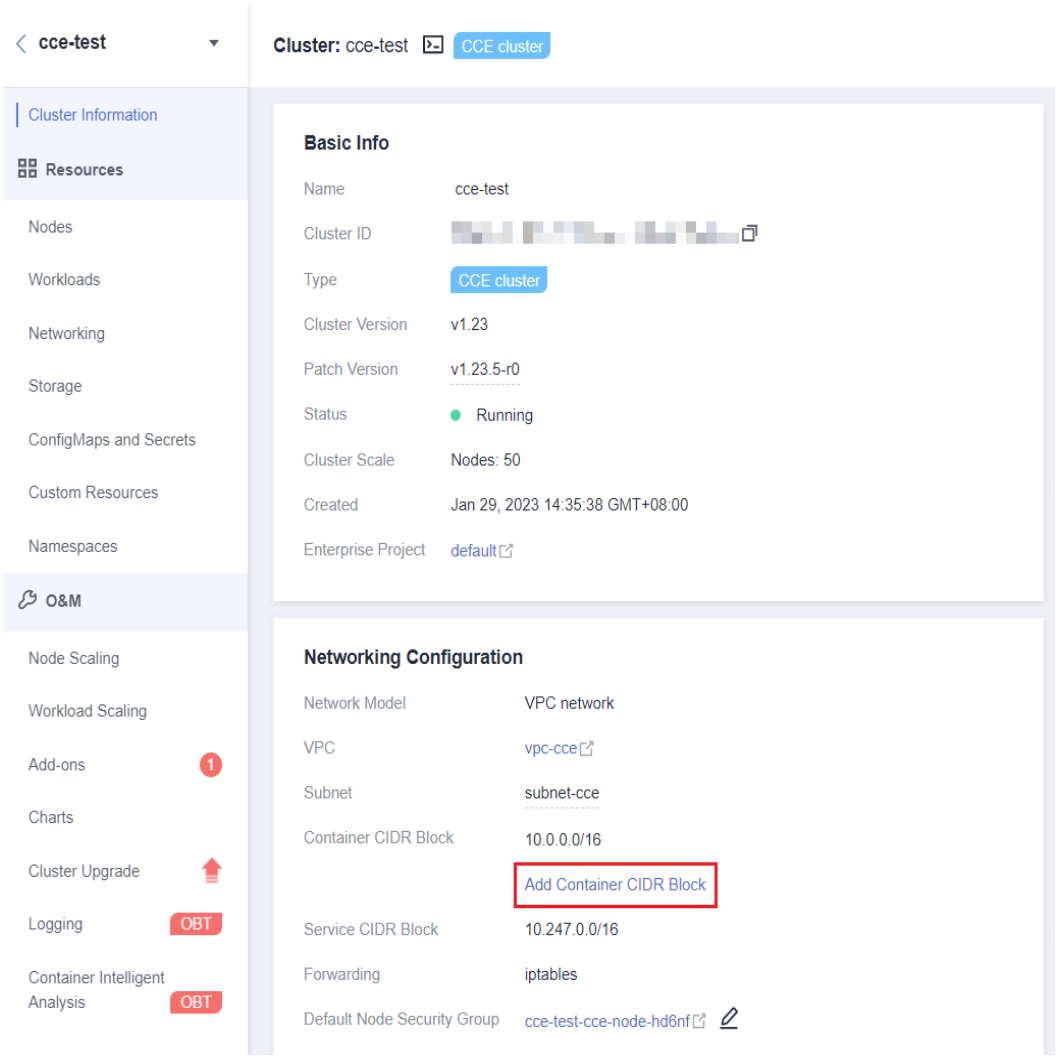
Log in to the CCE console and set **Container CIDR Block** when creating a cluster.

Available container CIDR blocks are 10.0.0.0/8-18, 172.16.0.0/16-18, and 192.168.0.0/16-18.

To add a container CIDR block after a cluster is created, go to the cluster information page and click **Add Container CIDR Block**.

NOTICE

- Currently, container CIDR blocks cannot be added to clusters that use the container tunnel network.
- The added container CIDR block cannot be deleted.
- The default service CIDR block is 10.247.0.0/16. Therefore, the container CIDR block cannot be 10.247.0.0/16.

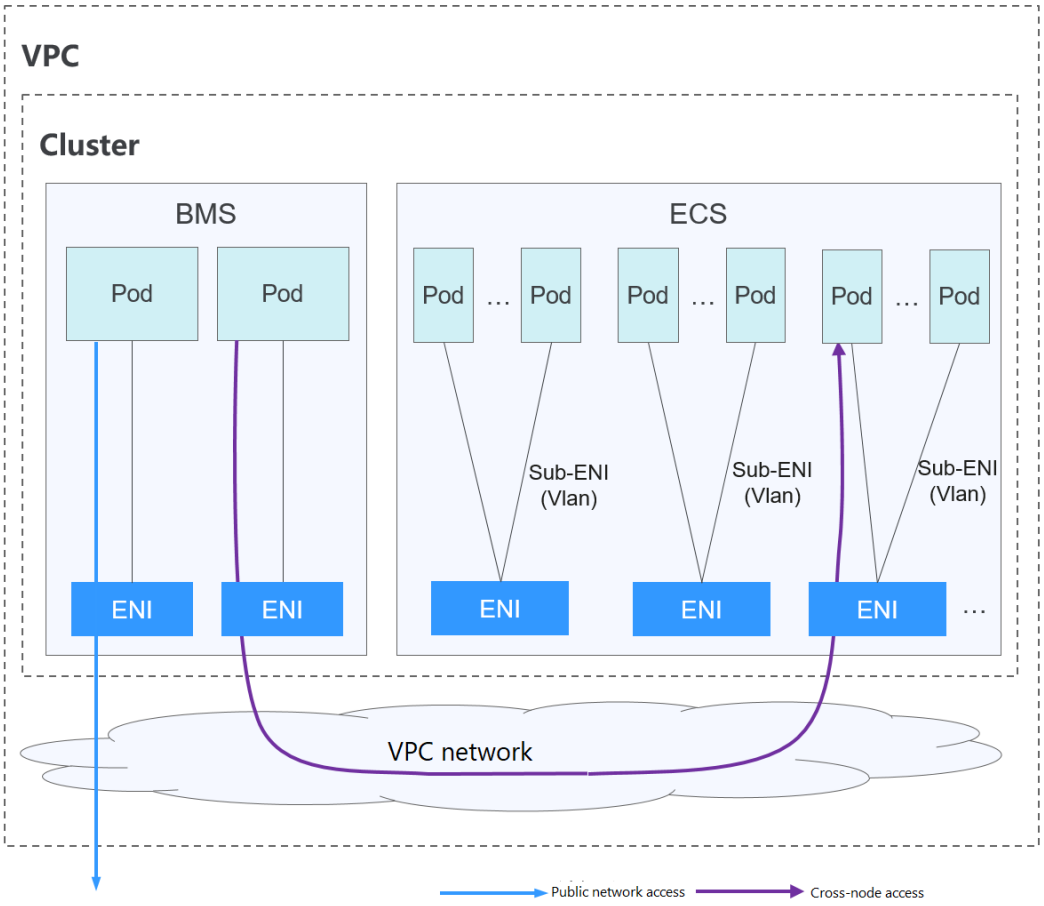


7.2.5 When Should I Use Cloud Native Network 2.0?

Cloud Native Network 2.0

Cloud Native Network 2.0 is a new container networking solution. This network model deeply integrates the native elastic network interfaces (ENIs) of VPC, uses the VPC CIDR block to allocate container addresses, and supports passthrough networking to containers through a load balancer.

Figure 7-10 Cloud Native Network 2.0



Notes and Constraints

This network model is available only to CCE Turbo clusters.

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 is applicable to scenarios that have high requirements on bandwidth and latency, such as online live broadcast and e-commerce seckill.
- Large-scale networking: Cloud Native Network 2.0 supports a maximum of 2000 ECS nodes and 100,000 containers.

7.2.6 What Is an Elastic Network Interface?

An elastic network interface is a virtual network card. You can create and configure elastic network interfaces and attach them to your cloud servers (ECSs or BMSs) to build flexible and highly available networks.

Elastic Network Interface Types

- A primary network interface is created together with an ECS instance by default, which cannot be detached from its ECS.
- An extension network interface can be created and attached to an ECS, and can be detached from the ECS. The number of extension network interfaces that you can attach to an ECS varies by ECS flavor.

Application Scenario

- Flexible migration

  You can detach an elastic network interface from a cloud server and then attach it to another server. The elastic network interface retains its private IP address, EIP, and security group rules. In this way, service traffic on the faulty server can be quickly migrated to the standby server, implementing quick service recovery.

- Independent traffic management

  You can attach multiple elastic network interfaces that belong to different subnets in a VPC to the same instance, and specify them to carry the private network traffic, public network traffic, and management network traffic of the instance, respectively. You can configure access control policies and routing policies for each subnet, and configure security group rules for each elastic network interface to isolate networks and service traffic.

Restrictions

- An instance and its extension network interfaces must be in the same AZ, VPC, and subnet. However, they can belong to different security groups.
- A primary network interface cannot be detached from its ECS.
- The number of extension network interfaces that you can attach to an ECS varies by ECS flavor.

7.2.7 How Can I Configure a Security Group Rule for a Cluster?

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 nodes and worker nodes, separately. The name of the master node security group is in the format of *{Cluster name}*-**cce-control-***{Random ID}*, and that of the worker node security group is in the format of *{Cluster name}*-**cce-node-***{Random ID}*. For a CCE Turbo cluster, an additional elastic network interface security group, with the name following the format of *{Cluster name}*-**cce-eni-***{Random ID}*, will be created.

To modify the security group rules, log in to the management console and choose **Service List** > **Networking** > **Virtual Private Cloud**. On the page displayed, choose **Access Control** > **Security Groups** in the navigation pane, locate the rpw containing the target security group, and modify the rules.

If you need to specify a node security group when creating a cluster, allow specific ports based on the rules of the default security group automatically created in the cluster to ensure normal communication in your cluster.

You can check the default security group rules of clusters using different network models in:

- **Security Group Rules in a Cluster That Uses the VPC Network Model**
- **Security Group Rules in a Cluster That Uses the Tunnel Network Model**
- **Security Group Rules in a CCE Turbo Cluster That Uses the Cloud Native 2.0 Network Model**

NOTICE

- Be careful when **modifying or removing** security group rules as it could **impact the cluster's operation**. Avoid modifying the rules for the ports essential to CCE.
- When adding a security group rule, ensure that **this rule does not conflict with the existing rules**. If there is a conflict, existing rules may become invalid, affecting cluster running.

Security Group Rules in a Cluster That Uses the VPC Network Model

Worker node security group

A security group, with a name following the format of *{Cluster name}*-**cce-node-***{Random ID}*, is automatically created for the worker nodes in a cluster. For details about the default ports, see **Table 7-3**.

Table 7-3 Default ports in the security group of the worker nodes in a cluster that uses the VPC network model

Direction	Port	Default Source Address	Description	Modification Suggestion	Impact After Modification
Inbound rules	All UDP ports	VPC CIDR block	Allow access between the worker nodes and between the worker nodes and the master nodes.	Modification not recommended	Modifying the configuration can interrupt the cluster functionality.
	All TCP ports				

Direction	Port	Default Source Address	Description	Modification Suggestion	Impact After Modification
	All ICMP ports	Master node security group	Allow the master nodes to access the worker nodes.	Modification not recommended	Modifying the configuration can interrupt the cluster functionality.
	TCP port range: 30000 to 32767	All IP addresses (0.0.0.0/0)	Allow access from the NodePort Services.	Modification made if necessary	The ports must allow traffic from the CIDR blocks of the VPC, container, and load balancer.
	UDP port range: 30000 to 32767				
	All	Container CIDR block	Allow containers within the cluster to access the nodes.	Modification not recommended	Modifying the configuration can interrupt the cluster functionality.
	All	Worker node security group	Restrict access from outside the worker node security group, but the access between pods in the worker node security group.	Modification not recommended	Modifying the configuration can interrupt the cluster functionality.
	TCP port 22	All IP addresses (0.0.0.0/0)	Allow SSH access to Linux ECSs.	**Modification recommended**	You are advised to allow access only from fixed IP addresses or IP address ranges.

Direction	Port	Default Source Address	Description	Modification Suggestion	Impact After Modification
Outbound rule	All	All IP addresses (0.0.0.0/0)	Allow traffic on all ports by default. You are advised to retain this setting.	Modification made if necessary	If you want to harden security by allowing traffic only on specific ports, remember to allow such ports. For details, see **Hardening Outbound Rules**.

Master node security group

A security group, with a name following the format of *{Cluster name}*-**cce-control-***{Random ID}*, is automatically created for the master nodes in a cluster. For details about the default ports, see **Table 7-4**.

Table 7-4 Default ports in the security group of the master nodes in a cluster that uses the VPC network model

Direction	Port	Default Source Address	Description	Modification Suggestion	Impact After Modification
Inbound rules	TCP port 5444	VPC CIDR block	Allow access from kube-apiserver, which provides lifecycle management for Kubernetes resources.	Modification not recommended	Modifying the configuration can interrupt the cluster functionality.
	TCP port 5444	Container CIDR block			
	TCP port 9443	VPC CIDR block	Allow the network add-on of the worker nodes to access the master nodes.	Modification not recommended	Modifying the configuration can interrupt the cluster functionality.

Direction	Port	Default Source Address	Description	Modification Suggestion	Impact After Modification
	TCP port 5443	All IP addresses (0.0.0.0/0)	Allow kube-apiserver of the master nodes to listen to the worker nodes.	**Modification recommended**	The port must allow traffic from the CIDR blocks of the VPC, the control plane of the hosted service mesh, and container. **NOTE** To use CloudShell, you need to allow traffic from 198.19.0.0/16 on port 5443. Otherwise, you cannot access the cluster using CloudShell.
	TCP port 8445	VPC CIDR block	Allow the storage add-on of the worker nodes to access the master nodes.	Modification not recommended	Modifying the configuration can interrupt the cluster functionality.
	All	Master node security group	Restrict access from outside the master node security group, but the access between pods in the master node security group.	Modification not recommended	Modifying the configuration can interrupt the cluster functionality.
Outbound rule	All	All IP addresses (0.0.0.0/0)	Allow traffic on all ports by default.	Modification not recommended	Modifying the configuration can interrupt the cluster functionality.

Security Group Rules in a Cluster That Uses the Tunnel Network Model

Worker node security group

A security group, with a name following the format of *{Cluster name}*-**cce-node-***{Random ID}*, is automatically created for the worker nodes in a cluster. For details about the default ports, see **Table 7-5**.

Table 7-5 Default ports in the security group of the worker nodes in a cluster that uses the tunnel network model

Direction	Port	Default Source Address	Description	Modification Suggestion	Impact After Modification
Inbound rules	UDP port 4789	All IP addresses (0.0.0.0/0)	Allow access between containers.	Modification not recommended	Modifying the configuration can interrupt the cluster functionality.
	TCP port 10250	Master node CIDR block	Allow the master nodes to access kubelet on the worker nodes to run commands, for example, **kubectl exec** *{pod}*.	Modification not recommended	Modifying the configuration can interrupt the cluster functionality.
	TCP port range: 30000 to 32767	All IP addresses (0.0.0.0/0)	Allow access from the NodePort Services.	Modification made if necessary	The ports must allow traffic from the CIDR blocks of the VPC, load balancer, and container.
	UDP port range: 30000 to 32767				
	TCP port 22	All IP addresses (0.0.0.0/0)	Allow SSH access to Linux ECSs.	**Modification recommended**	You are advised to allow access only from fixed IP addresses or IP address ranges.

Direction	Port	Default Source Address	Description	Modification Suggestion	Impact After Modification
	All	Worker node security group	Restrict access from outside the worker node security group, but the access between pods in the worker node security group.	Modification not recommended	Modifying the configuration can interrupt the cluster functionality.
Outbound rule	All	All IP addresses (0.0.0.0/0)	Allow traffic on all ports by default. You are advised to retain this setting.	Modification made if necessary	If you want to harden security by allowing traffic only on specific ports, remember to allow such ports. For details, see **Hardening Outbound Rules**.

Master node security group

A security group, with a name following the format of *{Cluster name}*-**cce-control-***{Random ID}*, is automatically created for the master nodes in a cluster. For details about the default ports, see **Table 7-6**.

Table 7-6 Default ports in the security group of the master nodes in a cluster that uses the tunnel network model

Direction	Port	Default Source Address	Description	Modification Suggestion	Impact After Modification
Inbound rules	UDP port 4789	All IP addresses (0.0.0.0/0)	Allow access between containers.	Modification not recommended	Modifying the configuration can interrupt the cluster functionality.

Direction	Port	Default Source Address	Description	Modification Suggestion	Impact After Modification
	TCP port 5444	VPC CIDR block	Allow access from kube-apiserver, which provides lifecycle management for Kubernetes resources.	Modification not recommended	Modifying the configuration can interrupt the cluster functionality.
	TCP port 5444	Container CIDR block			
	TCP port 9443	VPC CIDR block	Allow the network add-on of the worker nodes to access the master nodes.	Modification not recommended	Modifying the configuration can interrupt the cluster functionality.
	TCP port 5443	All IP addresses (0.0.0.0/0)	Allow kube-apiserver of the master nodes to listen to the worker nodes.	**Modification recommended**	The port must allow traffic from the CIDR blocks of the VPC, the control plane of the hosted service mesh, and container. **NOTE** To use CloudShell, you need to allow traffic from 198.19.0.0/16 on port 5443. Otherwise, you cannot access the cluster using CloudShell.
	TCP port 8445	VPC CIDR block	Allow the storage add-on of the worker nodes to access the master nodes.	Modification not recommended	Modifying the configuration can interrupt the cluster functionality.

Direction	Port	Default Source Address	Description	Modification Suggestion	Impact After Modification
	All	Master node security group	Restrict access from outside the master node security group, but the access between pods in the master node security group.	Modification not recommended	Modifying the configuration can interrupt the cluster functionality.
Outbound rule	All	All IP addresses (0.0.0.0/0)	Allow traffic on all ports by default.	Modification not recommended	Modifying the configuration can interrupt the cluster functionality.

Security Group Rules in a CCE Turbo Cluster That Uses the Cloud Native 2.0 Network Model

Worker node security group

A security group, with a name following the format of *{Cluster name}*-**cce-node-***{Random ID}*, is automatically created for the worker nodes in a cluster. For details about the default ports, see **Table 7-7**.

Table 7-7 Default ports in the security group of the worker nodes in a CCE Turbo cluster that uses the Cloud Native 2.0 network model

Direction	Port	Default Source Address	Description	Modification Suggestion	Impact After Modification
Inbound rules	TCP port 10250	Master node CIDR block	Allow the master nodes to access kubelet on the worker nodes to run commands, for example, **kubectl exec** *{pod}*.	Modification not recommended	Modifying the configuration can interrupt the cluster functionality.

Direction	Port	Default Source Address	Description	Modification Suggestion	Impact After Modification
	TCP port range: 30000 to 32767	All IP addresses (0.0.0.0/0)	Allow access from the NodePort Services.	Modification made if necessary	The ports must allow traffic from the CIDR blocks of the VPC, load balancer, and container.
	UDP port range: 30000 to 32767				
	TCP port 22	All IP addresses (0.0.0.0/0)	Allow SSH access to Linux ECSs.	**Modification recommended**	You are advised to allow access only from fixed IP addresses or IP address ranges.
	All	Worker node security group	Restrict access from outside the worker node security group, but the access between pods in the worker node security group.	Modification not recommended	Modifying the configuration can interrupt the cluster functionality.
	All	Container subnet CIDR block	Allow containers within the cluster to access the nodes.	Modification not recommended	Modifying the configuration can interrupt the cluster functionality.
Outbound rule	All	All IP addresses (0.0.0.0/0)	Allow traffic on all ports by default. You are advised to retain this setting.	Modification made if necessary	If you want to harden security by allowing traffic only on specific ports, remember to allow such ports. For details, see **Hardening Outbound Rules**.

Master node security group

A security group, with a name following the format of *{Cluster name}*-**cce-control-***{Random ID}*, is automatically created for the master nodes in a cluster. For details about the default ports, see **Table 7-8**.

Table 7-8 Default ports in the security group of the master nodes in a CCE Turbo cluster that uses the Cloud Native 2.0 network model

Direction	Port	Default Source Address	Description	Modification Suggestion	Impact After Modification
Inbound rules	TCP port 5444	All IP addresses (0.0.0.0/0)	Allow access from kube-apiserver, which provides lifecycle management for Kubernetes resources.	Modification not recommended	Modifying the configuration can interrupt the cluster functionality.
	TCP port 5444	VPC CIDR block		Modification not recommended	Modifying the configuration can interrupt the cluster functionality.
	TCP port 9443	VPC CIDR block	Allow the network add-on of the worker nodes to access the master nodes.	Modification not recommended	Modifying the configuration can interrupt the cluster functionality.
	TCP port 5443	All IP addresses (0.0.0.0/0)	Allow kube-apiserver of the master nodes to listen to the worker nodes.	**Modification recommended**	The port must allow traffic from the CIDR blocks of the VPC, the control plane of the hosted service mesh, and container. **NOTE** To use CloudShell, you need to allow traffic from 198.19.0.0/16 on port 5443. Otherwise, you cannot access the cluster using CloudShell.

Direction	Port	Default Source Address	Description	Modification Suggestion	Impact After Modification
	TCP port 8445	VPC CIDR block	Allow the storage add-on of the worker nodes to access the master nodes.	Modification not recommended	Modifying the configuration can interrupt the cluster functionality.
	All	Master node security group	Restrict access from outside the master node security group, but the access between pods in the master node security group.	Modification not recommended	Modifying the configuration can interrupt the cluster functionality.
	All	Container subnet CIDR block	Allow traffic from all source IP addresses in the container subnet CIDR block.	Modification not recommended	Modifying the configuration can interrupt the cluster functionality.
Outbound rule	All	All IP addresses (0.0.0.0/0)	Allow traffic on all ports by default.	Modification not recommended	Modifying the configuration can interrupt the cluster functionality.

Elastic network interface security group

In a CCE Turbo cluster, an additional security group, with the name following the format of *{Cluster name}*-**cce-eni-***{Random ID}*, is created. By default, containers in the cluster are bound to this security group. For details about the default ports, see **Table 7-9**.

Table 7-9 Default ports of the elastic network interface security group in a CCE Turbo cluster that uses the Cloud Native 2.0 network model

Direction	Port	Default Source Address	Description	Modification Suggestion	Impact After Modification
Inbound rules	All	Elastic network interface security group	Allow containers within the cluster to access each other.	Modification not recommended	Modifying the configuration can interrupt the cluster functionality.
		VPC CIDR block	Allow instances in the cluster VPC to access the containers.	Modification not recommended	Modifying the configuration can interrupt the cluster functionality.
Outbound rule	All	All IP addresses (0.0.0.0/0)	Allow traffic on all ports by default.	Modification not recommended	Modifying the configuration can interrupt the cluster functionality.

Hardening Outbound Rules

By default, all security groups created by CCE allow all the **outbound** traffic. You are advised to retain this configuration. To harden outbound rules, ensure that the traffic on the ports listed in the following table is allowed.

Table 7-10 Minimum configurations of outbound security group rules for the worker nodes

Port	Allowed CIDR	Description
TCP port 53	DNS server of the subnet	Allow traffic on the port for domain name resolution.
UDP port 53		
TCP port 5353	Container CIDR block	Allow traffic on the port for CoreDNS domain name resolution.
UDP port 5353		

Port	Allowed CIDR	Description
UDP port 4789 (required only by clusters that use the tunnel networks)	All IP addresses	Allow access between containers.
TCP port 5443	Master node CIDR block	Allow kube-apiserver of the master nodes to listen to the worker nodes.
TCP port 5444	CIDR blocks of the VPC and containers	Allow access from kube-apiserver, which provides lifecycle management for Kubernetes resources.
TCP port 6443	Master node CIDR block	None
TCP port 8445	VPC CIDR block	Allow the storage add-on of the worker nodes to access the master nodes.
TCP port 9443	VPC CIDR block	Allow the network add-on of the worker nodes to access the master nodes.
All ports	198.19.128.0/17	Allow access to VPC Endpoint (VPCEP).
UDP port 123	100.125.0.0/16	Allow the worker nodes to access the internal NTP server.
TCP port 443	100.125.0.0/16	Allow the worker nodes to access OBS over internal networks to pull the installation package.
TCP port 6443	100.125.0.0/16	Allow the worker nodes to report that the worker nodes have been installed.

7.2.8 How Do I Configure the IPv6 Service CIDR Block When Creating a CCE Turbo Cluster?

Context

To create an IPv4/IPv6 dual-stack CCE Turbo cluster, you need to set an IPv6 Service CIDR block. The default CIDR block is **fc00::/112**, which contains 65,536 IPv6 addresses. If you need to customize a Service CIDR block, you can refer to this section.

IPv6

IPv6 address

An IPv6 address is a 128-bit binary string, four times the length of an IPv4 address. Therefore, the decimal format of IPv4 addresses is no longer applicable. IPv6 addresses are expressed in hexadecimal format. To convert a 128-bit binary string, it is transformed into a 32-bit hexadecimal string. These hexadecimal strings are grouped into sets of four (case insensitive) and separated by a colon (:). IPv6 addresses are divided into eight groups.

An IPv6 address can be omitted in the following ways:

- Omission of leading 0s: 0s can be omitted if the colon group starts with 0s. The following IPv6 addresses are the same.
  - ff01:**0**d28:**0**3ee:**0000**:**0000**:**0000**:**0000**:**0**c23
  - ff01:d28:3ee:**0000**:**0000**:**0000**:**0000**:c23
  - ff01:d28:3ee:**0**:**0**:**0**:**0**:c23
- Omission of all-0s hextets: You can use a double colon (::) to represent a single contiguous string of all-0s segments. A double colon (::) can be used only once.

