Clusters

Overview

Overview

Immutable Infrastructure

Immutable Infrastructure

Node Management

Overview

Nodes are the fundamental building blocks
virtual machines or physical servers. Each
Pods, including Kubelet, Kube-proxy, and Container Runtime.

Add Nodes to On-Premises Clu

Platform administrators can add nodes to s

Manage No

Supports updati

Node Monit

View node mon

Managed Clusters

overview

Import Clusters

Register Cl

Public Cloud Cluster Initializati

Public cloud cluster initialization.

How to

Creating an On-Premise Cluster

Creating an On-Premise Cluster

Hosted Control Plane

Hosted Control Plane

Cluster Node Planning

Cluster Node Planning

etcd Encryption

etcd Encryption

How to

Add External Address for Built-
Optimize Pod Performance with
Updating Pu

Clusters Overview

The platform supports multiple Kubernetes cluster management models depending on how the underlying infrastructure is provisioned and how the control plane is deployed.

TOC

Platform-Provisioned Infrastructure

Description:

In this model, the platform provisions both the machines and the node operating systems. All nodes use an **Immutable OS**, which ensures a consistent, declarative, and easily recoverable infrastructure state. This model provides full automation across the entire cluster lifecycle — from provisioning to scaling and upgrades.

Examples of Immutable OS:

Common Immutable OS examples include **Fedora CoreOS**, **Flatcar Linux**, and **openSUSE MicroOS**. Currently, the platform supports **MicroOS** for immutable node management.

Responsibilities:

Component	Managed by
Machines / Nodes	Platform
Node OS	Platform (Immutable OS only)
Kubernetes	Platform

User-Provisioned Infrastructure

Description:

In this model, the user provides pre-provisioned physical or virtual machines. The platform installs and manages Kubernetes on these nodes, while node OS management — including provisioning, patching, or replacement — remains under the user's control.

This model is designed for organizations that already have established procedures or automation tools for managing their infrastructure or operating systems.

Responsibilities:

Component	Managed by
Machines / Nodes	User
Node OS	User
Kubernetes	Platform

Hosted Control Plane (HCP)

Description:

Hosted Control Plane (HCP) is a deployment model in which multiple clusters share a single control plane hosted in a dedicated management cluster. Only the control plane components are shared — the worker nodes are still provisioned following one of the two infrastructure models above (either platform-provisioned or user-provisioned).

Characteristics:

- Reduces control plane resource consumption.
- Supports mixed models: worker nodes can be immutable or user-provisioned.
- Ideal for large bare-metal or resource-constrained environments.

Connected Clusters

The platform also supports connecting and managing existing Kubernetes clusters, whether they are public cloud clusters or CNCF-compliant Kubernetes distributions.

Public Cloud Kubernetes

- Connects to managed Kubernetes services such as EKS, AKS, and GKE through cloud-specific providers (e.g., *Alauda Container Platform EKS Provider*).
- Cloud credentials can be securely stored in the platform.
- Enables creation and management of public cloud clusters directly from the platform.

CNCF-Compliant Kubernetes

- Connects any existing Kubernetes cluster conforming to CNCF standards.
- Supports unified visibility, policy control, and monitoring across environments.
- Refer to the Kubernetes Support Matrix.

Tunnel-Based Connectivity

- When the **Global cluster** cannot directly access a **Workload cluster**, a **Tunnel Server** (global side) and **Tunnel Agent** (workload side) establish secure communication.
- Suitable for disconnected or restricted network environments.

Choosing the Right Model

Scenario	Infra Provisioned By	Node OS Managed By	Kubernetes Managed By	Automation Level
Platform-provisioned Infrastructure	Platform	Platform (Immutable OS only)	Platform	Full
User-provisioned Infrastructure	User	User	Platform	Partial
Hosted Control Plane (HCP)	Platform	Shared nodes (Platform)	Platform	Partial
Connected Cluster (Cloud or CNCF)	External Provider	External Provider	Partial / External	Minimal

About Immutable Infrastructure

Immutable Infrastructure uses an immutable operating system to provision Kubernetes clusters. Unlike traditional OS-based clusters, all node configurations are baked into images and remain unchanged after deployment. Cluster upgrades and configuration changes are applied by replacing nodes with new images, ensuring consistency, reliability, and simplified operations throughout the cluster lifecycle.