  Example:

Before Omission	After Omission
ff01:d28:3ee:**0**:**0**:**0**:**0**:c23	ff01:d28:3ee::c23
0:0:0:0:0:0:0:1	::1
0:0:0:0:0:0:0:0	::

IPv6 address segment

An IPv6 address segment is usually expressed in CIDR format. It is usually represented by a slash (/) followed by a number, that is, *IPv6 address/Prefix length*. The function of the prefix is similar to that of the mask of the IPv4 address segment. The number of binary bits occupied by the network part represents the binary bits occupied by the network part. An IPv6 address consists of the network part and host part. The prefix specifies the number of bits occupied by the network part, and the remaining bits are the host part.

For example, **fc00:d28::/32** indicates an IPv6 address segment with a 32-bit prefix. The first 32 bits (**fc00:d28** in binary mode) are the network part and the last 96 bits are available host part.

Constraints on IPv6 Service CIDR Blocks

When setting the cluster service CIDR block, note the following constraints:

- The IPv6 Service CIDR block must belong to the **fc00::/8** CIDR block.

  The address is a unique local address (ULA). The ULA has a fixed prefix **fc00::/7**, including **fc00::/8** and **fd00::/8**. The two ranges are similar to the dedicated IPv4 network addresses **10.0.0.0/8**, **172.16.0.0/12**, and

192.168.0.0/16. They are equivalent to private CIDR blocks and can be used only on the local network.

- The prefix ranges from 112 to 120. You can adjust the number of addresses by adjusting the prefix value. The maximum number of addresses is 65,536.

Example of an IPv6 Service CIDR Block

According to the constraints, this section provides an example of setting an IPv6 CIDR block that contains 8192 IP addresses for your reference.

Step 1 Set the prefix length based on the number of addresses. The prefix length ranges from 112 to 120.

In this example, 8,192 IP addresses are required, which are represented by 13-bit binary strings. An IPv6 address has a total length of 128-bit binary strings. As a result, the prefix length of the IPv6 CIDR block is 115 (128-13). This means that the first 115 bits are used to distinguish the CIDR block, while the last 13 bits indicate 8,192 host IP addresses.

The following table shows how many IP addresses are there in an IPv6 CIDR block with the prefix ranging from 112 to 120.

Prefix Length	Number of IP Addresses
112	65,536
113	32,768
114	16,384
115	8,192
116	4,096
117	2,048
118	1,024
119	512
120	256

Step 2 Set the IPv6 network address, which must belong to the **fc00::/8** CIDR block.

In this example, the prefix length is 115. Because the network address must belong to the **fc00::/8** CIDR block, the first 8-bit binary digits are fixed. The network address that can be modified ranges from the ninth bit to the 115th bit. The 116th bit to the 128th bit are the host part.

If the IPv6 CIDR block is written in binary format, the following conditions must be met:

- It must belong to **fc00::/8**, and the first eight bits in the binary string cannot be modified. Otherwise, the IPv6 CIDR block does not belong to **fc00::/8**. The CIDR block is fixed at **1111 1110**, corresponding to **fc** in hexadecimal format.

- The prefix length must be 115, and it must contain 8,192 IP addresses. The last 13 bits in the binary string are used to indicate the host IP address and are always 0.

The following shows an example and the information in red cannot be modified:

```
Binary: 1111 1100 **** **** ... ***0 0000 0000 0000/115
          |  |  |  |      |  |  |  |
Hexadecimal:  f  c  x  x ...  y  0  0  0/115
```

x is a hexadecimal string. The last digit of the 4-bit binary string corresponding to **y** is always **0**. This means that the hexadecimal string **y** can be **0**, **2**, **4**, **6**, **8**, **a**, **c**, or **e**.

----End

7.2.9 Can Multiple NICs Be Bound to a Node in a CCE Cluster?

Multiple ENIs cannot be bound to a node in a CCE cluster. Do not manually bind multiple ENIs to nodes. Otherwise, network access to clusters will be affected.

7.3 Security Hardening

7.3.1 How Do I Prevent Cluster Nodes from Being Exposed to Public Networks?

Question

How do I prevent cluster nodes from being exposed to public networks?

Solution

- If access to port 22 of a cluster node is not required, you can define a security group rule that disables access to port 22.
- Do not bind an EIP to a cluster node unless necessary.

If remote login to a cluster node is required, you are advised to use Huawei Cloud Bastion Host (CBH) as the transit node to connect to the cluster node.

7.3.2 How Do I Configure an Access Policy for a Cluster?

After the public API Server address is bound to the cluster, modify the security group rules of port 5443 on the master node to harden the access control policy of the cluster.

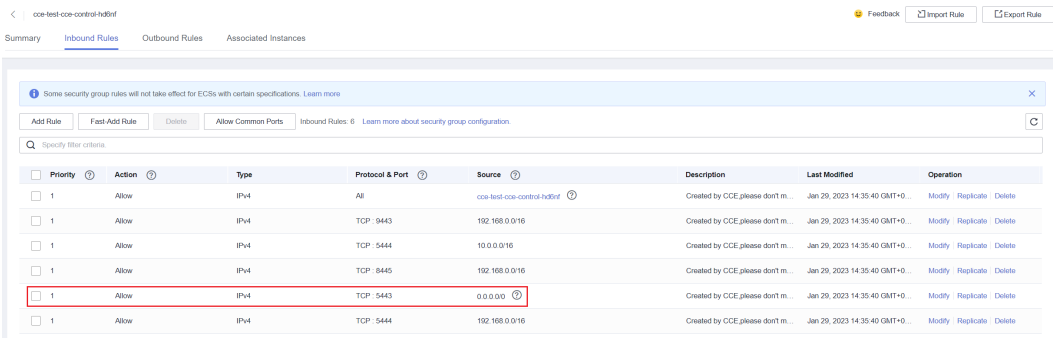
Step 1 Log in to the CCE console and click the cluster name to access the cluster console. On the **Overview** page, find and copy the cluster ID.

Step 2 Log in to the VPC console. In the navigation pane, choose **Access Control** > **Security Groups**.

Step 3 Select **Description** as the filter criterion and search for the target security group by cluster ID.

Step 4 Locate the row that contains the security group (starting with *{CCE cluster name}*-**cce-control**) of the master node and click **Manage Rules** in the **Operation** column.

Step 5 On the page displayed, locate the row that contains port 5443 and click **Modify** in the **Operation** column to modify its inbound rules.

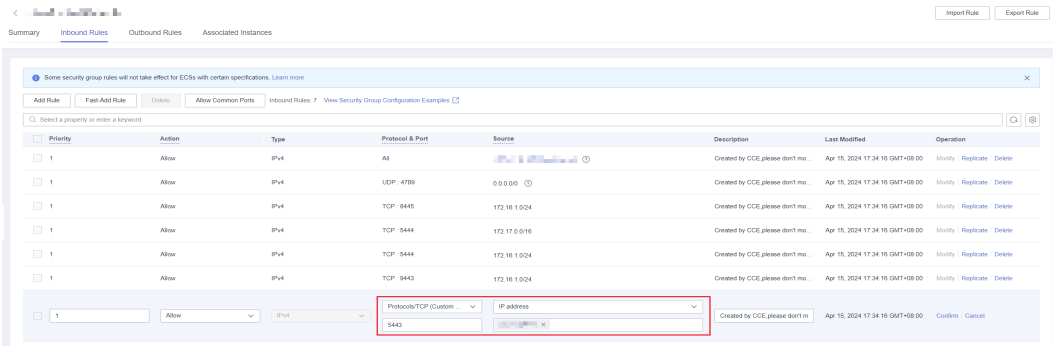


Step 6 Change the source IP address that can be accessed as required. For example, if the IP address used by the client to access the API Server is **100.*.*.***, you can add an inbound rule for port 5443 and set the source IP address to **100.*.*.***.

NOTE

In addition to the client IP address, the port must allow traffic from the CIDR blocks of the VPC, container, and the control plane of the hosted service mesh to ensure that the API Server can be accessed from within the cluster.

To use CloudShell, you need to allow traffic from 198.19.0.0/16 on port 5443. Otherwise, you cannot access the cluster using CloudShell.



Step 7 Click **Confirm**.

----End

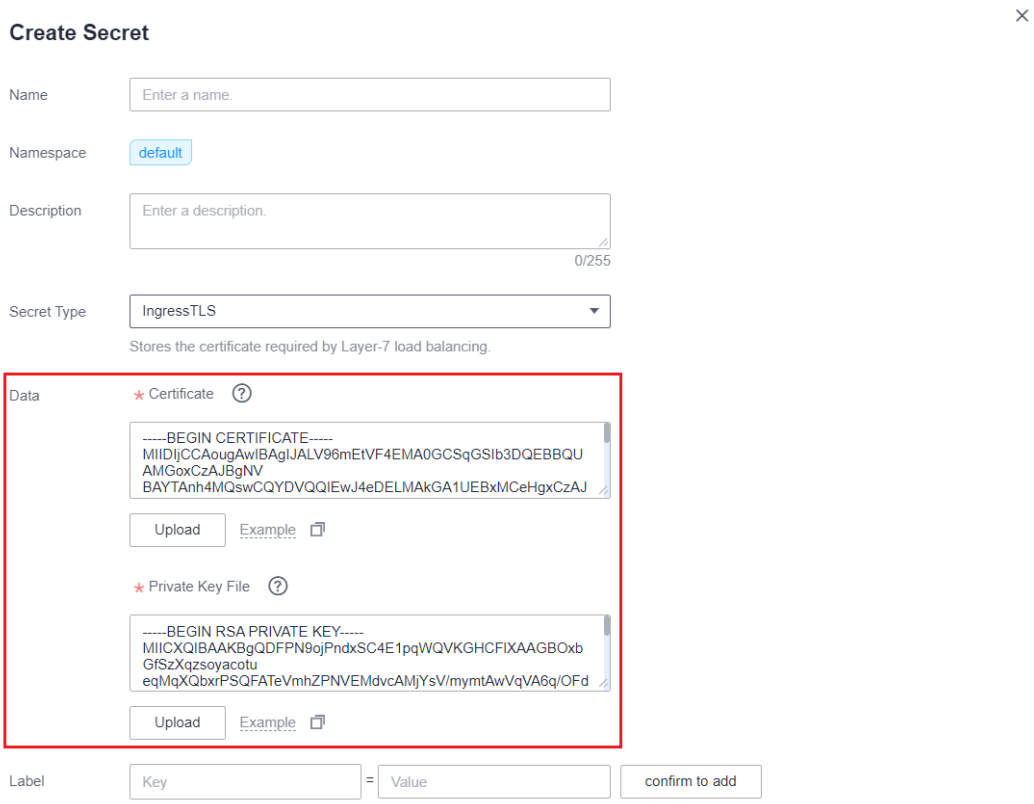
7.3.3 How Do I Obtain a TLS Key Certificate?

Scenario

If your ingress needs to use HTTPS, you must configure a secret of the IngressTLS or kubernetes.io/tls type when creating an ingress.

The following shows how to create an IngressTLS key certificate.

Figure 7-11 Creating a secret



When creating a secret, ensure that the certificate file uploaded in the secret data must match the private key file. Otherwise, the certificate file becomes invalid.

Solution

Generally, you need to obtain a valid certificate from the certificate provider. If you want to use it in the test environment, you can create a certificate and private key by the performing the following steps.

NOTE

Self-created certificates apply only to test scenarios. Such certificates are invalid and will affect browser access. Manually upload a valid one to ensure secure connections.

1. Generate a tls.key.

   ```
   openssl genrsa -out tls.key 2048
   ```

   The command will generate a private tls.key in the directory where the command is executed.

2. Generate a certificate using the private tls.key.

   ```
   openssl req -new -x509 -key tls.key -out tls.crt -subj /C=CN/ST=******/O=Devops/CN=example.com -days 3650
   ```

   The generated key must be in the following format:

   ```
   ----BEGIN RSA PRIVATE KEY-----
   ..........
   -----END RSA PRIVATE KEY-----
   ```

   The generated certificate must be in the following format:


```
-----BEGIN CERTIFICATE-----
.....
-----END CERTIFICATE-----
```

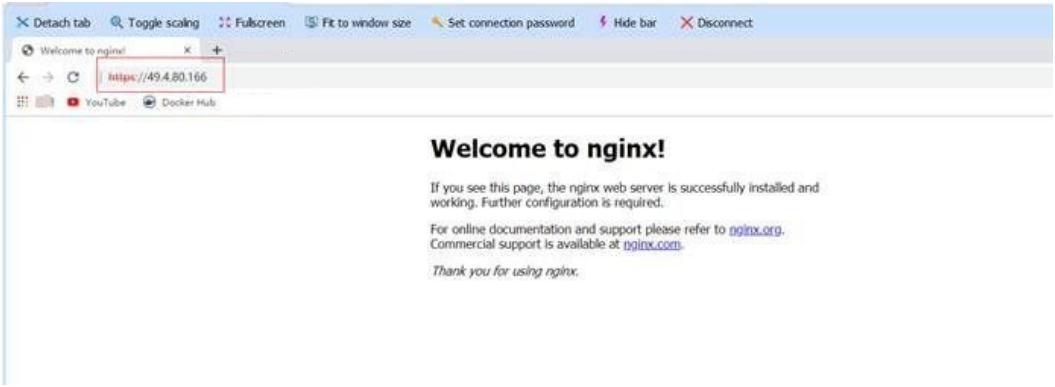
3. Import the certificate.

   When creating a TLS secret, import the certificate and private key file to the corresponding location.

Verification

The ingress address can be accessed through a browser. However, the certificate and secret are not issued by the CA, so the CA does not recognize them and shows a message saying they are insecure.

Figure 7-12 Verification result



7.3.4 How Do I Change the Security Group of Nodes in a Cluster in Batches?

Notes and Constraints

Do not add more than 1000 instances to the same security group. Otherwise, the security group performance may deteriorate. For more restrictions on security groups, see **Notes and Constraints**.

Procedure

Step 1 Log in to the VPC console and select the desired region and project in the upper left corner.

Step 2 In the navigation pane on the left, choose **Access Control** > **Security Groups**.

Step 3 On the **Security Groups** page, click **Manage Instance** in the **Operation** column.

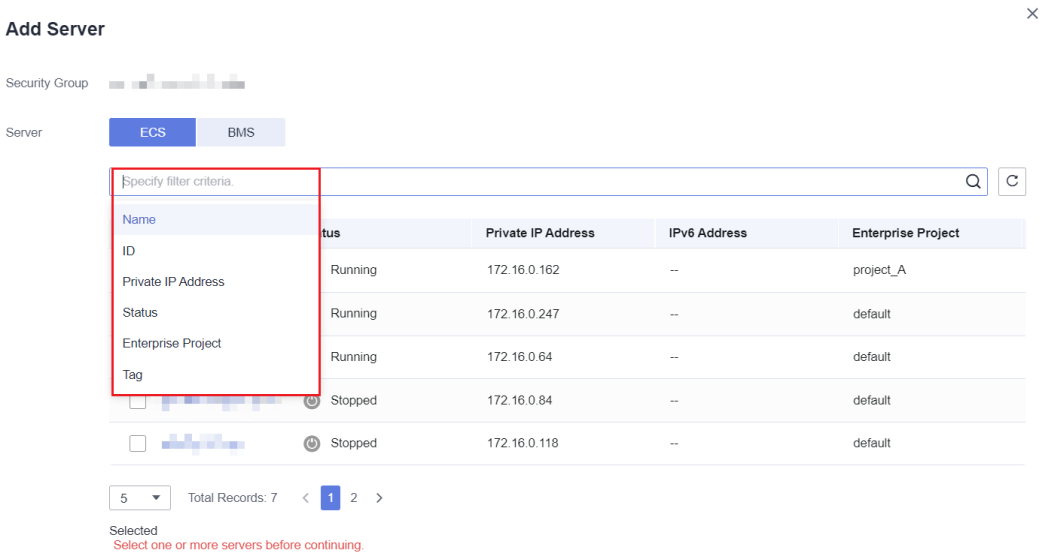
Step 4 On the **Servers** tab, click **Add**.

Step 5 Select the servers to be added to the security group and click **OK**. You can also search for servers by name, ID, private IP address, status, enterprise project, or tag.

You can change the maximum number of servers displayed on a page in the lower left corner to add a maximum of 20 servers to a security group at a time.

NOTE

After the node is added to a new security group, the original security group is retained. To remove the instance, click **Manage Instance** of the original security group and select the node servers to be removed.



----End

7.4 Network Configuration

7.4.1 How Does CCE Communicate with Other Huawei Cloud Services over an Intranet?

Common Huawei Cloud services that communicate with CCE over the intranet include RDS, DMS, Kafka, RabbitMQ, VPN, and ModelArts. The following scenarios are involved:

- In the same VPC network, CCE nodes can communicate with all services. When the containers communicate with other services, you need to check whether the security group rules in the inbound direction of the container CIDR block are enabled on the peer end. (This restriction applies only to CCE clusters that use the VPC network model.)
- If CCE nodes and other services are in different VPCs, you can use a peering connection or VPN to connect two VPCs. Note that the two VPC CIDR blocks cannot overlap with the container CIDR block. In addition, you need to configure a return route for the peer VPC or private network. For details, see **VPC Peering Connection**. (This restriction applies only to CCE clusters that use the VPC network model.)

NOTICE

- This logic works for all Huawei Cloud services.
- Clusters using the container tunnel network model support internal communication between services. There is no need to configure additional settings.
- You need to pay attention to the following points when configuring a cluster using the VPC network model:
  1. The source IP address displayed on the peer end is the container IP address.
  2. Custom routing rules added on CCE enable containers to communicate with each other on the nodes in a VPC.
  3. When a CCE container accesses other services, you need to **check whether the inbound security group rule or firewall of the container CIDR block is enabled on the peer end (destination end)**. For details, see **Security Group Configuration Examples**.
  4. If a VPN or VPC peering connection is used to enable communication between private networks, you need to configure **a VPC peering connection route that points to the container CIDR block** on the path and destination.
- Clusters using the **Cloud Native 2.0 network model** need to allow traffic from the container security groups based on service requirements. The default security group is named in the format of *{Cluster name}*-**cce-eni**-*{Random ID}*. For details, see **Security Group Rules in a CCE Turbo Cluster That Uses the Cloud Native 2.0 Network Model**.

7.4.2 How Do I Set the Port When Configuring the Workload Access Mode on CCE?

Workloads in a CCE cluster can access each other and can be accessed from the Internet.

- Workloads can access each other through ClusterIP (the virtual IP address of a cluster) and NodePort (a node IP address).

Table 7-11 Internal access description

Access Type	Description	Guide
Cluster IP (the virtual IP address of a cluster)	Used for mutual access between workloads in a cluster. For example, if a backend workload needs to communicate with a frontend workload, use this access type. When this access type is selected, a cluster IP address is automatically allocated.	• Container port: the port on a container on which the workload listens. The container port varies with the service. Typically, a container port is specified in the container image. • Service port: the port configured for the workload after the workload was associated with a Service. Enter an integer from 1 to 65535. Workloads in a cluster can access each other through *{Cluster IP}*:*{Access port number}*.
NodePort (through a node IP address)	The workload can be accessed through *{Node IP address}*:*{Node port number}*. If an EIP is bound to the node, the workload can be accessed from the external networks.	• Container port: the port on a container on which the workload listens. The container port varies with the service. Typically, a container port is specified in the container image. • Service port: the port configured for the workload after the workload was associated with a Service. Enter an integer from 1 to 65535. • Node port: the port on the node to which the container is mapped. After the configuration is complete, an actual port is open on all nodes in the project to which the user belongs. The workload can be accessed through *{Node IP}*:*{Node port number}*. If there are no special requirements, select **Automatically generated** so that the system automatically assigns an access port. If you select **Specified port**, enter an integer ranging from 30000 to 32767 and ensure that the value is unique in the cluster.

- A workload can be accessed from the Internet through NodePort (using an EIP), LoadBalancer, or DNAT.

  Table 7-12 External access description

Access Type	Description	Guide
NodePort (using an EIP)	If the node where the workload runs is bound with an EIP, the workload can be accessed through *{Node EIP}*:*{Node port number}*. The workload can then be accessed from the Internet.	• Container port: the port on a container on which the workload listens. The container port varies with the service. Typically, a container port is specified in the container image. • Service port: the port configured for the workload after the workload was associated with a Service. Enter an integer from 1 to 65535. • Node port: the port on the node to which the container is mapped. After the configuration is complete, an actual port is open on all nodes in the project to which the user belongs. The workload can be accessed through *{Node IP}*:*{Node port number}*. If there are no special requirements, select **Automatically generated** so that the system automatically assigns an access port. If you select **Specified port**, enter an integer ranging from 30000 to 32767 and ensure that the value is unique in the cluster.

Access Type	Description	Guide
LoadBalancer	ELB automatically distributes access traffic to multiple nodes to balance their service load. It supports higher levels of fault tolerance for workloads and expands workload service capabilities. You need to create an ELB instance in advance and select **ELB** as the CCE access type.	• Container port: the port on a container on which the workload listens. The container port varies with the service. Typically, a container port is specified in the container image. • Service port: the port registered with a load balancer. Enter an integer ranging from 1 to 65535. External users can use *{Virtual IP address of the load balancer}*:*{Service port number}* to access the workload.
DNAT	NAT gateways provide network address translation (NAT) for cloud servers so that multiple cloud servers can share an EIP. You need to buy a public NAT gateway in advance.	• Container port: the port on a container on which the workload listens. The container port varies with the service. Typically, a container port is specified in the container image. • Service port: the port registered on your NAT gateway. Enter an integer ranging from 1 to 65535. The system automatically creates DNAT rules. External users can access the workload through *{EIP of the NAT gateway}*:*{Service port number}*.

7.4.3 How Can I Achieve Compatibility Between Ingress's property and Kubernetes client-go?

Application Scenario

The Kubernetes ingress structure does not contain the **property** attribute. Therefore, the ingress created by client-go through API calling does not contain the **property** attribute. CCE provides a solution to ensure compatibility with the Kubernetes client-go.

Solution

When using client-go to create an ingress instance, make the following declaration in **annotation**:

kubernetes.io/ingress.property: '[{"host":"test.com","path":"/test","matchmode":"**STARTS_WITH**"},
{"host":"test.com","path":"/dw","matchmode":"**EQUAL_TO**"}]'

Matching rule: When a user calls the Kubernetes interface of CCE to create an ingress instance, CCE attempts to match the **host** and **path** fields in ingress rules. If the **host** and **path** fields in ingress rules are the same as those in annotation, CCE injects the **property** attribute to the path. The following is an example:

```
kind: Ingress
apiVersion: extensions/v1beta1
metadata:
  name: test
  namespace: default
  resourceVersion: '2904229'
  generation: 1
  labels:
    isExternal: 'true'
    zone: data
  annotations:
    kubernetes.io/ingress.class: cce
    kubernetes.io/ingress.property: '[{"host":"test.com","path":"/test","matchmode":"STARTS_WITH"},
{"Path":"/dw","MatchMode":"EQUAL_TO"}]'
spec:
  rules:
    - host: test.com
      http:
        paths:
          - path: /ss
            backend:
              serviceName: zlh-test
              servicePort: 80
          - path: /dw
            backend:
              serviceName: zlh-test
              servicePort: 80
```

The format after conversion is as follows:

```
kind: Ingress
apiVersion: extensions/v1beta1
metadata:
  name: test
  namespace: default
  resourceVersion: '2904229'
  generation: 1
  labels:
    isExternal: 'true'
    zone: data
  annotations:
    kubernetes.io/ingress.class: cce
    kubernetes.io/ingress.property: '[{"host":"test.com","path":"/ss","matchmode":"STARTS_WITH"},
{"host":"","path":"/dw","matchmode":"EQUAL_TO"}]'
spec:
  rules:
    - host: test.com
      http:
        paths:
          - path: /ss
            backend:
              serviceName: zlh-test
              servicePort: 80
            property:
              ingress.beta.kubernetes.io/url-match-mode: STARTS_WITH
          - path: /dw
            backend:
              serviceName: zlh-test
              servicePort: 80
```


Table 7-13 Descriptions of key parameters

Parameter	Type	Description
host	String	Domain name configuration. If this parameter is not set, **path** is automatically matched.
path	String	Matching path.
ingress.beta.kubernetes.io/url-match-mode	String	Route matching policy. The values are as follows: • **REGEX**: indicates regular expression match. • **STARTS_WITH**: indicates prefix match. • **EQUAL_TO**: indicates exact match.

Helpful Links

Layer-7 Load Balancing (Ingress)

7.4.4 How Do I Obtain the Actual Source IP Address of a Client After a Service Is Added into Istio?

Symptom

After Istio is enabled, the source IP address of the client cannot be obtained from access logs.

Solution

This section uses the Nginx application bound to an ELB Service as an example. The procedure is as follows:

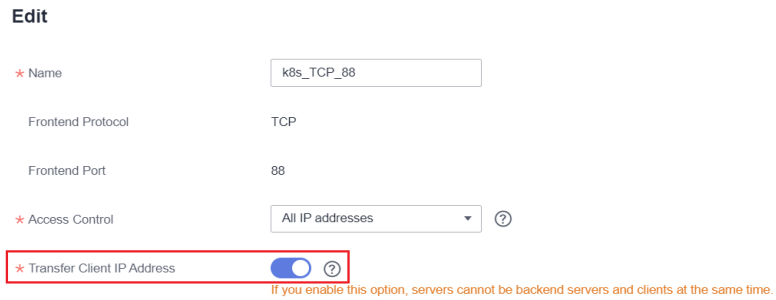
Step 1 Enabling the function of obtaining the client IP address on the load balancer

NOTE

Transparent transmission of source IP addresses is enabled for dedicated load balancers by default. You do not need to manually enable this function.

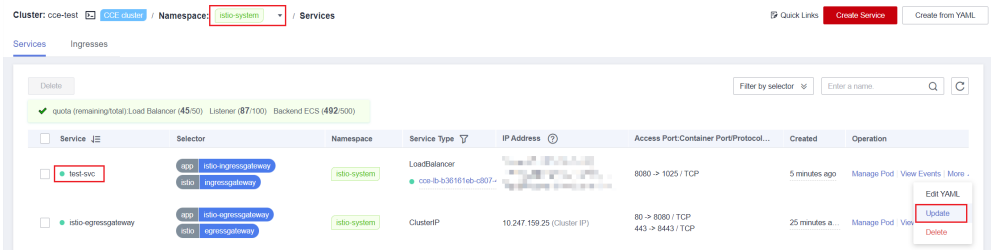
1. Click in the upper left corner of the management console and select a region and a project.
2. Choose **Service List** > **Networking** > **Elastic Load Balance**.
3. On the **Elastic Load Balance** page, click the name of the target load balancer.
4. Click the **Listeners** tab, locate the row containing the target listener, and click **Edit**. If modification protection exists, disable the protection on the basic information page of the listener and try again.
5. Enable **Transfer Client IP Address**.

Figure 7-13 Enabling the function

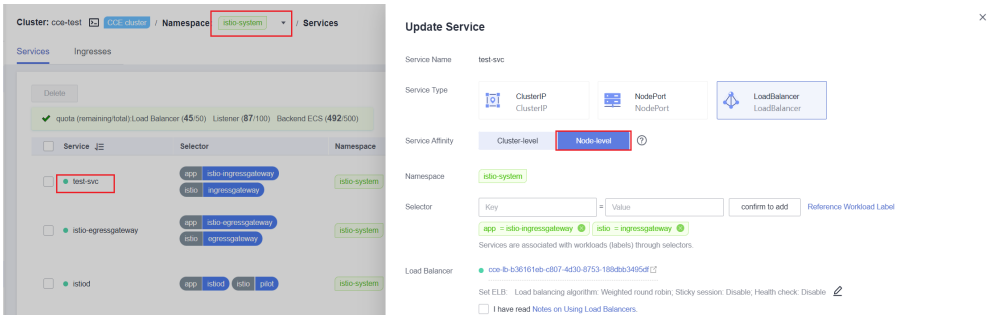


Step 2 Updating the gateway associated with a Service

1. Log in to the CCE console and click the cluster name to access the cluster console. In the navigation pane, choose **Services & Ingresses**.
2. On the displayed page, switch to the **istio-system** namespace and update the gateway associated with the Service.



3. Change the level of the Service automatically generated in the **istio-system** namespace to the node level.

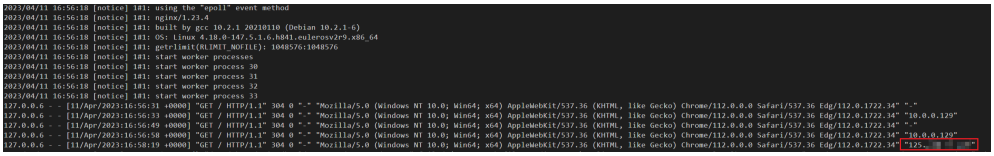


Step 3 Verifying the obtained source IP address

1. Use kubectl to access the cluster.
2. Query the Nginx application logs.

```
kubectl logs <pod_name>
```

In this example, the source IP address obtained by the Nginx application is as follows:



----End

7.4.5 Why Cannot an Ingress Be Created After the Namespace Is Changed?

Symptom

An ingress can be created in the default namespace, but cannot be created in other namespaces like **ns**.

Possible Cause

After a load balancer is created, an HTTP listener is created in the default namespace for port 80. In CCE, only ingresses of the same port can be created in the same namespace (the actual forwarding policies can be distinguished based on domain names and Services). Therefore, ingresses of the same port cannot be created in other namespaces (a port conflict message is displayed).

Solution

You can use YAML files to create ingresses. Port conflicts occur when you create ingresses on the CCE console, but not when you do so in the backend.

7.4.6 Why Is the Backend Server Group of an ELB Automatically Deleted After a Service Is Published to the ELB?

Symptom

After a Service is published to ELB, the workload is normal, but the pod port of the Service is not published in time. As a result, the backend server group of the ELB is automatically deleted.

Answer

1. If the ELB health check fails during ELB creation, the backend server group will be deleted and will not be added to the ELB after the Service becomes normal. If an existing SVC is updated, the backend server group is not deleted.
2. When a node is added or deleted, the node access mode in the cluster may change due to the cluster status change. To ensure normal service running, the ELB performs a refresh operation. The process is similar to that of updating the ELB.

Suggestions

Optimize the application to speed up the startup.

7.4.7 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.

7.4.8 How Can I Check Whether an ENI Is Used by a Cluster?

Scenarios

Pod subnets can be deleted from CCE Turbo clusters of v1.23.17-r0, v1.25.12-r0, v1.27.9-r0, v1.28.7-r0, v1.29.3-r0, or later versions.

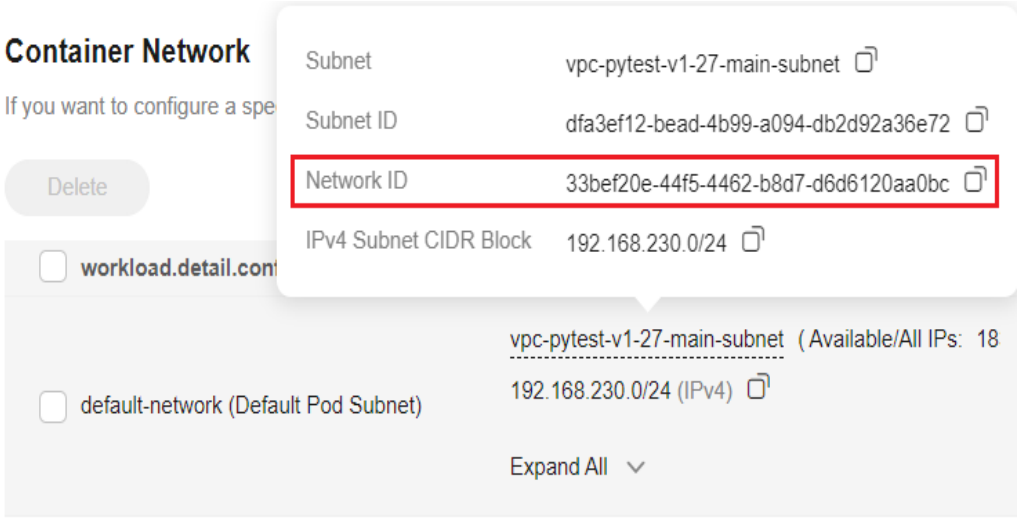
Deleting a pod subnet from a cluster can be risky. It is important to ensure that none of the ENIs currently in use by the cluster belong to the subnet, including those being used by pods and pre-bound to pods.

Procedure

Step 1 Log in to the CCE console and click the cluster name to access the cluster console.

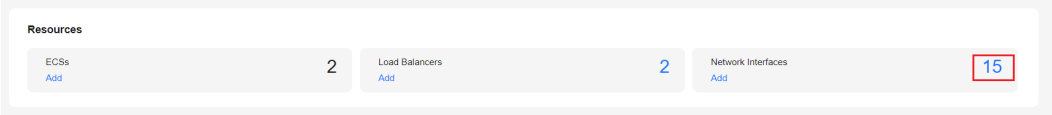
Step 2 In the navigation pane, choose **Settings** and click the **Network** tab.

Step 3 In the **Container Network** area, copy the network ID of the subnet. (The default-network is used as an example.)



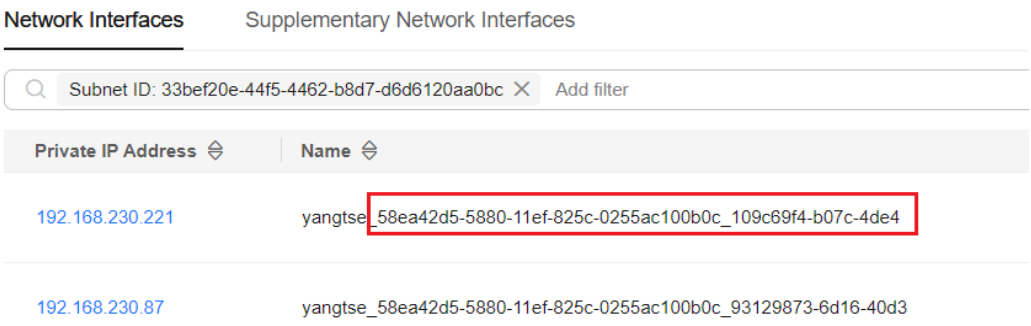
Step 4 Log in to the VPC console. In the navigation pane, choose **Virtual Private Cloud** > **Subnets**. In the right pane, obtain the target subnet based on the network ID.

Step 5 Click the subnet name to access the details page. Click the **Summary** tab, locate the **Resources** area, click the number next to **Network Interfaces**, and view the network interfaces and supplementary network interfaces associated with the subnet.



Step 6 Check the names or descriptions of the network interfaces. If the name or description of a network interface contains the ID of the cluster, it indicates that the network interface is used by the cluster. You can obtain the cluster ID on the **Overview** page of the CCE console.

To delete the subnet ENIs used in the cluster, submit a service ticket.



----End

7.4.9 How Can I Delete a Security Group Rule Associated with a Deleted Subnet?

Scenarios

Pod subnets can be deleted from CCE Turbo clusters of v1.23.17-r0, v1.25.12-r0, v1.27.9-r0, v1.28.7-r0, v1.29.3-r0, or later versions.

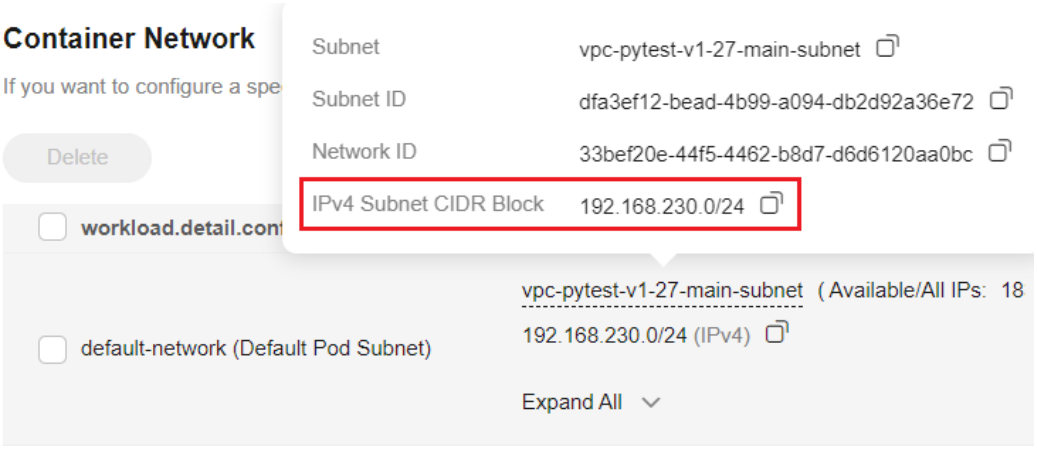
When you delete a subnet, CCE does not automatically remove the security group rules associated with the subnet in the default node security group created by CCE. You must manually delete these rules.

Procedure

Step 1 Log in to the CCE console and click the cluster name to access the cluster console.

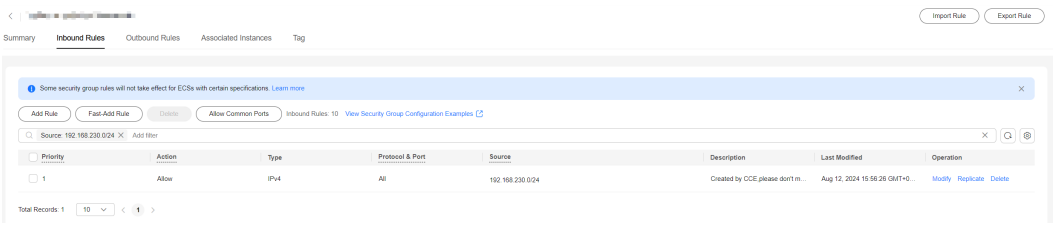
Step 2 In the navigation pane, choose **Settings** and click the **Network** tab.

Step 3 In the **Container Network** area, copy the IPv4 CIDR block of the subnet. (The default-network is used as an example.)



Step 4 In the navigation pane, choose **Overview**. In the **Networking Configuration** area, click the name of the default node security group.

Step 5 On the page displayed, click the **Inbound Rules** tab, locate the row containing the target subnet CIDR block based on the source IP address, and find the corresponding security group rule.



Step 6 Click **Delete** in the **Operation** column.

----End

7.4.10 How Can I Synchronize Certificates When Multiple Ingresses in Different Namespaces Share a Listener?

Context

In a cluster, multiple ingresses can share the same listener, allowing them to use the same port on a single load balancer. When two ingresses are set up with HTTPS certificates, the server certificate that is used will be based on the configuration of the earliest ingress.

If ingresses in separate namespaces use the same listener and TLS certificates, due to namespace isolation, the secrets associated with the TLS certificates may not display normally for the ingress that was created later.

The following table shows an example for the configurations of two ingresses.

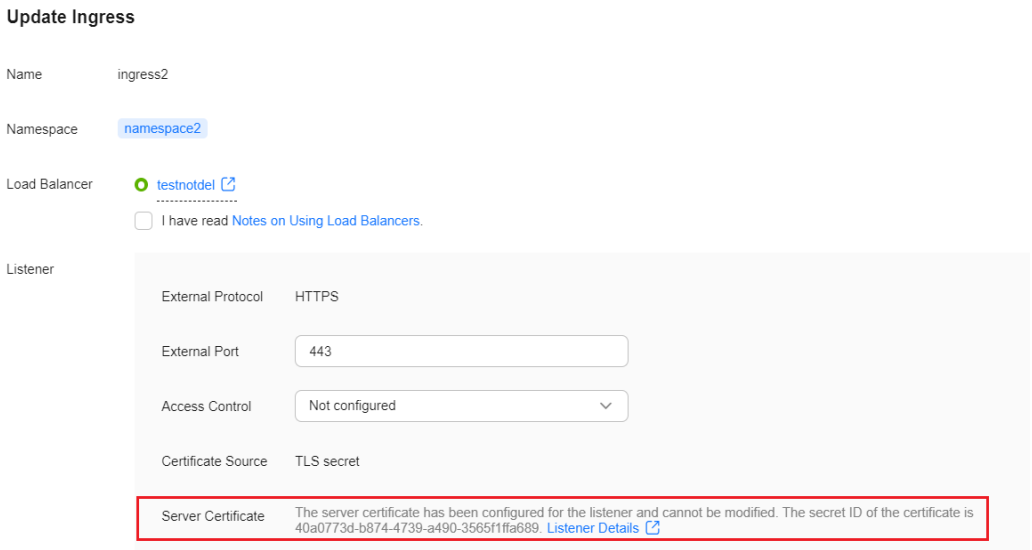
Ingress Name	ingress1	ingress2
Namespace	namespace1	namespace2
Creation Time	2024-04-01	2024-04-02

Protocol	HTTPS	HTTPS
Load Balancer	elb1	elb1
Port	443	443
Certificate Source	TLS key	TLS key
Secret Corresponding to the TLS Secret	namespace1/secret1	namespace2/secret2
Valid Certificate	namespace1/secret1	namespace1/secret1

Symptom

Within a given cluster, ingress1 and ingress2 are created in namespace1 and namespace2, respectively. Both ingresses connect to the same listener and use TLS certificates.

Ingress1's certificate is used because ingress1 was created first. But, ingress2 cannot read the configuration of secret1 because it is in a different namespace than namespace1. As a result, the configuration page of ingress2 will display the following information.



Solution

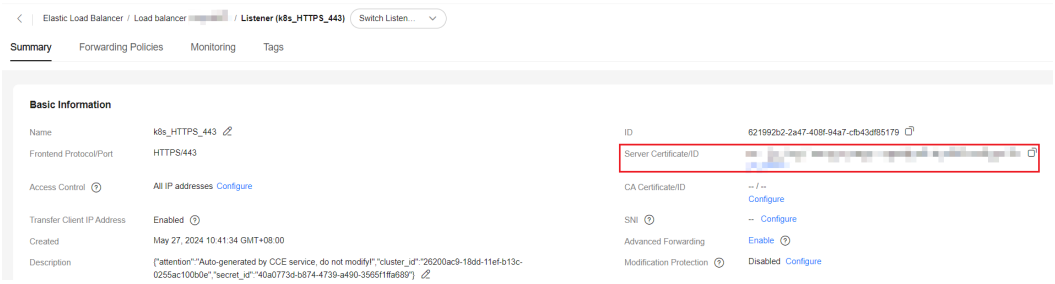
Each load balancer certificate has a corresponding TLS key, and the key content is identical. The CCE agency permissions enable access to certificate information without namespace restrictions. This means that you can switch the certificate source of ingress1 to the server certificate and assign the load balancer certificate corresponding to the TLS key. The configuration modification page of ingress2 displays the server certificate that works.

Step 1 Log in to the CCE console and click the cluster name to access the cluster console.

Step 2 In the navigation pane, choose **Services & Ingresses**, click the **Ingresses** tab, and click the load balancer link of ingress1 to go to the ELB console.

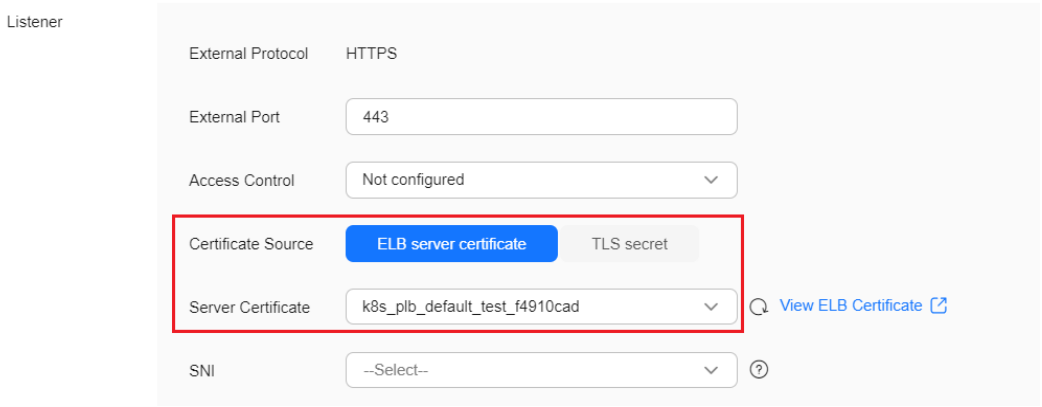
Step 3 Click the **Listeners** tab, find the listener based on the port configured for ingress1, and click the listener name to go to the details page.

Step 4 On the page displayed, find and record the server certificate.

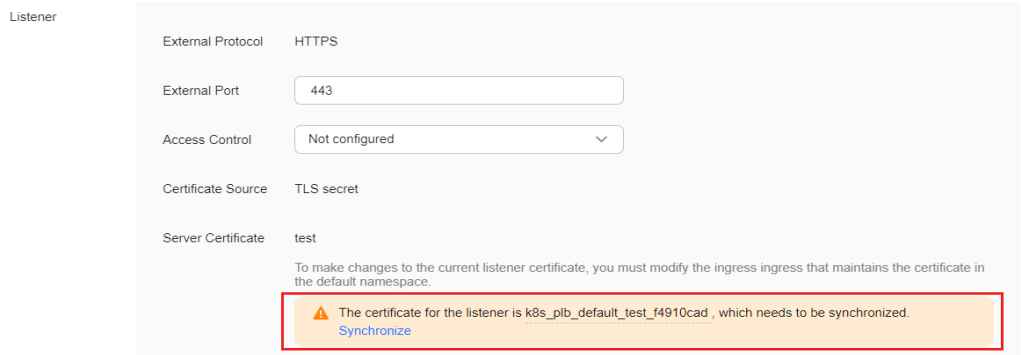


Step 5 Go back to the CCE console. On the **Ingresses** tab, locate the row containing ingress1 and choose **More** > **Update** in the **Operation** column. In the window that slides out from the right, set **Certificate Source** to **ELB server certificate**, select the server certificate obtained in the previous step, and click **OK**.

The certificate source of ingress1 has been changed from the TLS key to the server certificate, but the key content remains the same, as does the configuration that is applied.



Step 6 Switch to namespace2. On the **Ingresses** tab, locate the row containing ingress2 and choose **More** > **Update** in the **Operation** column. In the window that slides out from the right, locate the **Server Certificate** parameter in the **Listener** area, click **Synchronize**, and click **OK**.



Step 7 Verify that the configuration of ingress2 is displayed properly after the update is complete.

----End

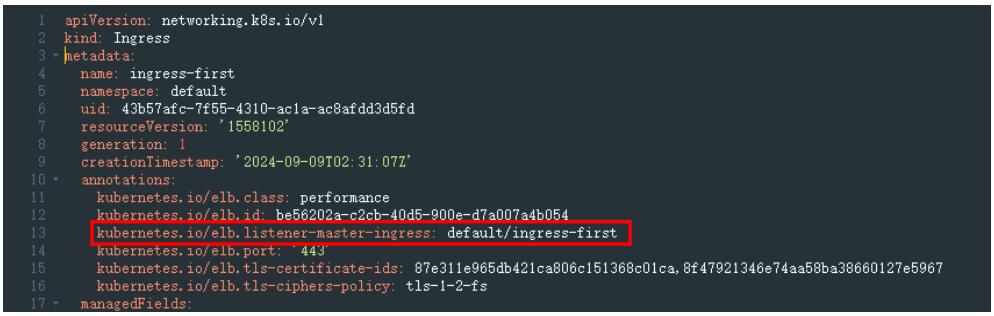
7.4.11 How Can I Determine Which Ingress the Listener Settings Have Been Applied To?

With CCE, you can associate multiple ingresses with a single load balancer listener and establish various forwarding policies. Listener configuration parameters are stored in annotations, which means that a listener can have different configuration parameters on different ingresses. This section describes how to determine which ingress the listener settings are applied to. It covers:

Determining Which Ingress the Listener Settings Have Been Applied To

Listener parameters can be configured for all ingresses associated with the same listener, so CCE uses the listener annotation configurations (excluding SNI certificates) from the earliest created ingress (the first ingress). The first ingress is determined by sorting the **metadata.createTimestamp** fields of the ingresses in ascending order.

- The first ingress information is written into annotations in clusters of v1.21.15-r0, v1.23.14-r0, v1.25.9-r0, v1.27.6-r0, v1.28.4-r0, v1.29.1-r0, and later. You can check **kubernetes.io/elb.listener-master-ingress** in the annotations of the existing ingresses.



- To get the first ingress in clusters earlier than v1.21.15-r0, v1.23.14-r0, v1.25.9-r0, v1.27.6-r0, v1.28.4-r0, and v1.29.1-r0, use the kubectl command to obtain the ingresses associated with the same load balancer listener, sort these ingresses in ascending order, based on their creation time, and check the first one.

  The query command is as follows: (Replace the load balancer ID and port number as needed.)

```
elb_id=${1}
elb_port=${2}
kubectl get ingress --all-namespaces --sort-by='.metadata.creationTimestamp' -o=custom-columns=Name:'metadata.name',Namespace:'metadata.namespace',elbID:'metadata.annotations.kubernetes\.io\/elb\.id',elbPort:'metadata.annotations.kubernetes\.io\/elb\.port',elbPorts:'metadata.annotations.kubernetes\.io\/elb\.listen-ports' | awk 'NR==1 {print; next} {if ($5 != "<none>") $4 = "<none>"; print}' | grep -E "^Name|${elb_id}" | grep -E "^Name|${elb_port}" | awk '{printf "%-30s %-30s %-38s %-10s %-10s\n", $1, $2, $3, $4, $5}'
```

  In the command output, the first column specifies the ingress names, the second specifies the namespaces, the third specifies the load balancer IDs, the fourth specifies the listener ports, and the fifth specifies the ports of multiple

listeners. (If multiple listener port numbers are configured, they will replace the listener port numbers and become effective.)

```
Name            Namespace   elbID                                elbPort   elbPorts
ingress-first   default     be56202a-c2cb-40d5-900e-d7a007a4b054   443      <none>
ingress-second  default     be56202a-c2cb-40d5-900e-d7a007a4b054   443      <none>
ingress-third   test        be56202a-c2cb-40d5-900e-d7a007a4b054   443      <none>
```

Configuring and Updating a Listener Certificate

You can configure an ingress certificate in a cluster using either of the following methods:

- Use a TLS certificate, keep it as a secret, and manage it on CCE. TLS certificates are configured using the **spec.tls** fields in ingresses. They will be automatically created, updated, or deleted via the ELB console.
- Use a certificate created in the ELB service. The certificate content is maintained on the ELB console. Such certificates are configured in the **annotation** fields in ingresses.

  ELB server certificates are also maintained on the ELB console, so you do not need to import the certificate content to CCE secrets. This allows for unified cross-namespace configuration. It is recommended that you use ELB server certificates to configure the certificates for ingresses.

  ELB server certificates are supported in clusters of versions v1.19.16-r2, v1.21.5-r0, and v1.23.3-r0.

To update the listener server certificate created by an ingress using a TLS certificate, follow these steps:

1. Check the command in **Determining Which Ingress the Listener Settings Have Been Applied To** and obtain all ingresses associated with the same listener.

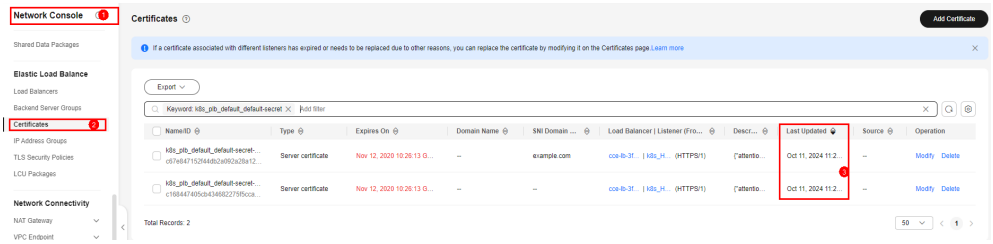
   The command output is as follows:

   ```
   Name            Namespace   elbID                                elbPort   elbPorts
   ingress-first   default     be56202a-c2cb-40d5-900e-d7a007a4b054   443      <none>
   ingress-second  default     be56202a-c2cb-40d5-900e-d7a007a4b054   443      <none>
   ingress-third   test        be56202a-c2cb-40d5-900e-d7a007a4b054   443      <none>
   ```

2. Obtain the certificate configuration in the first ingress.

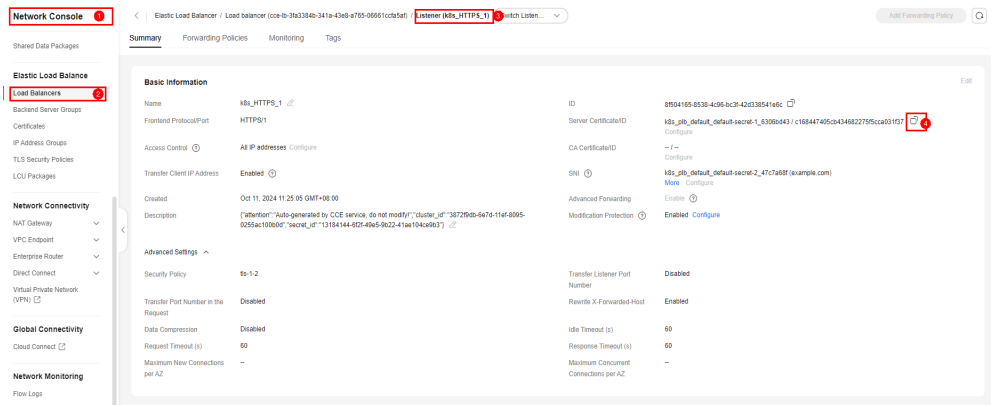
   ```
   apiVersion: networking.k8s.io/v1
   kind: Ingress
   metadata:
     name: ingress-first
     namespace: default
     ...
   spec:
     tls:
       - secretName: default-ns-secret-1
       - hosts:
           - 'example.com'
         secretName: default-ns-secret-2
   ...
   ```

3. Update the configurations of **default-ns-secret-1** and **default-ns-secret-2** in the first ingress.
4. After the key is updated, log in to the network console. In the navigation pane, choose **Elastic Load Balance** > **Certificates**. In the right pane, check whether the server certificate has been updated based on the update time.

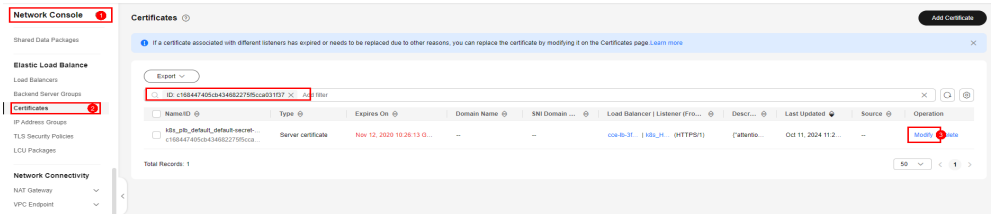


To update the listener server certificate created by an ingress using an ELB certificate, follow these steps:

1. On the ELB console, obtain the ID of the server certificate used by the load balancer listener.
   a. Log in to the network console. In the navigation pane, choose **Elastic Load Balance** > **Load Balancers**. In the right pane, click the name of the target load balancer.
   b. Click the **Listeners** tab and click the name of the target listener to go to the details page.
   c. On the **Summary** tab, obtain the server certificate ID.



2. In the navigation pane, choose **Elastic Load Balance** > **Certificates**, search for the certificate based on the obtained server certificate ID, locate the row containing the target certificate, and click **Modify** in the **Operation** column.



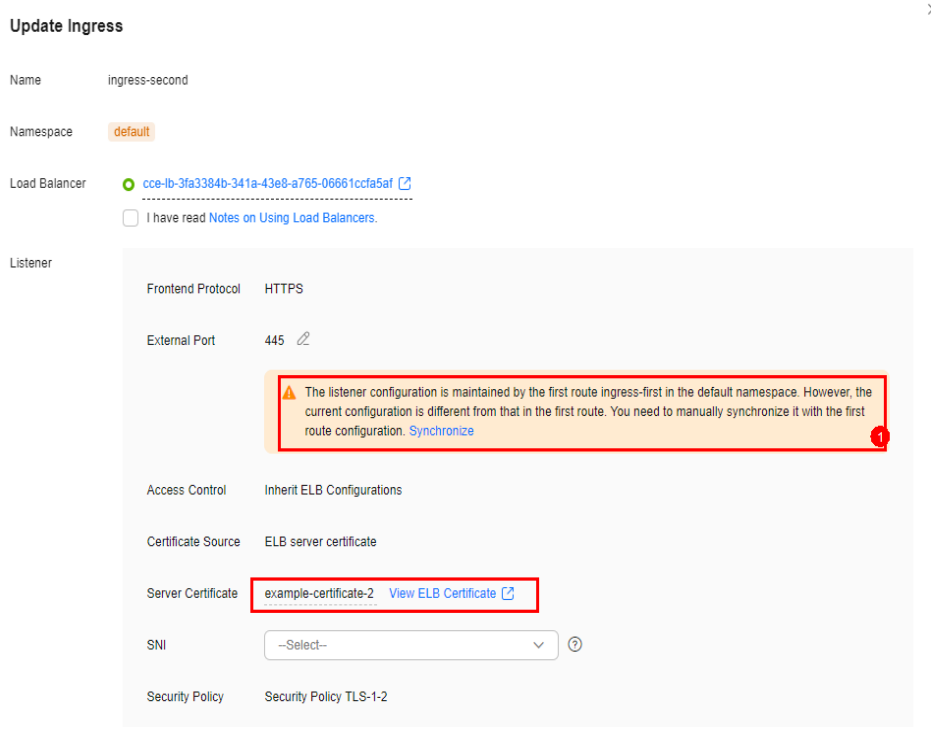
Impacts of Deleting the First Ingress on Listener Settings

Listener settings only apply to the first ingress configuration (excluding SNI certificates). If the first ingress is deleted, the earliest created ingress in use becomes the new first ingress, and the listener settings will be updated accordingly. This means that if the listener settings of the old and new first ingresses are different, there may be unexpected updates on the ELB console. To avoid this, check if the listener configuration of the ingress that will become the new first ingress is the same as the one for the original first ingress, or otherwise meets your expectations.

- You can synchronize the listener settings on the console. The procedure is as follows:
  a. Log in to the CCE console and click the cluster name to access the cluster console.
  b. In the navigation pane, choose **Services**, click the **Ingresses** tab, and choose **More** > **Update** in the **Operation** column.
  c. Click **Synchronize** to automatically synchronize the server certificate. This option is available when the listener settings of the ingress are inconsistent with those of the load balancer.

     NOTE

     If you synchronize a server certificate and an SNI certificate, and the current ingress is using a **TLS key**, the certificate will be replaced with an ELB server certificate. If the cluster version is earlier than v1.19.16-r2, v1.21.5-r0, v1.23.3-r0 and does not support ELB server certificates, you need to **manually synchronize the configurations using YAML**.



  d. Click **OK** to apply the modification.
- You can manually synchronize the listener settings using YAML. The procedure is as follows:
  a. Check the command in **Determining Which Ingress the Listener Settings Have Been Applied To** and obtain all ingresses associated with the same listener.

     The command output is as follows:

```
Name           Namespace  elbID                                  elbPort  elbPorts
ingress-first  default    be56202a-c2cb-40d5-900e-d7a007a4b054   443      <none>
ingress-second default    be56202a-c2cb-40d5-900e-d7a007a4b054   443      <none>
ingress-third  test       be56202a-c2cb-40d5-900e-d7a007a4b054   443      <none>
```

  b. Before deleting some ingresses such as **ingress-first** and **ingress-second**, synchronize the listener settings of **ingress-first** to the annotations of **ingress-third**.

If the listener server certificate was created using a TLS key, you need to synchronize the configurations saved in the ingress' **spec.tls** to **ingress-third**.

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: ingress-first
  namespace: default
  ...
spec:
  tls:
    - secretName: default-ns-secret-1
    - hosts:
        - 'example.com'
      secretName: default-ns-secret-2
...
```


8 Storage

8.1 How Do I Expand the Storage Capacity of a Container?

Application Scenario

The default storage size of a container is 10 GiB. 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 cluster name to access the cluster console.

Step 2 Choose **Nodes** from the navigation pane.

Step 3 Click the **Nodes** tab, locate the row containing the target node, and choose **More** > **Reset Node** in the **Operation** column.

NOTICE

Resetting a node may make the node-specific resources (such as local storage and workloads scheduled to this node) unavailable. Exercise caution when performing this operation to avoid impact on running services.

Step 4 Reconfigure node parameters.

If you need to adjust the container storage space, pay attention to the following configurations:

Storage Settings: Click **Expand** next to the data disk to configure the following parameter:

Space Allocation for Pods: indicates the base size of a pod. It is the maximum size that a workload's pods (including the container images) can grow to in the disk space. Proper settings can prevent pods from taking all the disk space

available and avoid service exceptions. It is recommended that the value is less than or equal to 80% of the container engine space. This parameter is related to the node OS and container storage rootfs and is not supported in some scenarios.

For more information about container storage space allocation, see **Data Disk Space Allocation**.

Step 5 After the node is reset, log in to the node and check whether the container capacity has been expanded. The command output varies with the container storage rootfs.

- Overlayfs: No independent thin pool is allocated. Image data is stored in **dockersys**. Run the following command to check whether the container capacity has been expanded:

  ```
  kubectl exec -it pod_name -- /bin/sh
  df -h
  ```

  If the information similar to the following is displayed, the overlay capacity has been expanded from 10 GiB to 15 GiB.

  ```
  Filesystem                Size  Used  Avail  Use%  Mounted on
  overlay                   15G   104K  15G    1%    /
  tmpfs                     64M   0     64M    0%    /dev
  tmpfs                     3.6G  0     3.6G   0%    /sys/fs/cgroup
  /dev/mapper/vgpaas-share  98G   4.0G  89G    5%    /etc/hosts
  ...
  ```

- Device Mapper: A thin pool is allocated to store image data. Run the following command to check whether the container capacity has been expanded:

  ```
  kubectl exec -it pod_name -- /bin/sh
  df -h
  ```

  If the information similar to the following is displayed, the thin pool capacity has been expanded from 10 GiB to 15 GiB.

  ```
  Filesystem                      Size  Used  Avail  Use%  Mounted on
  /dev/mapper/vgpaas-thinpool-snap-84  15G  232M  15G  2%  /
  tmpfs                           64M   0     64M    0%    /dev
  tmpfs                           3.6G  0     3.6G   0%    /sys/fs/cgroup
  /dev/mapper/vgpaas-kubernetes   11G   41M   11G    1%    /etc/hosts
  /dev/mapper/vgpaas-dockersys    20G   1.1G  18G    6%    /etc/hostname
  ...
  ```

----End

8.2 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 8-1 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.
Dedicated Distributed Storage Service (DSS)	Supported	Supported	Not supported

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.
- DSS volumes: CCE allows you to create DSS volumes and mount them to a container path. DSS offers dedicated, physical storage resources tailored to fulfill the needs of high-performance and low-latency storage. It is ideal for scenarios such as high-performance computing, real-time analysis and processing, and handling mixed workloads.

8.3 Can I Create a CCE Node Without Adding a Data Disk to the Node?

- If **System Component Storage** is set to **System Disk**, you do not need to add a data disk.
- Data disks are required if **System Component Storage** is set to **Data Disk**.

  A data disk dedicated for kubelet and the container engine will be attached to a new node. For details, see **Data Disk Space Allocation**. By default, CCE uses LVM to manage data disks. With LVM, you can adjust the disk space ratio for different resources on a data disk. For details, see **LVM Overview**.

  If the data disk is uninstalled or damaged, the container engine will malfunction and the node becomes unavailable.

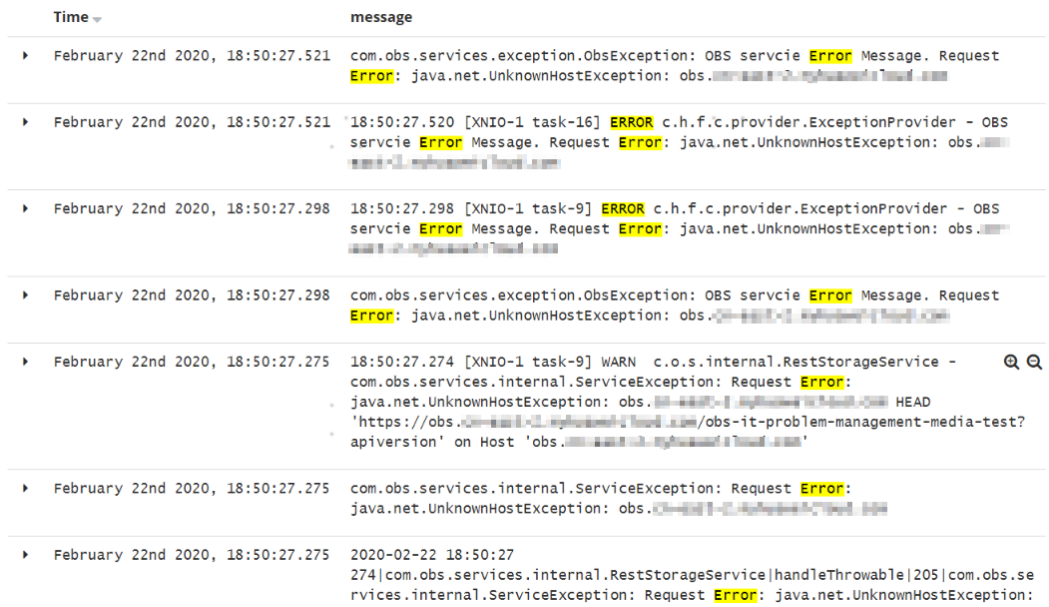
8.4 Can Data Be Restored If Underlying EVS Disks Are Deleted or Expired?

- If underlying EVS disks are deleted, you may encounter the following situations:
  - If you have created snapshots for the PVCs, you can restore data using the snapshots. For details, see **Using a Snapshot to Create a PVC**.
  - If you have enabled the EVS recycle bin, you can recover the EVS disks from the recycle bin. For details, see **Recovering Disks from the Recycle Bin**.
  - If the above scenarios are not met, the data cannot be restored.
- If an underlying yearly/monthly EVS disk has expired, you may encounter the following situations:
  - If the EVS disk is still in the retention period, you can renew the EVS disk to continue using it.
  - If the EVS disk is released due to arrears and a snapshot has been created for the PVC, you can restore data using the snapshot. For details, see **Using a Snapshot to Create a PVC**.

- If the EVS disk is released due to arrears and no snapshot has been created, the data cannot be restored.

8.5 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 servcie Error Message. Request Error: java.net.UnknownHostException: obs.
February 22nd 2020, 18:50:27.521	18:50:27.520 [XNIO-1 task-16] ERROR c.h.f.c.provider.ExceptionProvider - OBS servcie Error Message. Request Error: java.net.UnknownHostException: obs.
February 22nd 2020, 18:50:27.298	18:50:27.298 [XNIO-1 task-9] ERROR c.h.f.c.provider.ExceptionProvider - OBS servcie Error Message. Request Error: java.net.UnknownHostException: obs.
February 22nd 2020, 18:50:27.298	com.obs.services.exception.ObsException: OBS servcie Error Message. Request Error: java.net.UnknownHostException: obs.
February 22nd 2020, 18:50:27.275	18:50:27.274 [XNIO-1 task-9] WARN c.o.s.internal.RestStorageService - com.obs.services.internal.ServiceException: Request Error: java.net.UnknownHostException: obs. HEAD 'https://obs. /obs-it-problem-management-media-test?apiversion' on Host 'obs. '
February 22nd 2020, 18:50:27.275	com.obs.services.internal.ServiceException: Request Error: java.net.UnknownHostException: obs.
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 Error: 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 of 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. In this case, increase the proxy memory based on the actual scenario and then try to access the Service and upload files again.
2. Additionally, 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.

8.6 How Can I Achieve Compatibility Between ExtendPathMode and Kubernetes client-go?

Application 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 8-2 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	A third-level directory is added to the created volume directory/subdirectory to facilitate the obtaining of a single pod output file. The following types are supported. For details, see **Monitoring**. • **None**: The extended path is not configured. • **PodUID**: ID of a pod. • **PodName**: Name of a pod. • **PodUID/ContainerName**: ID of a pod or name of a container. • **PodName/ContainerName**: Name of a pod or container.

8.7 What Can I Do If a Storage Volume Fails to Be Created?

Symptom

The PV or PVC fails to be created. The following information is displayed in the event:

```
{"message": "Your account is suspended and resources can not be used.", "code": 403}
```

Possible Cause

The event indicates that your account is suspended or permissions are not granted to the account. Check whether your account is normal.

If the account is normal, check whether you have the permissions to access the namespace. You must have one of the development, O&M, and administrator permissions of the namespace, or have the customized permission to read and write PVCs and PVs. For details, see **Configuring Namespace Permissions (on the Console)**.

8.8 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.

8.9 Why Am I Getting an Error When I Changed the Owner Group and Permissions of the Mount Point of a General Purpose File System (SFS 3.0 Capacity-Oriented)?

Symptom

After a general purpose file system (SFS 3.0 Capacity-Oriented) is mounted to a directory in an OS, the directory becomes the mount point of the general purpose file system (SFS 3.0 Capacity-Oriented). When you run the **chown** and **chmod** commands to change the owner group or permissions of the mount point, information similar to the following is displayed:

```
chown: changing ownership of '***': Operation not permitted
```

Or

```
chmod: changing permissions of '***': Operation not permitted
```

Possible Cause

The owner group and permissions of the mount point of the general purpose file system (SFS 3.0 Capacity-Oriented) in the OS cannot be changed.

8.10 Why Cannot I Delete a PV or PVC Using the kubectl delete Command?

Symptom

An existing PV or PVC cannot be deleted by running the **kubectl delete** command and it remains in the terminating state.

Possible Cause

To prevent data loss caused by mis-deletion of PVs or PVCs, Kubernetes provides a data protection mechanism. A PV or PVC cannot be directly deleted using the **kubectl delete** command.

Solution

Run the **kubectl patch** command first to remove the protection mechanism and then delete the PV or PVC.

If you have run **kubectl delete** to delete a PV or PVC, the PV or PVC remains in the terminating state. It will be directly deleted after you run the **kubectl patch** command.

- Run the following command to delete a PV:
```
kubectl patch pv <pv-name> -p '{"metadata":{"finalizers":null}}'
kubectl delete pv <pv-name>
```
- Run the following command to delete a PVC:
```
kubectl patch pvc <pvc-name> -p '{"metadata":{"finalizers":null}}'
kubectl delete pvc <pvc-name>
```

8.11 What Should I Do If "target is busy" Is Displayed When a Pod with Cloud Storage Mounted Is Being Deleted?

Symptom

A pod remains in the terminating state when it is being deleted. When you get kubelet logs in the **/var/log/cce/kubernetes/kubelet.log** directory on the node where this pod runs, the following error message is displayed:

```
...unmount failed: exit status 32...Output: umount: <mount-path>: target is busy
```

Possible Cause

Other processes on the node are using the cloud storage device.

Solution

Log in to the node where the faulty pod runs, search for the process that is using the device, and stop that process.

Step 1 Log in to the node where the faulty pod runs.

Step 2 Run the following command to find the cloud storage device in the corresponding mount path: (*<mount-path>* specifies the mount path displayed in the error message.)

```
mount | grep <mount-path>
```

Information similar to the following is displayed:

```
/dev/sdatest on <mount-path> type ext4 (rw,relatime)
```

Step 3 Run the following command to find the ID of the process that uses the block storage:

```
fuser -mv /dev/sdatest
```

Step 4 Stop the process.

```
fuser -kmv /dev/sdatest
```

After the process is stopped, the cloud storage device is automatically uninstalled and the pod is deleted.

----End

8.12 What Should I Do If a Yearly/Monthly EVS Disk Cannot Be Automatically Created?

Symptom

When creating a yearly/monthly EVS disk, the payment permission cannot be added to **cce_cluster_agency**.

> **NOTE**
>
> To dynamically create yearly/monthly EVS disks, your cluster version must be v1.23.14-r0, v1.25.9-r0, v1.27.6-r0, v1.28.4-r0, or later. Additionally, you will need to have the Everest add-on 2.4.16 or later installed in the cluster.

Possible Cause

cce_cluster_agency is the system agency of CCE. It contains the cloud service resource operation permissions required by CCE components, but does not include the payment permission. For details, see **System Entrustment Description**. When creating yearly/monthly EVS disks, **cce_cluster_agency** must have the payment permissions, so you must manually add the **bss:order:pay** permission to **cce_cluster_agency**.

Solution

You can create a custom policy, add the **bss:order:pay** permission to it, and grant the policy to **cce_cluster_agency**.

Step 1 Create a custom policy.

1. Log in to the IAM console. In the navigation pane, choose **Permissions** > **Policies/Roles**. Then click **Create Custom Policy**.
2. Configure parameters for the policy.
   - **Policy Name**: Set it to **CCE Subscribe Operator**.
   - **Policy View**: Select **JSON**.
   - **Policy Content**: Configure it as follows:

```
{
    "Version": "1.1",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "bss:order:pay"
            ]
        }
    ]
}
```

3. Click **OK**.

Step 2 Grant the custom policy to **cce_cluster_agency**.

1. Log in to the IAM console. In the navigation pane, choose **Agencies**.
2. Locate the agency named **cce_cluster_agency** and click **Authorize**.

3. Search for the **CCE Subscribe Operator** custom policy, select it, and click **Next**.
4. Select an authorization scope as needed.

   By default, **All resources** is selected.
5. Click **OK**.

Step 3 Go back to the CCE console, create a yearly/monthly EVS disk again, and verify that this problem has been resolved.

----End

8.13 How Do I Restore a Disk After It Is Mistakenly Detached from a Storage Pool?

A storage pool is a custom resource (**nodelocalvolumes**) created by Everest. It is not recommended that you manually perform any operations on this type of resource under normal circumstances. Everest scans for idle disks every minute and verifies that the disks added to the storage pool are functioning properly.

Everest uses LVM to manage storage pools. Both the local PVs and local ephemeral volumes (EVs) are a volume group (VG) in LVM.

- VG used by the local PVs: **vg-everest-localvolume-persistent**
- VG used by the local EVs: **vg-everest-localvolume-ephemeral**

This section describes how to restore a local PV. To restore a local EV, use the corresponding VG.

NOTICE

This section's guide is solely intended for restoring an unavailable storage pool from that a disk has been accidentally detached from. Once the storage pool is restored, you can import PVs or EVs, but the original data cannot be recovered.

Symptom

A disk of a storage pool is detached by mistake, resulting in unavailable node storage pool.

Fault Locating

Run the kubectl command to check the **nodelocalvolumes** resource.

```
kubectl get nodelocalvolumes.localvolume.everest.io -n kube-system 192.168.1.137 -o yaml
```

The message displayed in **status** is "/dev/vde is lost."

```
...
status:
```



```
lastUpdateTime: "2024-07-16T07:13:55Z"
message: the device 6eef2886-f5ad-4b3f-a:/dev/vde is lost
phase: Unavailable
totalDisks:
- capacity: 100Gi
  name: /dev/vdb
  uuid: 3511993c-61e6-4faa-a
usedDisks:
- totalSize: 102392Mi
  type: persistent
  usedSize: 10Gi
  volume-group: vg-everest-localvolume-persistent
  volumes:
  - capacity: 50Gi
    name: /dev/vdd
    pv-uuid: Jo9Uur-evVi-RLWM-yaln-J6rz-3QCo-aHpvwB
    uuid: 40b8b92b-5852-4a97-9
  - capacity: 50Gi
    name: /dev/vde
    pv-uuid: ZxA9kY-5C28-96Z9-ZjOE-dCrc-yTgp-DOhUHo
    uuid: 6eef2886-f5ad-4b3f-a
```

Alarms are generated when you check LVM resources on the node.

- Run **vgdisplay**. The following alarm is displayed:

  ```
  WARNING: Couldn't find device with uuid ZxA9kY-5C28-96Z9-ZjOE-dCrc-yTgp-DOhUHo.
  WARNING: VG vg-everest-localvolume-persistent is missing PV ZxA9kY-5C28-96Z9-ZjOE-dCrc-yTgp-DOhUHo (last written to /dev/vde).
  --- Volume group ---
  VG Name               vg-everest-localvolume-persistent
  System ID
  Format                lvm2
  Metadata Areas        1
  Metadata Sequence No  3
  VG Access             read/write
  VG Status             resizable
  MAX LV                0
  Cur LV                1
  Open LV               1
  Max PV                0
  Cur PV                2
  Act PV                1
  VG Size               99.99 GiB
  PE Size               4.00 MiB
  Total PE              25598
  Alloc PE / Size       2560 / 10.00 GiB
  Free  PE / Size       23038 / 89.99 GiB
  VG UUID               31LHdA-yZPV-M7JX-ttwK-aynz-IyxY-usp22p
  ```

- Run **lvdisplay**. The following alarm is displayed:

  ```
  WARNING: Couldn't find device with uuid ZxA9kY-5C28-96Z9-ZjOE-dCrc-yTgp-DOhUHo.
  WARNING: VG vg-everest-localvolume-persistent is missing PV ZxA9kY-5C28-96Z9-ZjOE-dCrc-yTgp-DOhUHo (last written to /dev/vde).
  --- Logical volume ---
  LV Path                /dev/vg-everest-localvolume-persistent/pvc-fad963ae-f168-4e6c-a060-088b68ee503e
  LV Name                pvc-fad963ae-f168-4e6c-a060-088b68ee503e
  VG Name                vg-everest-localvolume-persistent
  LV UUID                NzZvao-peoH-q4Zr-YTs3-WgqX-vxt5-W20Og2
  LV Write Access        read/write
  LV Creation host, time , 2024-07-16 12:03:20 +0800
  LV Status              available
  # open                 1
  LV Size                10.00 GiB
  Current LE             2560
  Segments               1
  Allocation             inherit
  Read ahead sectors     auto
  - currently set to     256
  Block device           252:1
  ```


- Run **pvdisplay**. The following alarm is displayed:

```
 WARNING: Couldn't find device with uuid ZxA9kY-5C28-96Z9-ZjOE-dCrc-yTgp-DOhUHo.
 WARNING: VG vg-everest-localvolume-persistent is missing PV ZxA9kY-5C28-96Z9-ZjOE-dCrc-yTgp-DOhUHo (last written to /dev/vde).
 --- Physical volume ---
 PV Name               /dev/vdd
 VG Name               vg-everest-localvolume-persistent
 PV Size               50.00 GiB / not usable 4.00 MiB
 Allocatable           yes
 PE Size               4.00 MiB
 Total PE              12799
 Free PE               10239
 Allocated PE          2560
 PV UUID               Jo9Uur-evVi-RLWM-yaln-J6rz-3QCo-aHpvwB

 --- Physical volume ---
 PV Name               [unknown]
 VG Name               vg-everest-localvolume-persistent
 PV Size               50.00 GiB / not usable 4.00 MiB
 Allocatable           yes
 PE Size               4.00 MiB
 Total PE              12799
 Free PE               12799
 Allocated PE          0
 PV UUID               ZxA9kY-5C28-96Z9-ZjOE-dCrc-yTgp-DOhUHo
```

- Run **pvs**. The following alarm is displayed:

```
 WARNING: Couldn't find device with uuid ZxA9kY-5C28-96Z9-ZjOE-dCrc-yTgp-DOhUHo.
 WARNING: VG vg-everest-localvolume-persistent is missing PV ZxA9kY-5C28-96Z9-ZjOE-dCrc-yTgp-DOhUHo (last written to /dev/vde).
 PV         VG                                Fmt  Attr PSize   PFree
 /dev/vdc   vgpaas                            lvm2 a--  <50.00g      0
 /dev/vdd   vg-everest-localvolume-persistent lvm2 a--  <50.00g <40.00g
 [unknown]  vg-everest-localvolume-persistent lvm2 a-m  <50.00g <50.00g
```

Solution

Step 1 Restore the **nodelocalvolumes** resource.

```
kubectl edit nodelocalvolumes.localvolume.everest.io -n kube-system 192.168.1.137
```

Modify the preceding resource, delete the lost disk from **spec.volumes.type: persistent**, and delete the **status** field from the resource.

Step 2 Remove the corresponding PV from the VG.

NOTE

The information for VG is stored on its respective disk.

- If a VG has multiple disks but some PVs are missing, a message will appear indicating missing PVs.
- If a disk is removed from the VG, the VG will not display by running **vgdisplay**. If you cannot see **vg-everest-localvolume-persistent** or **vg-everest-localvolume-ephemeral** using the **vgdisplay** command, you can skip this step.

Run the following command to remove all lost PVs from the VG. The VG name for local PVs is *vg-everest-localvolume-persistent*. If a local EV is restored, the VG name changes to *vg-everest-localvolume-ephemeral*.

```
vgreduce --removemissing vg-everest-localvolume-persistent
```

Information similar to the following is displayed:

```
 WARNING: Couldn't find device with uuid ZxA9kY-5C28-96Z9-ZjOE-dCrc-yTgp-DOhUHo.
 WARNING: VG vg-everest-localvolume-persistent is missing PV ZxA9kY-5C28-96Z9-ZjOE-dCrc-yTgp-DOhUHo (last written to /dev/vde).
```



```
WARNING: Couldn't find device with uuid ZxA9kY-5C28-96Z9-ZjOE-dCrc-yTgp-DOhUHo.
Wrote out consistent volume group vg-everest-localvolume-persistent.
```

Run **vgdisplay** again and check that the command output is normal.

```
vgdisplay vg-everest-localvolume-persistent
```

Information similar to the following is displayed:

```
--- Volume group ---
  VG Name               vg-everest-localvolume-persistent
  System ID
  Format                lvm2
  Metadata Areas        1
  Metadata Sequence No  4
  VG Access             read/write
  VG Status             resizable
  MAX LV                0
  Cur LV                1
  Open LV               1
  Max PV                0
  Cur PV                1
  Act PV                1
  VG Size               <50.00 GiB
  PE Size               4.00 MiB
  Total PE              12799
  Alloc PE / Size       2560 / 10.00 GiB
  Free  PE / Size       10239 / <40.00 GiB
  VG UUID               31LHdA-yZPV-M7JX-ttwK-aynz-lyxY-usp22p
```

Step 3 Restart everest-csi-driver on the corresponding node.

1. Check the pod name of everest-csi-driver.

   ```
   kubectl get pod -A -owide
   ```

   Information similar to the following is displayed:

   ```
   NAMESPACE     NAME                                READY   STATUS             RESTARTS     AGE
   IP              NODE            NOMINATED NODE   READINESS GATES
   kube-system   everest-csi-driver-7clbg              1/1     Running            0            5d4h
   192.168.1.38    192.168.1.38    <none>           <none>
   kube-system   everest-csi-driver-jvj9f              1/1     Running            0            5d4h
   192.168.1.137   192.168.1.137   <none>           <none>
   ```

2. Delete the pod on the node whose IP address is 192.168.1.137.

   ```
   kubectl delete pod -n kube-system everest-csi-driver-jvj9f
   ```

3. Verify that the **nodelocalvolumes** status becomes normal and continue to import PVs.

Step 4 Restart the node.

Sometimes, even after resolving the issue, CCE Node Problem Detector may still show the node as unavailable due to abnormal metrics. If this happens, restarting the node can solve the problem.

----End

8.14 How Can I Delete the Underlying Storage Volume If It Remains After a Dynamically Created PVC Is Deleted?

Symptom

After a dynamically created PVC with the reclaim policy in its StorageClass set to **Delete** was deleted from a cluster, the underlying storage volume of the PVC is not deleted simultaneously.

Trigger Conditions

- A PVC and its bound PV are deleted simultaneously.
- The PV bound to a PVC is deleted first and then the PVC is deleted, but the PV deletion fails due to the PVC/PV binding.

Possible Cause

Under normal circumstances, when a dynamically created PVC is deleted in the open source csi-provisioner module, the PVC is deleted first, followed by a change in the status of the PV bound to the PVC to **Released**. The csi-provisioner module then listens for PV changes, proceeds to delete the underlying storage volume, and deletes the PV, completing the deletion chain.

During abnormal operations, the PV bound to a PVC may be directly deleted without deleting the PVC first. However, the **kubernetes.io/pv-protection** finalizer on the PV prevents immediate deletion. Instead, **deletionTimestamp** is added to the PV. After the PVC is deleted, the PV status is changed to **Released**. Although csi-provisioner listens for PV changes, it skips the process of deleting the underlying storage volume because **deletionTimestamp** has been added to the PV. As a result, csi-provisioner directly deletes the PV, and both the PVC and PV are deleted, but the underlying storage volume remains. For details about the code logic, see **controller**.

Solution

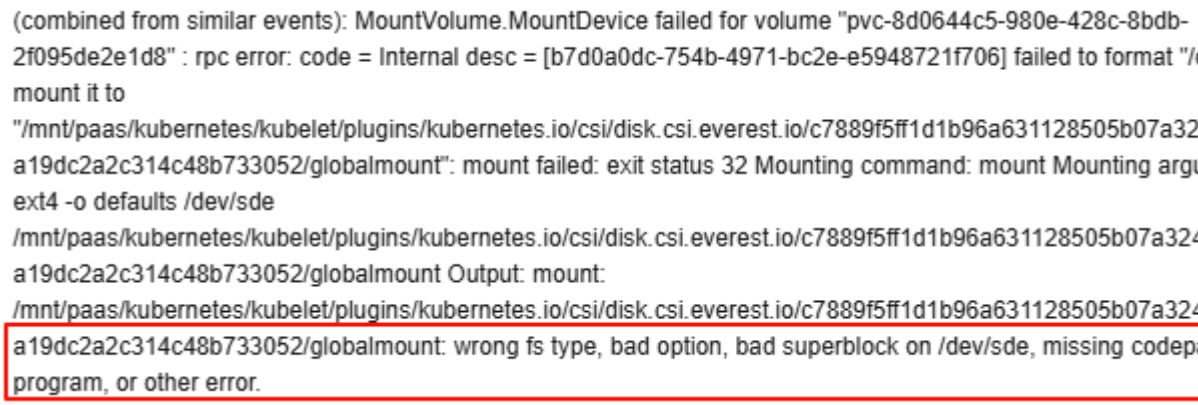
1. Manually delete the residual underlying storage volumes.
2. Directly delete the dynamically created PVCs that remained after the deletion. The PVs and underlying storage volumes will be deleted automatically.

8.15 Why Does a PV Fail to Be Mounted to a Pod After the PV Is Re-bound to a Released EVS Disk?

Symptom

When an EVS volume is mounted to a pod, the error message "wrong fs type, bad option, bad superblock on /dev/sda, missing codepage or helper program, or other error" is displayed.

Figure 8-1 Error message example



Possible Cause

The file system type of the underlying EVS disk must match that of the PV when it is first bound to a pod. If a released EVS disk is re-bound to a PV, the PV's file system type must remain consistent with the original PV. If the file system types do not match, CCE detects inconsistencies between the PV file system type and the disk file system type, resulting in a mounting failure.

Solution

Create a new EVS disk for future use.

9 Namespace

9.1 How Many Namespaces Can Be Created in a Cluster?

In a Kubernetes cluster, there is no set limit on the number of namespaces that can be created. However, the actual number of namespaces that can be created may be limited by other factors.

- Resource limit

  In a CCE cluster, as the cluster scale increases, the resources of the master nodes also increase, allowing for support of more namespaces. However, each namespace consumes control plane resources to store metadata. Therefore, a high number of namespaces could lead to resource shortages on the cluster control plane, impacting cluster stability.

- Performance limit

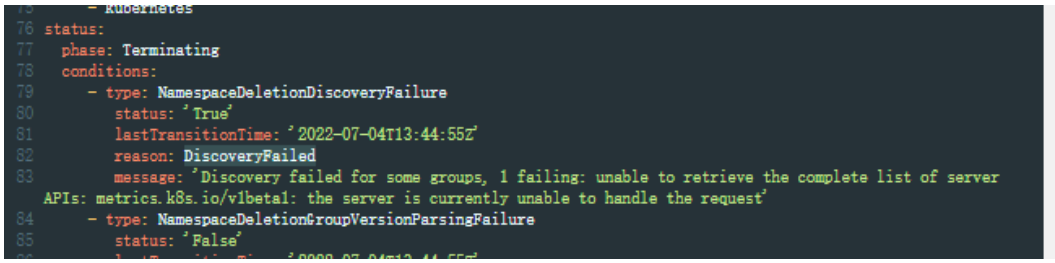
  For the cluster control plane, having a large number of namespaces can overload the API server and slow down response speed. Retrieving resources from all namespaces requires traversing more data, leading to decreased cluster performance.

It is recommended that you create namespaces only when necessary. Additionally, setting appropriate resource quotas and limits for each namespace can help maintain cluster stability and performance.

9.2 What Should I Do If a Namespace Fails to Be Deleted 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.



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 Cause

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.

9.3 How Do I Delete a Namespace in the Terminating State?

A Kubernetes namespace is typically in the active or terminating state. If a namespace is deleted when there are still running resources, the namespace enters the terminating state. In this case, the namespace will be automatically deleted only after Kubernetes reclaims the resources in it.

However, in some cases, even if no resource is running in the namespace, the namespace in the terminating state still cannot be deleted.

To solve this problem, perform the following operations:

Step 1 View the namespace details.

```
$ kubectl  get ns  | grep rdb
rdbms              Terminating   6d21h

$ kubectl  get ns rdbms -o yaml
```



```
apiVersion: v1
kind: Namespace
metadata:
  annotations:
    kubectl.kubernetes.io/last-applied-configuration: |
      {"apiVersion":"v1","kind":"Namespace","metadata":{"annotations":{},"name":"rdbms"}}
  creationTimestamp: "2020-05-07T15:19:43Z"
  deletionTimestamp: "2020-05-07T15:33:23Z"
  name: rdbms
  resourceVersion: "84553454"
  selfLink: /api/v1/namespaces/rdbms
  uid: 457788ddf-53d7-4hde-afa3-1fertg21ewe1
spec:
  finalizers:
  - kubernetes
status:
  phase: Terminating
```

Step 2 View resources in the namespace.

```
# View resources that can be isolated using namespaces in the cluster.
$ kubectl api-resources -o name --verbs=list --namespaced | xargs -n 1 kubectl get --show-kind --ignore-not-found -n rdbms
```

The command output shows that no resource is occupied in the **rdbms** namespace.

Step 3 Delete the namespace.

Directly delete the **rdbms** namespace.

```
$ kubectl  delete ns rdbms
Error from server (Conflict): Operation cannot be fulfilled on namespaces "rdbms": The system is ensuring all content is removed from this namespace.  Upon completion, this namespace will automatically be purged by the system.
```

The deletion fails and a message is displayed, indicating that the system will automatically delete the namespace after confirming that no resource is running in it.

Step 4 Forcibly delete the namespace.

```
$ kubectl  delete ns rdbms --force --grace-period=0
warning: Immediate deletion does not wait for confirmation that the running resource has been terminated. The resource may continue to run on the cluster indefinitely.
Error from server (Conflict): Operation cannot be fulfilled on namespaces "rdbms": The system is ensuring all content is removed from this namespace.  Upon completion, this namespace will automatically be purged by the system.
```

After running this command, the namespace still cannot be deleted.

Step 5 Call the Kubernetes native APIs to delete resources in the namespace. In most cases, resources in a namespace cannot be forcibly deleted. Use the Kubernetes native APIs instead.

View the namespace details.

```
$ kubectl  get ns rdbms  -o json > rdbms.json
```

Check the JSON configuration defined by the namespace, edit the JSON file, and delete the **spec** part.

```
$ cat rdbms.json
{
    "apiVersion": "v1",
    "kind": "Namespace",
    "metadata": {
        "annotations": {
            "kubectl.kubernetes.io/last-applied-configuration": "{\"apiVersion\":\"v1\",\"kind\":\"Namespace\",\"metadata\":{\"annotations\":{},\"name\":\"rdbms\"}}\n"
```



```
        },
        "creationTimestamp": "2019-10-14T12:17:44Z",
        "deletionTimestamp": "2019-10-14T12:30:27Z",
        "name": "rdbms",
        "resourceVersion": "8844754",
        "selfLink": "/api/v1/namespaces/rdbms",
        "uid": "29067ddf-56d7-4cce-afa3-1fbdbb221ab1"
    },
    "spec": {
        "finalizers": [
            "kubernetes"
        ]
    },
    "status": {
        "phase": "Terminating"
    }
}
```

After the PUT request is executed, the namespace is automatically deleted.

```
$ curl --cacert /root/ca.crt --cert /root/client.crt --key /root/client.key -k -H "Content-Type:application/json" -
X PUT --data-binary @rdbms.json https://x.x.x.x:5443/api/v1/namespaces/rdbms/finalize
{
  "kind": "Namespace",
  "apiVersion": "v1",
  "metadata": {
    "name": "rdbms",
    "selfLink": "/api/v1/namespaces/rdbms/finalize",
    "uid": "29067ddf-56d7-4cce-afa3-1fbdbb221ab1",
    "resourceVersion": "8844754",
    "creationTimestamp": "2019-10-14T12:17:44Z",
    "deletionTimestamp": "2019-10-14T12:30:27Z",
    "annotations": {
      "kubectl.kubernetes.io/last-applied-configuration": "{\"apiVersion\":\"v1\",\"kind\":\"Namespace
\",\"metadata\":{\"annotations\":{},\"name\":\"rdbms\"}}\n"
    }
  },
  "spec": {

  },
  "status": {
    "phase": "Terminating"
  }
```

If the namespace still cannot be deleted, check whether the **finalizers** field exists in the metadata. If the field exists, run the following command to access the namespace and delete the field:

```
kubectl edit ns rdbms
```

NOTE

- For details about how to obtain a cluster certificate, see **Accessing a Cluster Using an X.509 Certificate**.
- **https://***x.x.x.x***:5443** indicates the address for accessing the cluster. To obtain the private IP address, log in to the CCE console, access the cluster console, and view the connection information.

Step 6 Check whether the namespace has been deleted.

```
$ kubectl  get ns  | grep rdb
```

----End

10 Chart and Add-on

10.1 How Can I Troubleshoot Exceptions That Occur with an Add-on?

If there is an exception during add-on installation, upgrade, or configuration modification, the console will show an error code. By referring to the error code, you can identify the issue and explore the possible causes and solutions. This section provides information on common error codes, their possible causes, and corresponding solutions.

Resource Conflict

Symptom

An internal error occurs during the add-on installation. The error code is CCE.03500001.

Possible Cause

When an internal error occurs, the error message will provide details about the specific cause of the error. For example, if you see the message "ClusterRole \"gatekeeper-manager-role\" in namespace \"\" exists and cannot be imported into the current release," it means that the ClusterRole resource has been created in the cluster, but it is not being managed by the add-on.

Solution

Use kubectl to delete the conflicting resources that are not managed by the add-on and install the add-on again.

Installation Timed Out

Symptom

When an add-on installation or upgrade fails, the system will display a message indicating that the installation has timed out.

```
Release "*****" failed: failed pre-install: timed out waiting for the condition
```


Possible Cause

The add-on pod is not ready.

Solution

On the **Overview** page, view the Kubernetes events to determine the reason why the pod is not ready.

Cause	Solution
The pod cannot be scheduled.	Event: FailedScheduling Cause: The nodes in the cluster cannot accept the pod. The possible causes are as follows: (You can determine the specific cause based on the event details.) • The cluster nodes do not have sufficient CPU and memory resources to meet the requirements of the add-on pod. You can see "Insufficient memory," "Insufficient CPU," or other messages in the events. • The add-on pod is unable to tolerate some taints on the node, and you can see "pod didn't tolerate" or other messages in the events. • There are insufficient nodes to meet the anti-affinity requirements of the add-on pod. You can see "didn't match pod anti-affinity rules" or other messages in the events. Solution: Take the following steps and ensure that the add-on pod's scheduling requirements are met. Once done, install the add-on again. • Check the node taints and delete unnecessary taints. For details, see **Managing Node Taints**. • Properly allocate container resources. For details, see **Properly Allocating Container Computing Resources**. • Add more nodes to the cluster. For details, see **Creating a Node Pool**.
The pod cannot be created.	Rectify the creation failure by referring to **How Can I Locate the Root Cause If a Workload Is Abnormal?**

Add-on Resources Not Exist

Symptom

When an add-on is updated or upgraded, a message is displayed indicating that update has failed and an error 404 is reported.

```
update release failed: update release failed {"error":{"message"."Get release failed: get v3 release by cluster failed, error: release: not found","code":"SVCSTG.CCECAM.4040204"}}, 404
```

Possible Cause

The related resources have been changed or deleted, so an exception occurred while getting the add-on related resources. As a result, the add-on cannot be updated or upgraded directly.

Solution

Uninstall the add-on and install the latest version of it.

10.2 What Should I Do If the NGINX Ingress Controller Add-on Fails to Be Installed in a Cluster and Remains in the Creating State?

Context

You have purchased and set up a CCE cluster and want to access the deployed applications from public networks. Currently, the most efficient way is to register the Service paths of an application on the ingress to allow public network access.

However, after the NGINX Ingress Controller add-on is installed, the add-on is always in the **Creating** state, and the **nginx-ingress-controller** pod is always in the **Pending** state.

Solution

Remove the resource limits assigned to the add-on. The memory that can be used for the NGINX Ingress Controller add-on is limited, so the add-on cannot be started.

Scene Simulation

Step 1 Create a cluster with three nodes, 2 vCPUs and 4 GiB of memory for each node.

Step 2 Install the NGINX Ingress Controller add-on and select 2 vCPUs and 2 GiB of memory.

Step 3 The nginx-ingress deployment is installed, but the nginx-ingress-controller fails to be installed.

Figure 10-1 Add-on always in the Creating state

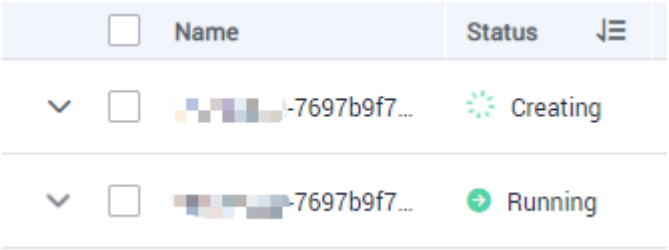
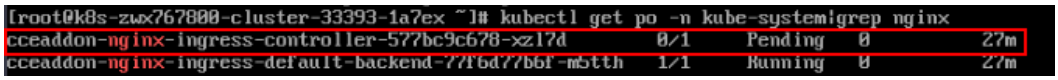


Figure 10-2 Add-on installation failed



Step 4 Check the error message. The following information indicates that resources are insufficient.

```
status:
  phase: Pending
  conditions:
    - type: PodScheduled
      status: 'False'
      lastProbeTime: '2020-02-13T01:20:57Z'
      lastTransitionTime: '2020-02-12T08:50:21Z'
      reason: Unschedulable
      message: '0/3 nodes are available: 3 Insufficient cpu, 3 Insufficient memory.'
  qosClass: Guaranteed
```

Step 5 Add a node with 4 vCPUs and 8 GiB of memory. After that, the NGINX Ingress Controller add-on can be installed.

----End

Possible Cause

Each node in the cluster is configured with 2 vCPUs and 4 GiB of memory initially. System processes such as kubelet, kube-proxy, and Docker consume a portion of system resources. This reduces the available memory on each node to below 2000 MiB. This amount is insufficient for the NGINX Ingress Controller add-on, preventing its installation.

Suggested Solution

Purchase a node with at least 4 vCPUs and 8 GiB of memory.

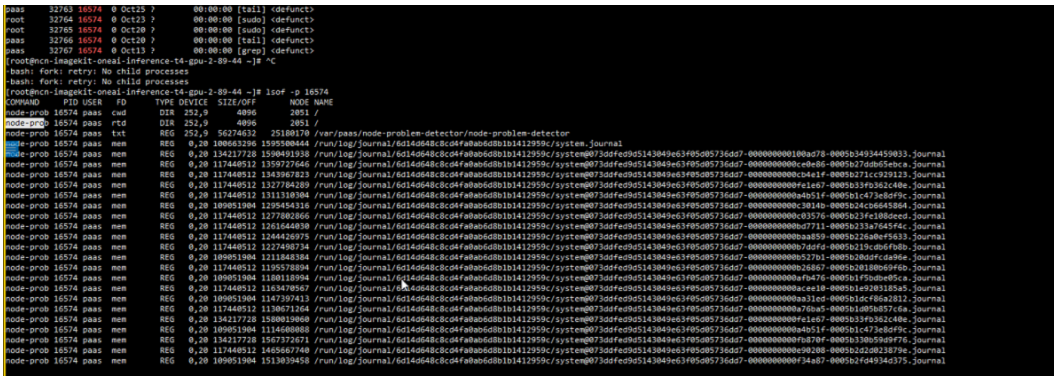
10.3 What Should I Do If Residual Process Resources Exist Due to an Earlier CCE Node Problem Detector Add-on Version?

Description

When the node load is heavy, residual CCE Node Problem Detector process resources may exist.

Symptom

After successful login to the ECS node where the CCE cluster runs, it is found that there are a large number of CCE Node Problem Detector processes exist.



Solution

Upgrade the CCE Node Problem Detector add-on to the latest version.

Step 1 Log in to the CCE console and click the cluster name to access the cluster console. In the navigation pane, choose Add-ons, locate the CCE Node Problem Detector add-on, and click **Upgrade**.

NOTE

If the CCE Node Problem Detector add-on version is 1.13.6 or later, you do not need to upgrade it.

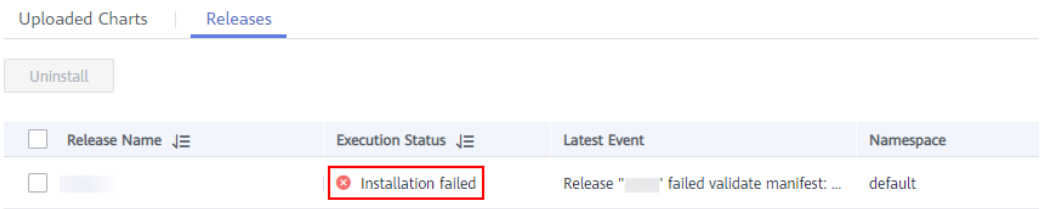
Step 2 In the window that slides out from the right, configure the parameters and click **OK**.

----End

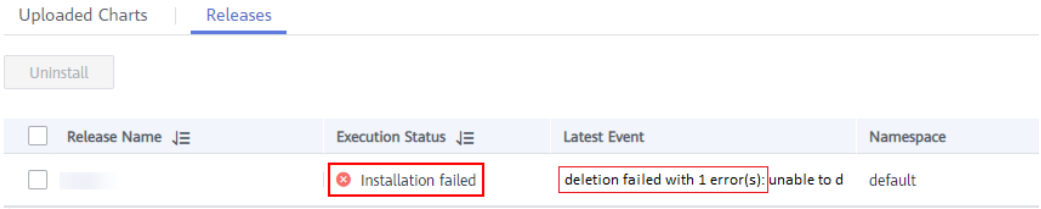
10.4 What Should I Do If a Chart Release Cannot Be Deleted Because the Chart Format Is Incorrect?

Symptom

If an uploaded chart contains incorrect or incompatible resources, the chart will fail to be installed.



In this case, the chart release cannot work properly. You may not be able to delete the release, the error message "deletion failed" is displayed, and the release is still on the GUI.



Solution

In this case, you can run the kubectl commands to delete the release.

NOTE

This problem cannot be solved by deleting the residual chart release. To prevent this problem from occurring again, update the API version of the resources in the chart so that the API version of the resources matches the Kubernetes version.

During chart installation, some resources specified in the chart may have been successfully created. You need to manually delete these resources first. After the residual resources are deleted, you need to delete the chart instance.

For a Helm v2 chart release, query the ConfigMap corresponding to the chart release in the kube-system namespace. For example:

```
[paas@192-168-0-40 ~]$ kubectl -n kube-system get cm
NAME                                 DATA   AGE
9a37566a.cce.io                      0      25d
aosredis.v1                          1      55s
cceaddon-coredns.v1                  1      25d
cceaddon-everest.v1                  1      17d
cceaddon-metrics-server.v1           1      25d
cceaddon-npd-custom-config           0      25d
cceaddon-npd.v1                      1      25d
cceaddon-prometheus.v1               1      25h
cluster-autoscaler-status            1      8d
cluster-versions                     1      25d
coredns                              1      25d
extension-apiserver-authentication   6      25d
ingress-controller-leader-nginx      0      22d
[paas@192-168-0-40 ~]$
```

After the ConfigMap is deleted, the chart release is deleted successfully.

```
[paas@192-168-0-40 ~]$ kubectl -n kube-system delete cm aosredis.v1
configmap "aosredis.v1" deleted
[paas@192-168-0-40 ~]$
```

For a Helm v3 chart release, query the Secret corresponding to the chart release in the namespace. For example:

```
[root@cce-1717-vpc-node2 ~]# kubectl -n default get secret
NAME                            TYPE                                  DATA   AGE
default-secret                  kubernetes.io/dockerconfigjson        1      21h
default-token-978pv             kubernetes.io/service-account-token   3      21h
paas.elb                        cfe/secure-opaque                     3      21h
sh.helm.release.v1.test-nginx.v1   helm.sh/release.v1                 1      139m
[root@cce-1717-vpc-node2 ~]#
```

After the Secret is deleted, the chart release is deleted successfully.

```
[root@cce-1717-vpc-node2 ~]# kubectl -n default delete secret sh.helm.release.v1.test-nginx.v1
secret "sh.helm.release.v1.test-nginx.v1" deleted
[root@cce-1717-vpc-node2 ~]#
```


Note: If you perform operations on the console, CCE automatically bumps the original v2 chart release to v3 when you obtain or update the chart release. The release information is stored in the Secret. The release information in the original ConfigMap is not deleted. You are advised to query and delete the chart release in both the ConfigMap and Secret.

10.5 Does CCE Support nginx-ingress?

Introduction to nginx-ingress

nginx-ingress is a popular ingress-controller. It functions as a reverse proxy to import external traffic to a cluster and expose Services in Kubernetes clusters to external systems. In layer-7 load balancing (ingress), domain names are used to match Services. In this way, Services in a cluster can be accessed through domain names.

This chart is composed of ingress-controller and nginx.

- ingress-controller monitors Kubernetes ingresses and updates nginx configurations.

  NOTE

  For details about ingresses, see **https://kubernetes.io/docs/concepts/services-networking/ingress/**.

- nginx implements load balancing for requests and supports layer-7 request forwarding.

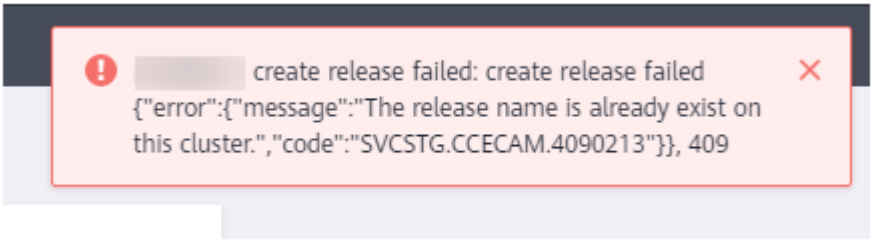
Installation Method

You can install the nginx-ingress add-on on the **Add-ons** page on the CCE console and configure its parameters.

10.6 What Should I Do If Installation of an Add-on Fails and "The release name is already exist" Is Displayed?

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 the 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 -A |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 -A | 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 the add-on and then uninstall it. Ensure that the residual add-on resources are cleared. After the uninstallation is complete, install the add-on again.

NOTE

During the initial installation of the add-on, it is possible to encounter abnormal behavior caused by residual resources from a previous add-on release. This is a normal occurrence. In such cases, you can resolve the issue by uninstalling the add-on from the console. This will ensure that any remaining resources are cleared, allowing for a proper installation of the add-on again.

----End

10.7 What Should I Do If a Chart Creation or Upgrade Fails and "rendered manifests contain a resource that already exists" Is Displayed?

Symptom

When a chart cannot be created or upgraded, the error message "Create release by helm failed: rendered manifests contain a resource that already exists. Unable to continue with install: ..." is displayed.

CCE add-ons are installed using charts. The same error may occur when an add-on is installed or upgraded.

Possible Cause

This error message indicates that there are some residual resources related to the chart or add-on in the cluster. The possible causes are as follows:

- Chart resources (such as releases) were mistakenly or abnormally deleted in the backend.
- The namespace installed using the chart was directly deleted, leaving residual resources.
- A resource with the same name as a chart component exists in the cluster but lacks the **app.kubernetes.io/managed-by: Helm** label.

Solution

Access the cluster using kubectl, manually delete the resources associated with the error, and reinstall the chart or add-on.

Step 1 Check the error message to identify the resource causing the conflict. Pay attention to the information after "Unable to continue with install:".

For example:

```
create release failed: create release failed {"error":{"message":"Create release by helm failed:rendered manifests contain a resource that already exists. Unable to continue with install: ClusterRole \"cceaddon-nginx-ingress\" in namespace \"\" exists and cannot be imported into the current release: invalid ownership metadata; annotation validation error: key \"meta.helm.sh/release-namespace\" must equal \"kube-system\": current value is \"example\"","code":"SVCSTG.CCECAM.5000208"}}
```

- **Conflicting resource**: ClusterRole named **cceaddon-nginx-ingress**, which does not belong to any namespace (ClusterRoles are not namespace-level resources.)
- **Conflicting field**: **metadata.annotations.meta.helm.sh/release-namespace**
  - **Expected value**: **kube-system**
  - **Actual value**: **example**

Step 2 Run the following kubectl command to delete the conflicting resource in the cluster: (The command here is only an example. Delete the resources based on the error message.)

```
kubectl delete clusterRole cceaddon-nginx-ingress
```

Step 3 After the resource conflict is resolved, reinstall the chart or add-on. If the conflict message reappears, repeat the process.

----End

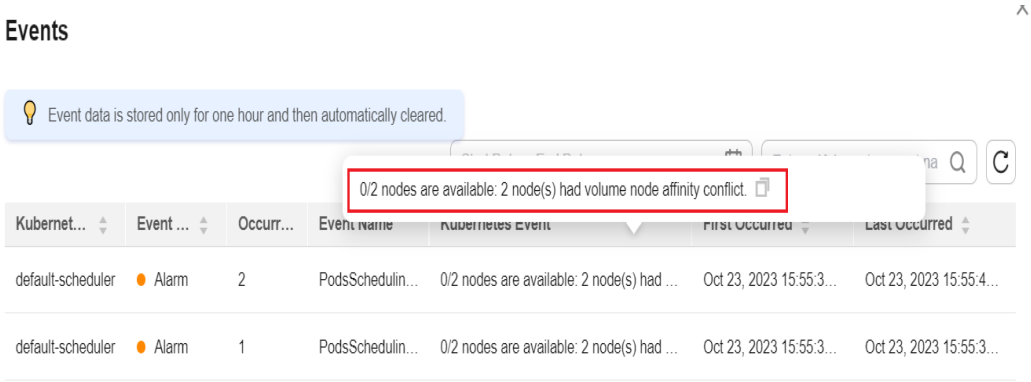
10.8 What Can I Do If the kube-prometheus-stack Add-on Instance Fails to Be Scheduled?

Symptom

During the installation of kube-prometheus-stack, the add-on remains in the partially ready state. The message "0/x nodes are available: x node(s) had volume node affinity conflict." is displayed in the event of the prometheus pod.

The same problem may occur during the installation of grafana.

Figure 10-3 Failed to schedule the prometheus pod



Possible Cause

The PV required by the prometheus pod already exists in the cluster, but the corresponding EVS disk is not in the same AZ as the node where the prometheus pod resides. As a result, the pod scheduling fails. This may be because kube-prometheus-stack is not installed for the first time in the cluster.

- If kube-prometheus-stack is installed for the first time, the attaching of an EVS disk (with the PVC named **pvc-prometheus-server-0**) to the prometheus pod will be delayed. When the EVS disk is created, it will automatically be in the same AZ as the node where the prometheus pod resides. For example, if the AZ of the node where the pod is running is AZ 1, the disk will be automatically created in AZ 1.
- When kube-prometheus-stack is uninstalled from the cluster, the PV mounted to the prometheus pod will not be deleted and the existing monitoring data will be retained. If the add-on is installed again, the nodes in the cluster may be newly created. If none of them is in AZ 1, the prometheus pod cannot run.

The causes may also result in the scheduling failure of a grafana pod.

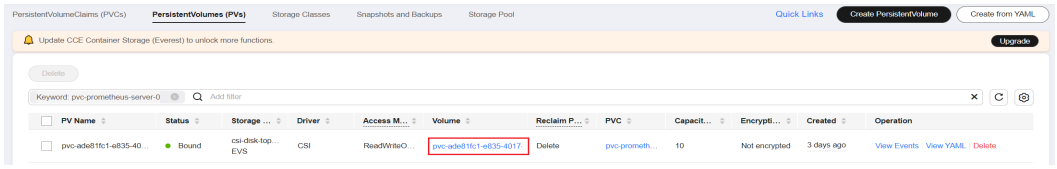
Solution

Check the AZ of the EVS disk corresponding to the existing PV mounted to the pod and create a node in the same AZ as this EVS disk.

Step 1 Log in to the CCE console and click the cluster name to access the cluster console.

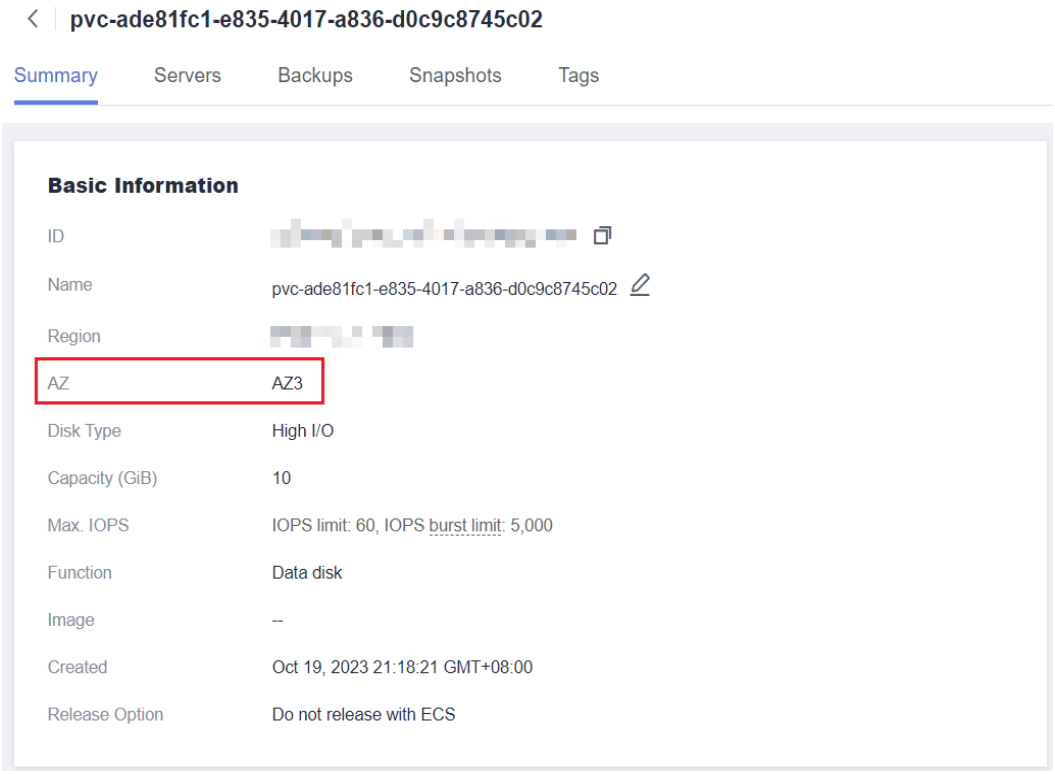
Step 2 In the navigation pane, choose **Storage**. Click the **PVs** tab, locate the row that contains the **pvc-prometheus-server-0** PVC in the **PVC** column, and click the volume name in the **Volume** column to go to the EVS disk details page.

Figure 10-4 Locating the target volume



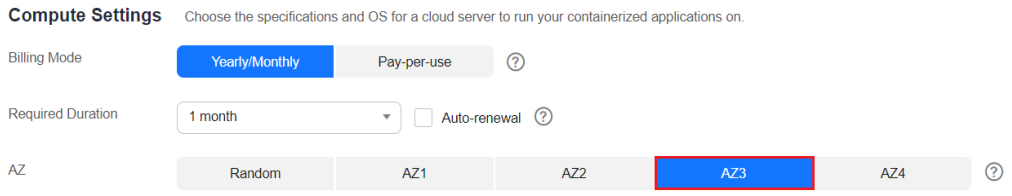
Step 3 In the **Basic Information** area, view the AZ of the EVS disk.

Figure 10-5 Viewing the details of the EVS disk



Step 4 On the CCE console, click the cluster name to access the cluster console. Choose **Nodes** in the navigation pane, click the **Nodes** tab, and click **Create Node** to create a node in the same AZ as the EVS disk.

Figure 10-6 Creating a node in a specified AZ



Step 5 Reschedule the node. This operation is automatically performed by the workload scheduler.

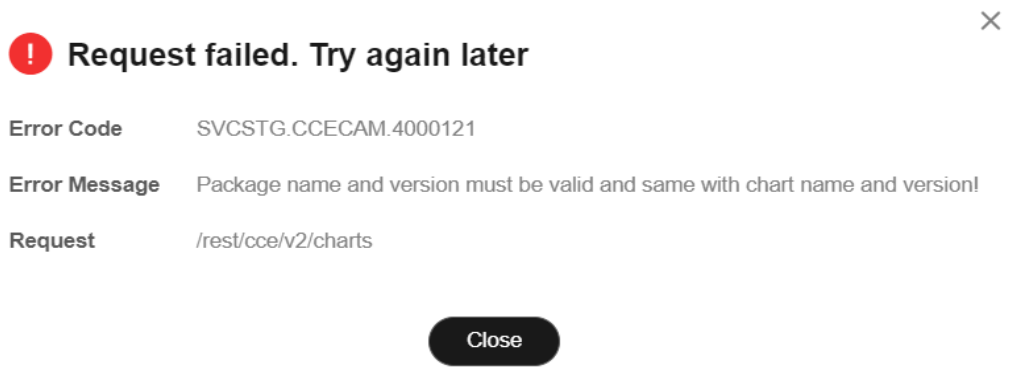
----End

10.9 What Can I Do If a Chart Fails to Be Uploaded?

Symptom

When a chart is uploaded, a request error "Request failed. Try again later" is displayed. The error code is **SVCSTG.CCECAM.4000121**, and the error message is "Package name and version must be valid and same with chart name and version!"

Figure 10-7 Chart upload failure



Possible Cause

If the preceding error information is displayed, the values of the **name** and **version** fields in the **Chart.yaml** file are inconsistent with those in the chart package.

NOTE

To customize the name and version of a chart package, modify the **name** and **version** fields in the **Chart.yaml** file.

Solution

Step 1 Check the **name** and **version** fields in the **Chart.yaml** file.

The following shows an example **Chart.yaml** file, where the chart name is **newer-nginx-ingress** and the version is **4.4.2**:

```
annotations:
  artifacthub.io/changes: |
    - Adding support for disabling liveness and readiness probes to the Helm chart
    - add:(admission-webhooks) ability to set securityContext
    - Updated Helm chart to use the fullname for the electionID if not specified
    - Rename controller-wehbooks-networkpolicy.yaml
  artifacthub.io/prerelease: "false"
apiVersion: v2
appVersion: 1.5.1
description: Ingress controller for Kubernetes using NGINX as a reverse proxy and
  load balancer
home: https://github.com/kubernetes/ingress-nginx
icon: https://upload.wikimedia.org/wikipedia/commons/thumb/c/c5/Nginx_logo.svg/500px-
Nginx_logo.svg.png
keywords:
- ingress
- nginx
kubeVersion: '>=1.20.0-0'
maintainers:
- name: rikatz
- name: strongjz
- name: tao12345666333
name: newer-nginx-ingress
sources:
- https://github.com/kubernetes/ingress-nginx
version: 4.4.2
```

Step 2 Change the name of the chart package based on the **Chart.yaml** file. The chart package is named in the format of *{Name}*-*{Version}*.**tgz**, for example, **newer-**

nginx-ingress-4.4.2.tgz. *{Version}* indicates the version number. It is named in the format of *{Major version number}.{Minor version number}.{Revision number}*.

Step 3 Upload the chart again.

----End

10.10 How Do I Configure the Add-on Resource Quotas Based on Cluster Scale?

After changing the cluster scale, adjust the add-on resource quotas based on the cluster scale to ensure that the add-on pods can run properly. For example, if you expand the cluster scale from 50 worker nodes to 200 worker nodes or more, increase the CPU and memory quotas of the add-on pods to avoid exceptions such as OOM caused by too many nodes required for scheduling the add-on pods.

Configuring Resource Quotas for coredns

Queries per Second (QPS) of the coredns add-on is positively correlated with the CPU consumption. If the number of nodes or containers in the cluster grows, the coredns pod will bear heavier workloads. Adjust the number of add-on pods and their CPU and memory quotas based on the cluster scale.

Table 10-1 Recommended values for coredns

Nodes	Recommended Configuration (QPS)	Pods	CPU Request (m)	CPU Limit (m)	Memory Request (MiB)	Memory Limit (MiB)
50	2500	2	500	500	512	512
200	5000	2	1000	1000	1024	1024
1000	10000	2	2000	2000	2048	2048
2000	20000	4	2000	2000	2048	2048

Configuring Resource Quotas for everest

After the cluster scale is adjusted, the everest specifications need to be modified based on the cluster scale and the number of PVCs. The requested CPU and memory can be increased based on the number of nodes and PVCs. For details, see **Table 10-2**.

In non-typical scenarios, the formulas for estimating the limit values are as follows:

- everest-csi-controller
  - CPU limit: 250m for 200 or fewer nodes, 350m for 1000 nodes, and 500m for 2000 nodes

  - Memory limit = (200 MiB + Number of nodes x 1 MiB + Number of PVCs x 0.2 MiB) x 1.2
- everest-csi-driver
  - CPU limit: 300m for 200 or fewer nodes, 500m for 1000 nodes, and 800m for 2000 nodes
  - Memory limit: 300 MiB for 200 or fewer nodes, 600 MiB for 1000 nodes, and 900 MiB for 2000 nodes

Table 10-2 Recommended configuration limits in typical scenarios

Configuration Scenario			everest-csi-controller		everest-csi-driver	
Nodes	PVs/PVCs	Add-on Pods	CPU Cores (Limit = Request)	Memory (Limit = Request)	CPU Cores (Limit = Request)	Memory (Limit = Request)
50	1000	2	250m	600 MiB	300m	300 MiB
200	1000	2	250m	1 GiB	300m	300 MiB
1000	1000	2	350m	2 GiB	500m	600 MiB
1000	5000	2	450m	3 GiB	500m	600 MiB
2000	5000	2	550m	4 GiB	800m	900 MiB
2000	10000	2	650m	5 GiB	800m	900 MiB

Configuring Resource Quotas for autoscaler

autoscaler automatically adjusts the number of nodes in a cluster based on workloads. Adjust the number of add-on pods and their CPU and memory quotas based on the cluster scale.

Table 10-3 Recommended values for autoscaler

Node	Pod	CPU Request (m)	CPU Limit (m)	Memory Request (MiB)	Memory Limit (MiB)
50	2	1000	1000	1000	1000
200	2	4000	4000	2000	2000
1000	2	8000	8000	8000	8000
2000	2	8000	8000	8000	8000

Configuring Resource Quotas for volcano

After the cluster scale is increased, the resource quotas required by volcano need to be modified based on the cluster scale.

- If the number of nodes is less than 100, retain the default configuration. The requested CPU is 500m, and the limit is 2000m. The requested memory is 500 MiB, and the limit is 2000 MiB.
- If the number of nodes is greater than 100, increase the requested CPU by 500m and the requested memory by 1000 MiB each time 100 nodes (10,000 pods) are added. Increase the CPU limit by 1500m and the memory limit by 1000 MiB.

  NOTE

  Formulas for calculating the requests:

  - CPU request: Calculate the number of nodes multiplied by the number of pods, perform interpolation search using the product of the number of nodes in the cluster multiplied by the number of pods in **Table 10-4**, and round up the request and limit which are closest to the specifications.

    For example, for 2000 nodes (20,000 pods), the product of the number of nodes multiplied by the number of pods is 40 million, which is close to 700/70,000 in the specification (Number of nodes x Number of pods = 49 million). Set the CPU request to 4000m and the limit to 5500m.
  - Memory request: Allocate 2.4 GiB of memory to every 1000 nodes and 1 GiB of memory to every 10,000 pods. The memory request is the sum of the two values. (The obtained value may be different from the recommended value in **Table 10-4**. You can use either of them.)

    Memory request = Number of nodes/1000 x 2.4 GiB + Number of pods/10000 x 1 GiB

    For example, for 2000 nodes and 20,000 pods, the memory request value is 6.8 GiB (2000/1000 x 2.4 GiB + 20000/10000 x 1 GiB).

Table 10-4 Recommended requested resources and resource limits for volcano-controller and volcano-scheduler

Nodes/Pods in a Cluster	CPU Request (m)	CPU Limit (m)	Memory Request (MiB)	Memory Limit (MiB)
50/5000	500	2000	500	2000
100/10000	1000	2500	1500	2500
200/20000	1500	3000	2500	3500
300/30000	2000	3500	3500	4500
400/40000	2500	4000	4500	5500
500/50000	3000	4500	5500	6500
600/60000	3500	5000	6500	7500
700/70000	4000	5500	7500	8500

Configuring Resource Quotas for Other Add-ons

Resource quotas of other add-ons may also be insufficient due to cluster scale expansion. If, for example, the CPU or memory usage of the add-on pods increases and even OOM occurs, modify the resource quotas as required.

For example, the resources occupied by the Cloud Native Cluster Monitoring add-on are related to the number of pods in the cluster. If the cluster scale is expanded, the number of pods may also grow. In this case, increase the resource quotas of the add-on pods.

10.11 How Can I Clean Up Residual Resources After the NGINX Ingress Controller Add-on in the Unknown State Is Deleted?

Symptom

The NGINX Ingress Controller add-on is in the unknown state, and after this add-on is uninstalled, residual components still remain.

Involved Kubernetes resources include:

- Namespace-level resources: secret, ConfigMap, Deployment, Service, Role, RoleBinding, lease, ServiceAccount, and job
- Cluster-level resources: ClusterRole, ClusterRoleBinding, IngressClass, and ValidatingWebhookConfiguration

Solution

Step 1 Use kubectl to access the cluster.

Step 2 Search for related resources.

```
className="nginx"
namespace="kube-system"
className=`if [[ ${className} == "nginx" ]]; then echo ""; else echo "-${className}";fi`
kubectl get -n ${namespace} secret sh.helm.release.v1.cceaddon-nginx-ingress${className}.v1 cceaddon-nginx-ingress${className}-admission
kubectl get -n ${namespace} cm cceaddon-nginx-ingress${className}-controller
kubectl get -n ${namespace} deploy cceaddon-nginx-ingress${className}-controller cceaddon-nginx-ingress${className}-default-backend
kubectl get -n ${namespace} svc cceaddon-nginx-ingress${className}-controller-admission cceaddon-nginx-ingress${className}-default-backend cceaddon-nginx-ingress${className}-controller
kubectl get -n ${namespace} role cceaddon-nginx-ingress${className}
kubectl get -n ${namespace} rolebinding cceaddon-nginx-ingress${className}
kubectl get -n ${namespace} lease ingress-controller-leader${className}
kubectl get -n ${namespace} serviceAccount cceaddon-nginx-ingress${className}
kubectl get clusterRole cceaddon-nginx-ingress${className}
kubectl get clusterRoleBinding cceaddon-nginx-ingress${className}
kubectl get ingressClass ${className}
kubectl get ValidatingWebhookConfiguration cceaddon-nginx-ingress${className}-admission
```

className specifies the name of a controller. **namespace** specifies the namespace where NGINX Ingress Controller was installed.

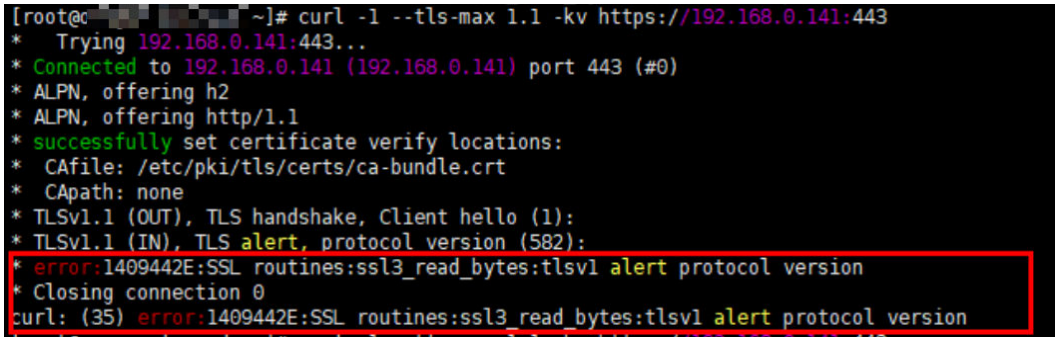
Step 3 Manually delete the residual resources if the preceding resources are present.

----End

10.12 Why Can't TLS v1.0 or v1.1 Be Used After the NGINX Ingress Controller Add-on Is Upgraded?

Symptom

After the NGINX Ingress Controller add-on is upgraded to 2.3.3 or later, if the TLS version of the client is earlier than v1.2, an error is reported during the negotiation between the client and NGINX Ingress Controller.



Solution

NGINX Ingress Controller 2.3.3 and later versions support only TLS v1.2 and v1.3 by default. If additional TLS versions are needed, you can add the **@SECLEVEL=0** field to the **ssl-ciphers** parameter configured for the NGINX Ingress Controller add-on. For details, see **TLS/HTTPS**.

CAUTION

For secure data transmission, it is advised to avoid using TLS v1.0 or v1.1. These outdated protocols pose security risks that could lead to a data leak or be exploited by attackers. You are advised to upgrade TLS to v1.2 or later versions for enhanced communication security.

Step 1 Log in to the CCE console and click the cluster name to access the cluster console. In the navigation pane, choose **Add-ons**, locate the NGINX Ingress Controller add-on, and click **Manage**.

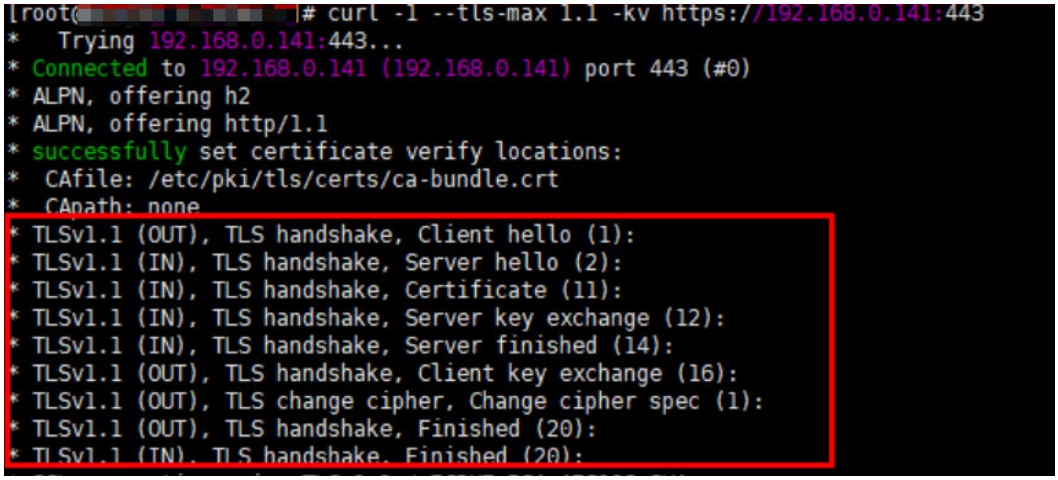
Step 2 In the right corner of the installed add-on list, click **Edit**.

Step 3 Add the following configuration to the **Nginx Parameters**:

```
{
  "ssl-ciphers": "@SECLEVEL=0 ECDHE-ECDSA-AES128-GCM-SHA256:ECDHE-RSA-AES128-GCM-SHA256:ECDHE-ECDSA-AES256-GCM-SHA384:ECDHE-RSA-AES256-GCM-SHA384:ECDHE-ECDSA-CHACHA20-POLY1305:ECDHE-RSA-CHACHA20-POLY1305:DHE-RSA-AES128-GCM-SHA256:DHE-RSA-AES256-GCM-SHA384:DHE-RSA-CHACHA20-POLY1305:ECDHE-ECDSA-AES128-SHA256:ECDHE-RSA-AES128-SHA256:ECDHE-ECDSA-AES128-SHA:ECDHE-RSA-AES128-SHA:ECDHE-ECDSA-AES256-SHA384:ECDHE-RSA-AES256-SHA384:ECDHE-ECDSA-AES256-SHA:ECDHE-RSA-AES256-SHA:DHE-RSA-AES128-SHA256:DHE-RSA-AES256-SHA256:AES128-GCM-SHA256:AES256-GCM-SHA384:AES128-SHA256:AES256-SHA256:AES128-SHA:AES256-SHA:DES-CBC3-SHA",
  "ssl-protocols": "TLSv1 TLSv1.1 TLSv1.2 TLSv1.3"
}
```


Step 4 Click **OK**.

Step 5 Use TLS v1.1 for access again and verify that the response is normal.



----End

10.13 What Can I Do If a Pod Cannot Be Started After the CCE AI Suite (Ascend NPU) Add-on Is Upgraded from 1.x.x to 2.x.x?

Symptom

After upgrading the CCE AI Suite (Ascend NPU) add-on from 1.x.x to 2.x.x but deploying services using the original configuration, a service pod remains in the pending state and cannot start.

Running the **kubectl describe pod** command reveals an error message, which indicates that the container creation has failed.

```
Error response from daemon: Duplicate mount point: /usr/local/bin/npu-smi
```

```
started: false
state:
  waiting:
    message: 'Error response from daemon: Duplicate mount point: /usr/local/bin/npu-smi'
    reason: CreateContainerError
ostIP: 10.94.11.187
hase: Pending
odIP: 10.94.17.129
```

Possible Cause

In version 1.x.x, the add-on cannot automatically mount drivers and npu-smi to a service pod. This must be done manually. In version 2.x.x, the add-on can automatically mount drivers and npu-smi to service pods. This leads to a conflict between the npu-smi directories automatically mounted to a service pod and those manually mounted before the upgrade. As a result, the *Duplicate mount point: /usr/local/bin/npu-smi* error occurs.

CCE AI Suite (Ascend NPU) 2.x.x automatically mounts the following directories to service pods:

```
/usr/local/Ascend/driver/lib64/common
/usr/local/Ascend/driver/lib64/driver
/usr/local/bin/npu-smi
```

The way CCE AI Suite (Ascend NPU) mounts drivers and npu-smi to service pods differs depending on its version. The following table shows the details.

Type		CCE AI Suite (Ascend NPU) 1.x.x	CCE AI Suite (Ascend NPU) 2.0.0 to 2.1.6	CCE AI Suite (Ascend NPU) 2.1.7 to the Latest Version
310 series card	Driver version < 23.0.rc0	You must manually mount the drivers and npu-smi to a service pod.	You must manually mount the drivers and npu-smi to a service pod.	You must manually mount the drivers and npu-smi to a service pod.
	Driver version ≥ 23.0.rc0	You must manually mount the drivers and npu-smi to a service pod.	The add-on can automatically mount the drivers to a service pod, but it cannot mount the npu-smi.	**The add-on can automatically mount the drivers and npu-smi to a service pod.**

Solution

When using a CCE AI Suite (Ascend NPU) of version 2.x.x, remove the fields for mounting drivers and npu-smi directories from the YAML file of a service pod. The add-on will automatically handle the mounting of drivers and npu-smi directories.

To manually mount them to a service pod, change the mount path **/usr/local/bin/npu-smi** to **/usr/local/sbin/npu-smi** to resolve the path conflict.

10.14 How Can I Drain a GPU Node After Upgrading or Rolling Back the CCE AI Suite (NVIDIA GPU) Add-on?

Symptom

When GPU virtualization workloads are present on a GPU node, upgrading or rolling back the CCE AI Suite (NVIDIA GPU) add-on led to failures in upgrading or rolling back certain components like GPU virtualization and runtime components. To ensure smooth operation of the add-on, a drainage operation must be carried out on the GPU node to clear the GPU virtualization workloads. You are advised to follow the rolling drainage policy, which involves draining only one or a few GPU nodes at a time to avoid disrupting services on a large scale.

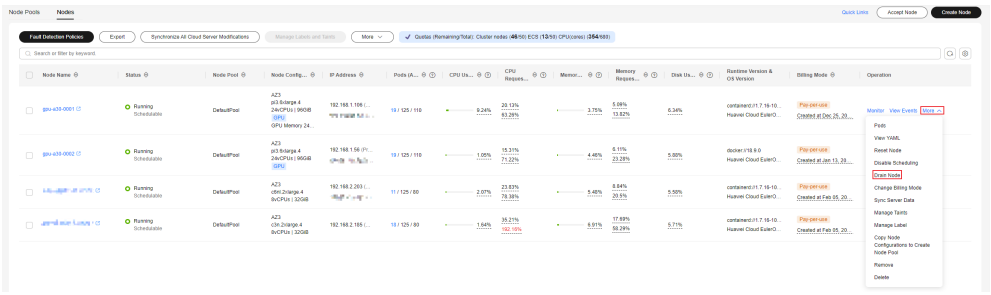
Solution

When draining a GPU node, make sure to reserve enough GPU resources on other nodes to meet pod scheduling needs. This helps avoid pod scheduling issues due to inadequate resources and ensures smooth service operation.

Step 1 Log in to the CCE console and click the cluster name to access the cluster **Overview** page.

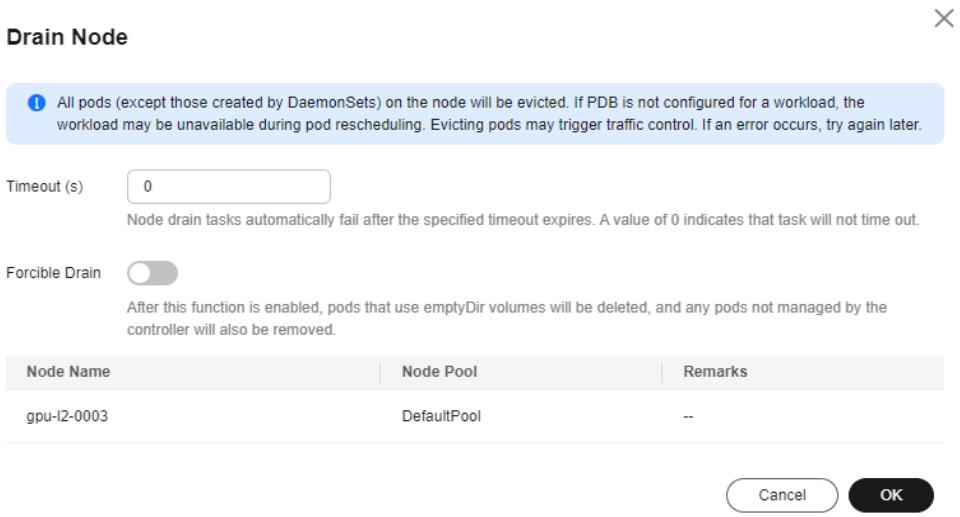
Step 2 In the navigation pane, choose **Cluster** > **Nodes**. In the right pane, click the **Nodes** tab, locate the row containing the target GPU node, and choose **More** > **Drain Node** in the **Operation** column.

Figure 10-8 Draining a node



Step 3 In the **Drain Node** dialog box displayed, click **OK**. If there are pods with emptyDir volumes mounted or pods that are not managed by controllers, enable forcible drainage.

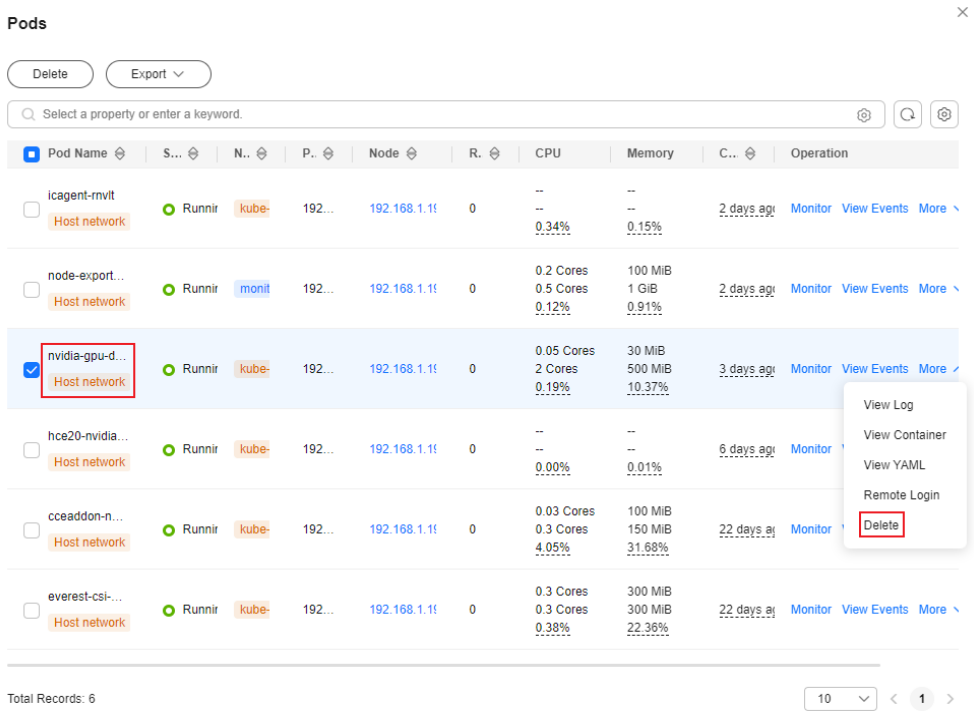
Figure 10-9 Configuring node drainage



Step 4 In the node list, locate the row containing the GPU node and choose **More** > **Pods** in the **Operation** column. In the window that slides out from the right, locate the row containing the **nvidia-gpu-device-plugin-*xxx*** pod and choose **More** > **Delete** in the **Operation** column. In the **Delete Pod** dialog box displayed, click **Yes**.

You can see that the **nvidia-gpu-device-plugin-*xxx*** pod is in the **Abnormal** state. Once it transitions to the **Running** state and **Drained** appears in the **Status** column, the GPU virtualization workloads on the GPU node have been drained.

Figure 10-10 Deleting the nvidia-gpu-device-plugin-xxx pod



Step 5 In the node list, locate the row containing the GPU node and choose **More** > **Enable Scheduling** in the **Operation** column. You can continue repeating the previous steps until all GPU nodes have been drained.

----End

10.15 Why Am I Unable to Install a NVIDIA Driver on EulerOS 2.9?

Symptom

In EulerOS 2.9, manually installing a GPU driver without using the CCE AI Suite (NVIDIA GPU) add-on results in an error message similar to the following displayed:

```
ERROR: Unable to find the kernel source tree for the currently running kernel. Please make sure you have installed the kernel source files for your kernel and that they are properly configured; on Red Hat Linux systems, for example, be sure you have the 'kernel-source' or 'kernel-devel' RPM installed. If you know the correct kernel source files are installed, you may specify the kernel source path with the '--kernel-source-path' command line option.
...
```


Fault Locating

If an error occurs during the GPU driver installation, you can check the driver logs and determine the error cause by running the following command:

```
cat /var/log/nvidia-installer.log
```

If the command output contains the following information, the GPU driver dependencies are not installed:

```
...
Cannot generate ORC metadata for CONFIG_UNWINDER_ORC=y, please install libelf-dev, libelf-devel or elfutils-libelf-devel
...
```

Solution

You can solve this problem using either of the following methods:

- Reinstall the GPU driver using the CCE AI Suite (NVIDIA GPU) add-on. For details, see **Installing the Add-on**.
- Manually install the missing dependencies and then reinstall the GPU driver. The procedure is as follows:

  a. Install the dependencies that are missing in **Fault Locating**:
  ```
  yum install -y gcc gcc-c++ perl make elfutils-libelf-devel libX11 libXext binutils
  ```
  If information similar to the following is displayed, the dependencies have been installed.
  ```
  ...
  Complete!
  ```

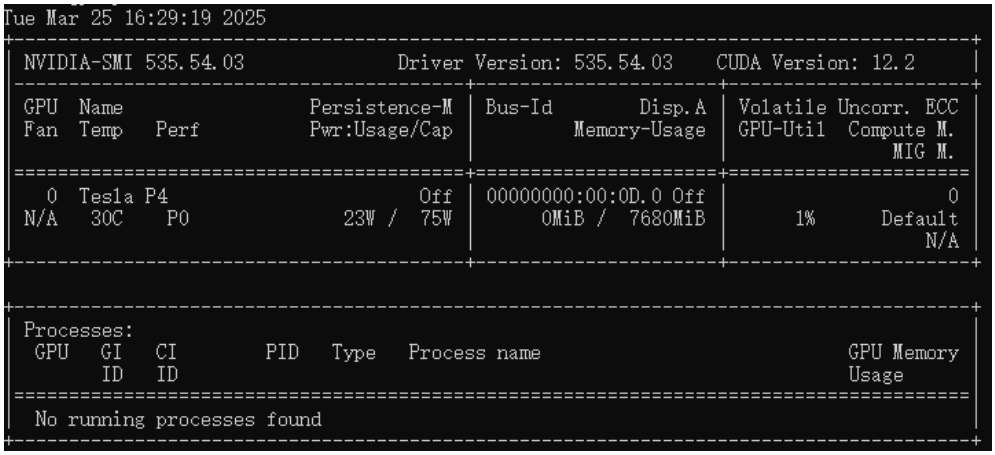
  b. Reinstall the driver:
  ```
  ./NVIDIA-Linux-x86_64-535.54.03.run --silent --accept-license
  ```

  - If the command output does not contain **ERROR**, the driver has been installed. For example:
  ```
  Verifying archive integrity... OK
  ...
  WARNING: This NVIDIA driver package includes Vulkan components, but no Vulkan ICD loader was detected on this system. The NVIDIA Vulkan ICD will not function without the loader. Most distributions package the Vulkan loader; try instal1ling the "vulkan-loader", "vulkan-icd-1oaderor,""libvulkanl" package.
  ```

  - If information similar to the following is displayed in the command output, add **--no-drm** to the installation command:
  ```
  ERROR: Unable to load the kernel module 'nvidia-drm ko'. ...
  ERROR: The nvidia-drm kernel module failed to load. This kernel module is required for the proper operation of DRM-KMS. If you do not need to use DRM-KMS, you can try to install this driver package again with the '--no-drm' option.
  ...
  ```
  Reinstall the driver:
  ```
  ./NVIDIA-Linux-x86_64-535.54.03.run --silent --accept-license --no-drm
  ```

  c. Check whether the nvidia-smi driver can be used properly:
  ```
  nvidia-smi
  ```
  If information similar to that in the following figure is displayed, nvidia-smi can be used properly.

Figure 10-11 Command output



10.16 Why Is a VolcanoJob (vcjob) Resource Unable to Function Properly After the Volcano Scheduler Add-on Upgrade?

Symptom

After the Volcano Scheduler add-on is upgraded from 1.4.7 or earlier to a version later than 1.4.7, a newly created VolcanoJob (vcjob) resource cannot run properly. The error information in the API server logs and that reported by the volcano-admission component are as follows:

The error information in the API server logs:

```
...
W0318 14:57:51.376736      13 dispatcher.go:142] Failed calling webhook, failing open validatejob.volcano.sh: failed calling webhook "validatejob.volcano.sh": failed to call webhook: Post "https://volcano-admission-service.kube-system.svc:443/jobs/validate?timeout=30s": EOF
E0318 14:57:51.376768      13 dispatcher.go:149] failed calling webhook "validatejob.volcano.sh": failed to call webhook: Post "https://volcano-admission-service.kube-system.svc:443/jobs/validate?timeout=30s": EOF
...
```

The error information reported by the volcano-admission component:

```
... no kind "AdmissionReview" is registered for version "admission.k8s.io/v1beta1" ...
```

Possible Cause

The **webhooks.admissionReviewVersions** field information in the early version is incompatible with that in the upgraded version.

In earlier versions (1.4.7 or earlier), the value of the **webhooks.admissionReviewVersions** field of the MutatingWebhookConfiguration and ValidatingWebhookConfiguration resource objects is **v1beta1**. In later versions (1.4.7 or later), the value of this field is **v1**. If the HA mode is used during the add-on upgrade, and the number of volcano-admission replicas is increased, the ReplicaSet of the old version will start a **pod** of the old version. If the pod is not destroyed in a timely manner, it will forcibly overwrite the configurations in the MutatingWebhookConfiguration and ValidatingWebhookConfiguration resource objects and reset the value of the **webhooks.admissionReviewVersions** field to

v1beta1. Kubernetes cannot identify the field, so the created vcjob resource cannot run properly.

For details about the MutatingWebhookConfiguration and ValidatingWebhookConfiguration resource objects, see **Table 10-5**.

Table 10-5 Involved resource objects

Resource Type	Resource Name
MutatingWebhookConfiguration	volcano-admission-service-jobs-mutate
	volcano-admission-service-podgroups-mutate
	volcano-admission-service-queues-mutate
	volcano-admission-service-pods-mutate
ValidatingWebhookConfiguration	volcano-admission-service-jobs-validate
	volcano-admission-service-pods-validate
	volcano-admission-service-queues-validate

Solution

Change the value of **webhooks.admissionReviewVersions** of the following resource objects from **v1beta1** to **v1** and then create the vcjob again.

Step 1 Run the **kubectl edit** command to modify the fields in **Table 10-5** one by one. The following uses **volcano-admission-service-jobs-mutate** as an example to describe how to modify the **webhooks.admissionReviewVersions** field.

Modify the YAML file of **volcano-admission-service-jobs-mutate**:

```
kubectl edit MutatingWebhookConfiguration volcano-admission-service-jobs-mutate
```

In the YAML file, press **i** to edit the file content and change the value of **webhooks.admissionReviewVersions** from **v1beta1** to **v1**.

```
apiVersion: admissionregistration.k8s.io/v1
kind: MutatingWebhookConfiguration
metadata:
  annotations:
    meta.helm.sh/release-name: cceaddon-volcano
    meta.helm.sh/release-namespace: kube-system
  creationTimestamp: "2025-02-25T08:11:52Z"
  generation: 2
  labels:
    app.kubernetes.io/managed-by: Helm
    release: cceaddon-volcano
  name: volcano-admission-service-jobs-mutate
  resourceVersion: "252406"
  uid: 7e9bdaaf-1b6c-4975-a171-ada8456c12e5
webhooks:
- admissionReviewVersions:
  - v1beta1   # Change the value to v1.
 ...
```

After the modification is complete, press **Esc** to exit the editing and enter **:wq** to save the modification.

Step 2 After the **webhooks.admissionReviewVersions** fields of the objects in **Table 10-5** are modified, delete the vcjob resource that fails to run properly.

```
kubectl delete vcjob -n namespace vcjob_name
```

Information similar to the following is displayed:

```
vcjob vcjob_name deleted
```

Step 3 Recreate the vcjob resource:

```
kubectl create -f vcjob.yaml   # Replace vcjob.yaml with the YAML file for creating the vcjob resource.
```

Information similar to the following is displayed:

```
job.batch.volcano.sh/vcjob_name created
```

Step 4 Check whether the vcjob resource has been created:

```
kubectl get vcjob_name -n namespace
```

If the value of **STATUS** is **Running**, the vcjob resource has been created.

```
NAME         STATUS    MINAVAILABLE   RUNNINGS   AGE
vcjob_name   Running   1                         2m30s
```

----End

Workaround

Before upgrading the add-on, you can take the following measures to avoid this problem:

- To upgrade the Volcano Scheduler add-on from version 1.4.7 or earlier to any version up to and including 1.13.7, go to the **Upgrade Add-on** page. Set **Add-on Specifications** to **Preset** and choose **Standalone**, or set **Add-on Specifications** to **Custom** and configure the number of pods to 1. After making these changes, proceed with upgrading the add-on. After the add-on is upgraded, find the Volcano Scheduler add-on on the **Add-ons** page and click **Edit**. In the window that slides out from the right, change the add-on specifications as required.
- To upgrade the Volcano Scheduler add-on from version 1.4.7 or earlier to a version later than 1.13.7, go to the **Upgrade Add-on** page. Set **Add-on Specifications** to **Custom**, set the number of replicas of the **volcano-admission** component to 1, and proceed with upgrading the add-on. After the add-on is upgraded, find the Volcano Scheduler add-on on the **Add-ons** page and click **Edit**. In the window that slides out from the right, change the add-on specifications as required.

11 API & kubectl FAQs

11.1 How Can I Access a Cluster API Server?

You can use either of the following methods to access a 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 toke 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.

For details, see **Kubernetes APIs**.

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 lease one label of the workload uses **app** as its key.
  b. The label of a Service is the same as that of the workload.

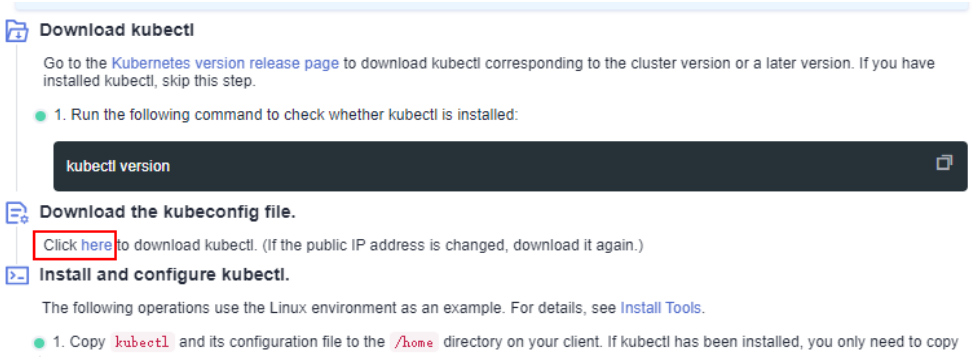
11.3 How Do I Download kubeconfig for Connecting to a Cluster Using kubectl?

Step 1 Log in to the CCE console and click the name of the target cluster to access the cluster console.

Step 2 In the **Connection Information** area, check the kubectl access mode.

Step 3 In the window that slides out from the right, download the kubectl configuration file (**kubeconfig.json**).

Figure 11-1 Downloading kubeconfig.json



----End

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 Cause

"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

security groups, see **How Can I Configure a Security Group Rule for a Cluster?**

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

To grant the Kubernetes permissions to the user, perform the following operations:

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.

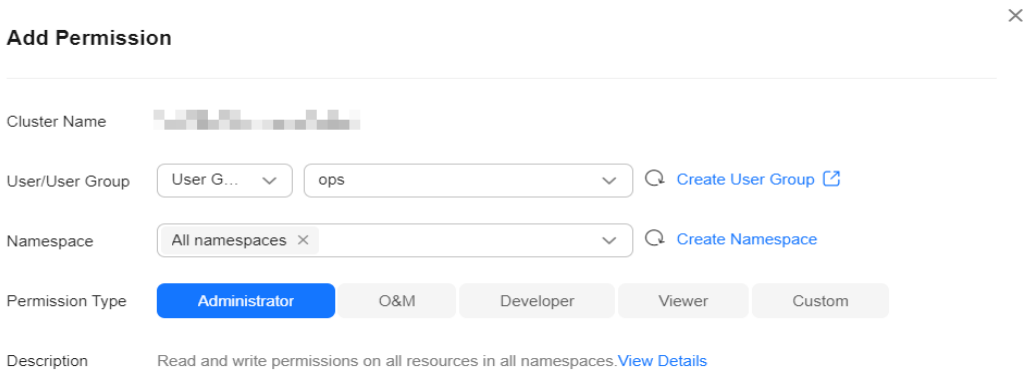
Step 3 Click **Add Permission** in the upper right corner.

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.

Figure 11-2 Configuring namespace permissions

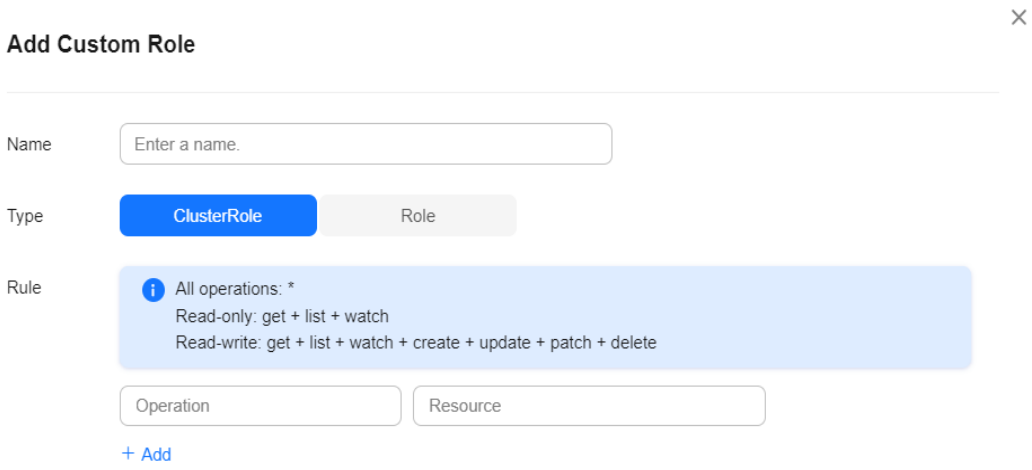


Permissions can be customized as required. After selecting **Custom** for **Permission Type**, click **Add Custom Role** on the right of the **Custom** parameter. In the dialog box displayed, enter a name and select a rule. After the custom rule is created, you can select a value from the **Custom** drop-down list box.

Custom permissions are classified into ClusterRole and Role. Each ClusterRole or Role contains a group of rules that represent related permissions. For details, see **Using RBAC Authorization**.

- A ClusterRole is a cluster-level resource that can be used to configure cluster access permissions.
- A Role is used to configure access permissions in a namespace. When creating a Role, specify the namespace to which the Role belongs.

Figure 11-3 Custom permission



Step 5 Click **OK**.

----End

12 DNS FAQs

12.1 What Should I Do If Domain Name Resolution Fails in a CCE Cluster?

Check Item 1: Whether the coredns Add-on Has Been Installed

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** and 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 CoreDNS instance specifications based on the QPS. If a cluster has more than 100 nodes, you are advised to use NodeLocal DNSCache to improve DNS performance. For details, see **Using NodeLocal DNSCache to Improve DNS Performance**.

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** and verify that CoreDNS is running.

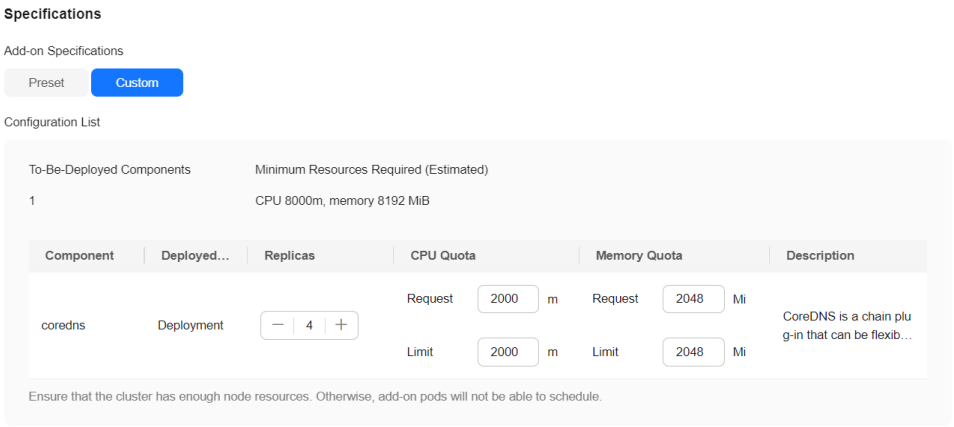
Step 3 Click the CoreDNS add-on name to view the add-on pod list.

Step 4 Click **Monitor** of the add-on pods to view the CPU and memory usage.

If the add-on performance reaches the bottleneck, adjust the coredns add-on specifications.

1. Click **Edit** under the CoreDNS add-on to access the add-on details page.
2. In the **Specifications** area, configure the CoreDNS add-on. You can use the CoreDNS QPS as required.

3. Select **Custom** and configure the number of replicas, CPU quota, and memory quota.



4. Click **OK**.

----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 operations:

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}{"\n"}'
  ```
  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:

  Nodes that use Docker:

  ```
  docker ps --no-trunc | grep coredns | awk '{print $1}'
  ```

  Nodes that use containerd:

  ```
  crictl ps --no-trunc | grep coredns | awk '{print $1}'
  ```

Example:

```
cat /sys/fs/cgroup/cpu/kubepods/
pod27f58662-3979-448e-8f57-09b62bd24ea6/6aa98c323f43d689ac47190bc84cf4fadd23bd8dd25307f773df25003ef0eef0/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:

   Configuring the Stub Domain for CoreDNS

   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**.

12.2 Why Does a Container in a CCE Cluster Fail to Perform DNS Resolution?

Symptom

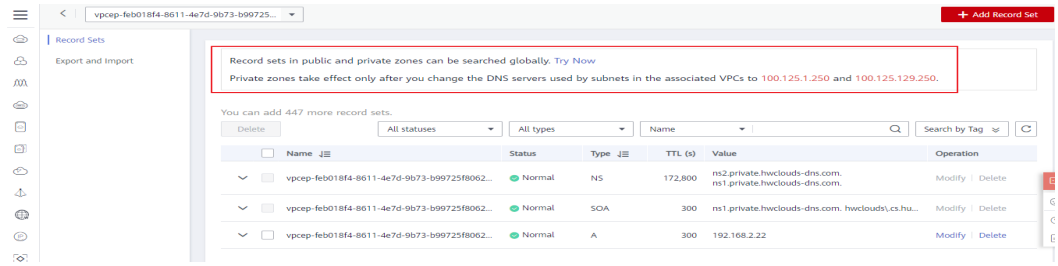
A customer bound its domain name to the private domain names in the DNS service and also to a specific VPC. It is found that the ECSs in the VPC can properly resolve the private domain name but the containers in the VPC cannot.

Application Scenario

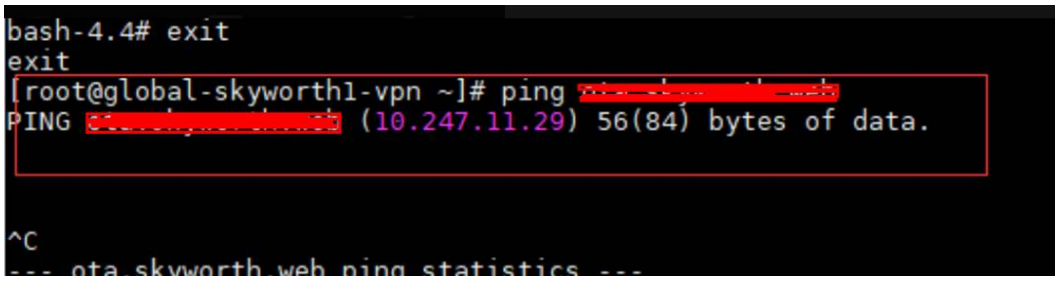
Containers in a VPC cannot resolve domain names.

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.



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.

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.

12.3 Why Cannot the Domain Name of the Tenant Zone Be Resolved After the Subnet DNS Configuration Is Modified?

Symptom

After a DNS server record, for example, 114.114.114.114, is added to the DNS configuration of the user cluster subnet, the domain name of the tenant zone cannot be resolved.

Cause Analysis

CCE configures the subnet DNS information of the user on the node, which is also used by the coredns add-on. As a result, the domain name fails to be resolved by the node container occasionally.

Solution

You are advised to modify the stub domain of the coredns add-on to update the DNS configuration of the user cluster subnet. For details, see **Configuring the Stub Domain for CoreDNS**.

12.4 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.

12.5 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) 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.

12.6 How Can I Address the Issue of CoreDNS Using Deprecated APIs?

Symptom

In a v1.19 CCE cluster where CoreDNS is installed, there may be an issue where CoreDNS continues to listen to v1beta1 of EndpointSlice resources even after upgrading the cluster to v1.25 or v1.27. In v1.25 CCE clusters, v1beta1 of EndpointSlice resources are no longer supported. This results in CoreDNS not being able to receive the latest Service or endpoint change events, leading to the following impacts:

- Failure in resolving new Service domain names: The new domain names cannot be resolved during the use of Services.
- Exception in parsing headless Services: After workload changes and pod recreation, CoreDNS cannot resolve the IP address of the new pod and instead returns the IP address of the old one from the cache, affecting service access.

Cause Analysis

In v1.19 CCE clusters, CoreDNS is using the community version 1.8.4. When CoreDNS starts with this version, it first looks for v1 EndpointSlice resources. If such resources are not found, it falls back to using v1beta1 EndpointSlice resources. Kubernetes 1.19 only supports v1beta1 EndpointSlice resources, so CoreDNS relies on these resources to get Service and endpoint change events. However, Kubernetes 1.25 removes support for the v1beta1 EndpointSlice API. Therefore, after upgrading the cluster to v1.25, CoreDNS still tries to listen to the now-removed v1beta1 EndpointSlice resources, leading to a failure in obtaining Service and endpoint events.

You can check the CoreDNS logs. For details, see **Viewing Container Logs**. If the logs contain the following information, it means that CoreDNS is listening to deprecated APIs:

```
[ERROR] plugin/kubernetes: k8s.io/client-go@v0.26.1/tools/cache/reflector.go:169: Failed to watch *v1beta1.EndpointSlice: failed to list *v1beta1.EndpointSlice: the server could not find the requested resource
```



```
[INFO] plugin/kubernetes: k8s.io/client-go@v0.26.1/tools/cache/reflector.go:169: failed to list
*v1beta1.EndpointSlice: the server could not find the requested resource
```

Solution

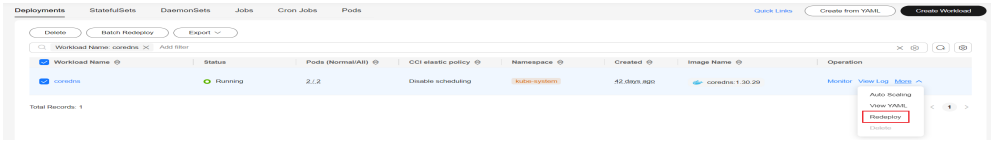
Restart the CoreDNS pod to enable CoreDNS to listen to v1 EndpointSlice resources.

CAUTION

The restart may cause temporary fluctuation of the domain name resolution in the cluster. You are advised to perform this operation during off-peak hours.

You can perform the following operations to recreate the CoreDNS pod:

1. Log in to the CCE console and click the cluster name to access the cluster console. In the navigation pane, choose **Workloads** and select the **kube-system** namespace.
2. Locate the row containing the **coredns** workload and choose **More** > **Redeploy**.



3. Wait for the **coredns** workload to run again.

13 Image Repository FAQs

13.1 How Do I Create a Docker Image and Solve the Problem of Slow Image Pull?

Creating a Docker Image

For details about how to use Dockerfile to customize a Docker image for a simple web application, see **Docker Basics** or **How Do I Create a Docker Image?**

Accelerating Image Pull

Public images may be pulled slowly due to carrier network. You can upload frequently used images to SWR to improve the image pull speed.

Introduction to SWR

SWR provides full-lifecycle container image management, which is easy-to-use, secure, and reliable. SWR enables users to quickly deploy containerized services. SWR can be used as an image repository to store and manage Docker images.

SWR FAQs

General FAQs

13.2 How Do I Upload My Images to CCE?

SWR manages images for CCE. It provides the following ways to upload images:

- **Uploading an Image Through a Container Engine Client**
- **Uploading an Image Through SWR Console**

For details about how to smoothly migrate from Harbor to SWR, see **Synchronizing Images Across Clouds from Harbor to SWR**.

14 Permissions

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.

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 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 IAM permissions can **download** the cluster certificate. Note that information leakage may occur during certificate transmission.

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.

14.4 Why Can't an IAM User Make API Calls?

Symptom

When an IAM user makes an API call, an error message similar to the following is displayed:

```
"code":403,"message":"This user only supports console access, not programmatic access."
```

This error message indicates that the IAM user does not have programmatic access permissions.

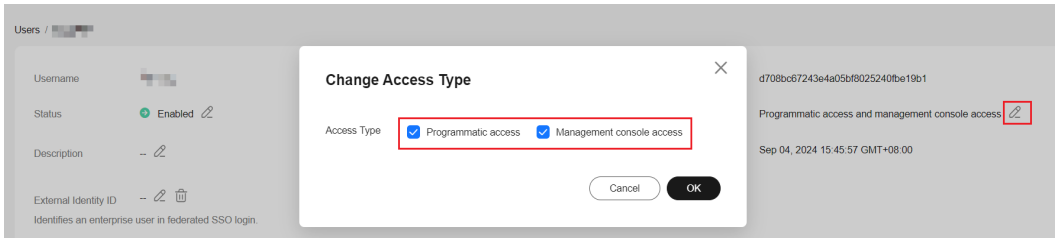
Solution

Step 1 Contact the account administrator and log in to the **IAM console**.

Step 2 Locate the IAM user to be modified and click the username.

Step 3 Change the access mode and select both **Programmatic access** and **Management console access**.

Figure 14-1 Changing the access mode of an IAM user



Step 4 Click **OK**.

----End

14.5 What Is an OBS Global Access Key and How Do I Check Whether a Global Access Key Is Used in a Cluster?

When creating an OBS PVC in a CCE cluster, you need to select an access key (AK/SK). OBS access keys are classified into the following types:

- (Recommended) Custom access key: When a custom access key is used, the YAML file of the PVC contains the **csi.storage.k8s.io/node-publish-secret-namespace** and **csi.storage.k8s.io/node-publish-secret-name** annotations, which specify the secret namespace and secret name for storing keys in the cluster, respectively.

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: jiooij
  namespace: default
  uid: 43c04970-4951-4f31-942f-6290858a8f98
  resourceVersion: '1348742'
  creationTimestamp: '2024-12-24T07:46:10Z'
  annotations:
    csi.storage.k8s.io/fstype: obsfs
    csi.storage.k8s.io/node-publish-secret-name: aksk-wwx
    csi.storage.k8s.io/node-publish-secret-namespace: default
    everest.io/enterprise-project-id: '0'
    everest.io/obs-volume-type: standard
    pv.kubernetes.io/bind-completed: 'yes'
```

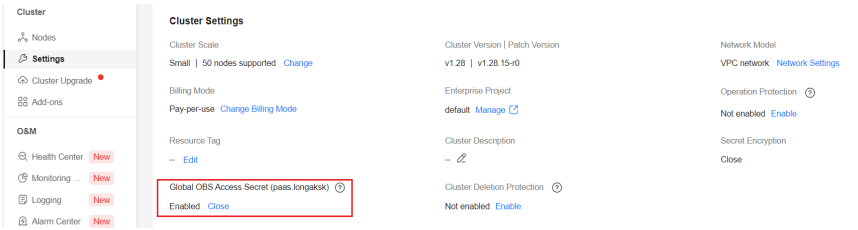
- (Not recommended and supported only by existing users who have uploaded files) Global access key: When a global access key is used, the YAML file of the PVC does not contain the **csi.storage.k8s.io/node-publish-secret-namespace** and **csi.storage.k8s.io/node-publish-secret-name** annotations, and the secret of the global access key is fixed in the **kube-system** namespace and named **paas.longaksk**.


```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: sadfdsaf
  namespace: default
  uid: 73675783-76b6-4d09-94fb-aec17d409b4b
  resourceVersion: '1347883'
  creationTimestamp: '2024-12-24T07:42:29Z'
  annotations:
    csi.storage.k8s.io/fstype: obsfs
    everest.io/obs-volume-type: standard
    pv.kubernetes.io/bind-completed: 'yes'
    pv.kubernetes.io/bound-by-controller: 'yes'
    volume.beta.kubernetes.io/storage-provisioner: everest-csi-provisioner
    volume.kubernetes.io/storage-provisioner: everest-csi-provisioner
  finalizers:
    - kubernetes.io/pvc-protection
```

NOTE

The global access secret (**paas.longaksk**) is used by project. Once a global access secret is used, it is automatically created for each cluster within the same project. However, this can lead to security and management complexities. Therefore, it is not recommended that you use global access secrets.

If you do not need to use the global access key, you can disable it in **Settings** of the cluster. After the operation, CCE deletes the global access key **paas.longaksk** from the **kube-system** namespace. If you need to use this function later, you can enable it again in the settings.

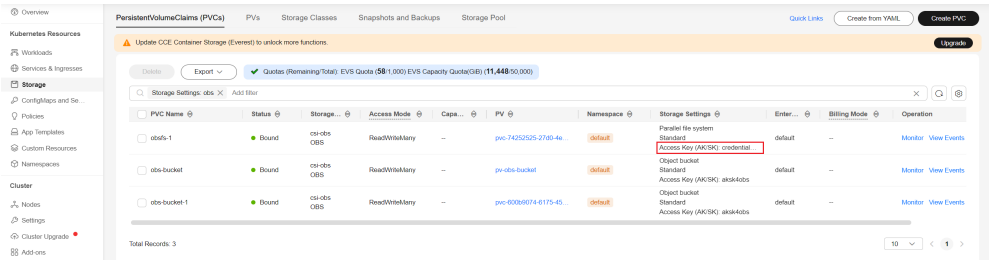


Checking Whether a Global Access Key Is Used in a Cluster

- Using the console

  In the navigation pane, choose **Storage**. In the right pane, select all namespaces, click the **PersistentVolumeClaims (PVCs)** tab, add a filter **Storage Settings: obs**, and view the results in the **Storage Settings** column.

  If the access key is empty, the global access key is used. If the access key is not empty, a custom access key is used.



- Using kubectl

```
kubectl get pvc --all-namespaces -o=custom-columns=Name:'metadata.name',Namespace:'metadata.namespace',secretName:'metadata.annotations.csi\.storage\.k8s\.io\/node-publish-secret-name' | grep \<none\>
```

  You can use the command above to find out which PVC in the cluster is using the global access key. The first column in the command output shows the PVC

name, and the second column indicates the namespace where the PVC is located.

```
obs-test     default     <none>
```

If the command output is blank, it means no PVC is using the global access key in the cluster.

15 Related Services

15.1 What Are the Differences Between CCE and CCI?

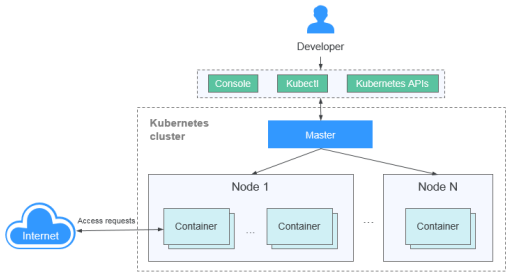
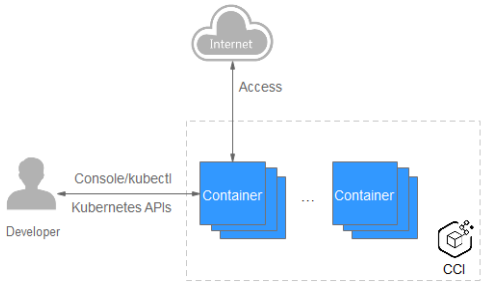
Description

Table 15-1 Introduction to CCE and CCI

Cloud Container Engine (CCE)	Cloud Container Instance (CCI)
CCE provides highly scalable, high-performance, enterprise-class Kubernetes clusters and supports Docker containers. CCE is a one-stop container platform that provides full-stack container services from Kubernetes cluster management, lifecycle management of containerized applications, application service mesh, and Helm charts to add-on management, application scheduling, and monitoring and O&M. With CCE, you can easily deploy, manage, and scale containerized applications on Huawei Cloud. For details, see **What Is Cloud Container Engine?**	CCI is a serverless container service that allows you to run containers without creating or managing server clusters. With CCI, you only need to manage containerized services running on Kubernetes. You can quickly create and run container workloads on CCI without managing clusters and servers. Because of the serverless architecture, CCI frees you from containerized application O&M and allows you to focus on the services themselves. Serverless is an architectural approach that eliminates the need to create, manage, or monitor servers. You simply request the necessary resources for your applications, while server management is handled by dedicated teams. This enables you to focus entirely on application development, boosting efficiency and cutting IT costs. Traditionally, to run containerized workloads using Kubernetes, you need to create a Kubernetes cluster first.

Creation Mode

Table 15-2 Creation modes

Cloud Container Engine (CCE)	Cloud Container Instance (CCI)
CCE is a hosted Kubernetes service for container management. It allows you to create native Kubernetes clusters with just a few clicks. You need to create clusters and nodes to use CCE. They are easy to create on an intuitive console and highly available. You do not need to manage master nodes. 	CCI provides a serverless container engine. When deploying containers on Huawei Cloud, you do not need to purchase and manage ECSs, eliminating the need for O&M and management. You do not need to create clusters, master nodes, or work nodes, but directly start applications. 

Billing

Table 15-3 Different billing modes

Aspect	Cloud Container Engine (CCE)	Cloud Container Instance (CCI)
Pricing	Related resources (such as nodes and bandwidth) will be created when CCE is used. You need to pay for these resources.	CCI instance resources include CPUs, memory, and GPUs. You will be billed by the actual instance resource specifications.
Billing mode	Pay-per-use and yearly/monthly billing modes are supported.	Pay-per-use billing mode is supported.
Minimum pricing unit	By hour	Billed by second. The bill run period is hour.

Application Scenario

Table 15-4 Different application scenarios

Cloud Container Engine (CCE)	Cloud Container Instance (CCI)
Applicable to all scenarios. Generally, large-scale and long-term stable applications are running. For example: • E-commerce • Service mid-end • IT system	Applicable to scenarios with obvious peak and off-peak hours. Resources can be flexibly requested to improve resource utilization. For example: • Batch computing • High-performance computing • Scale-out upon traffic bursts • CI/CD test

Cluster Creation

Table 15-5 Creation modes

Cloud Container Engine (CCE)	Cloud Container Instance (CCI)
Process of using CCE: 1. **Creating a cluster** Configure basic information such as the name, region, and network. 2. **Creating a node** Specify the node specifications and data disk size. 3. **Configuring the cluster** Install cluster add-ons, such as networking, monitoring, and logs. 4. **Creating a workload in the cluster**	Process of using CCI: 1. **Creating a namespace** Configure basic information such as the name, region, and network. 2. **Creating a workload**

Cooperation Between CCE and CCI

The CCE Cloud Bursting Engine for CCI add-on can schedule Deployments, StatefulSets, and jobs running on CCE to CCI when there are traffic spikes. This can reduce consumption caused by cluster scaling.

Functions:

- Automatic pod scaling within seconds: When CCE cluster resources are insufficient, there is no need to add nodes to the CCE cluster. The CCE Cloud Bursting Engine for CCI add-on automatically creates pods in CCI, eliminating the overhead of resizing the CCE cluster.
- Seamlessly integration with Huawei Cloud SWR: You can use public and private images in SWR repositories.

- Support for operations like event synchronization, monitoring, logging, exec command execution, and status query for CCI pods
- You can view the capacity information about virtual elastic nodes.
- CCE and CCI pods can communicate with each other through the Service networks.

For details, see **Elastic Scaling of CCE Pods to CCI**.

15.2 What Are the Differences Between CCE and ServiceStage?

CCE primarily focuses on pod deployment for users, while ServiceStage is designed for service usage.

Technically speaking, ServiceStage can be seen as another form of encapsulation for CCE.

Basic Concepts

Cloud Container Engine (CCE)

CCE provides highly scalable, high-performance, enterprise-class Kubernetes clusters and supports Docker containers. CCE is a one-stop container platform that provides full-stack container services from Kubernetes cluster management, lifecycle management of containerized applications, application service mesh, and Helm charts to add-on management, application scheduling, and monitoring and O&M. With CCE, you can easily deploy, manage, and scale containerized applications on Huawei Cloud.

ServiceStage

ServiceStage is an application and microservice management platform that helps enterprises simplify application lifecycle management from deployment, monitoring, and O&M, to governance. ServiceStage provides a full-stack solution for enterprises to develop microservice, mobile, and web applications. This solution helps enterprises easily migrate various applications onto the cloud, allowing enterprises to focus on service innovation for digital transformation.