Note

Because Immutable Infrastructure releases on a different cadence from Alauda Container Platform, the Immutable Infrastructure documentation is now available as a separate documentation set at Immutable Infrastructure ↗.

Node Management

Overview

Nodes are the fundamental building blocks
virtual machines or physical servers. Each
Pods, including Kubelet, Kube-proxy, and Container Runtime.

Add Nodes to On-Premises Clu

Platform administrators can add nodes to s

Manage No

Supports updati

Node Monit

View node mon

Overview

Nodes are the fundamental building blocks of a cluster. Nodes added to a cluster can be either virtual machines or physical servers. Each node contains the essential components required to run Pods, including Kubelet, Kube-proxy, and Container Runtime.

Users with platform management permissions can manage nodes under clusters.

Note: Adding nodes to imported clusters or deleting nodes from imported clusters is not supported.

TOC

Node Types

- **Control Plane Nodes**: Responsible for running cluster components such as kube-apiserver, kube-scheduler, kube-controller-manager, etcd, container networking, and some platform management components.
  - When applications are allowed to be deployed on control plane nodes, control plane nodes can also function as compute nodes.
  - At least 1 control plane node must be added. Setting 2 control plane nodes is not supported. With 3 or more control plane nodes, the cluster becomes a high-availability

cluster (for high-availability clusters, it is recommended to use an odd number of nodes, preferably 3 or 5).

  - When the number of control plane nodes is 3 or more, the cluster has multi-replica disaster recovery capabilities and is considered a high-availability cluster.

- **Compute Nodes**: Responsible for hosting business Pods running on the cluster. The number of compute nodes required in a cluster can typically be planned based on business volume.

Linux Node Availability Check

If you need to build an on-premises cluster, please first refer to the cluster check to ensure all node configurations meet the requirements. All prerequisites must be satisfied, otherwise cluster deployment may fail.

Supported Operating Systems and CPU Models

Please refer to Supported Operating Systems and CPU Models.

Add Nodes to On-Premises Clusters

When a cluster needs to scale up or when abnormal nodes on the cluster need to be replaced with new nodes, you can add control plane nodes and compute nodes to existing **on-premises** workload clusters on the platform by adding nodes.

TOC

Constraints and Limitations

- Nodes to be added to the cluster must be prepared in advance. Please refer to the Node Availability Check Reference to prepare and check nodes to be added to the cluster. Ensure all conditions are met, otherwise cluster deployment may fail.
- The hardware architecture of nodes to be added must be consistent with the cluster's hardware architecture.
- To avoid unpredictable errors, the operating system type of nodes to be added should be consistent with other nodes in the cluster.

- SSH ports and authentication information for nodes added in the same **Add Node** dialog must be unified.
- **Cluster planning guideline**: A cluster must have at least 1 control plane node. Setting exactly 2 control plane nodes is not supported. With 3 or more control plane nodes, the cluster becomes a high-availability cluster (for high-availability clusters, it is recommended to use an odd number of nodes, preferably 3 or 5). **Note**: This requirement applies only when adding or changing control plane capacity; you can safely add worker/compute nodes without being forced to add control plane nodes.
- A node can only belong to one cluster. Nodes to be added cannot be occupied by other clusters.

Prerequisites

- When the global cluster cannot directly access nodes to be added to the cluster through SSH service and needs to access through a proxy, prepare the proxy service in advance. Currently, only SOCKS5 proxy is supported.

Procedure

1. In the left navigation bar, click **Clusters** > **Clusters**.
2. Click the ***cluster name*** of type **On-Premises** where you want to add nodes.
3. Under the **Nodes** tab, click **Add Node**.
4. Refer to Node Configuration Parameters to configure relevant parameters.
5. Click **Add** to perform availability check on the nodes.
   After the check passes, node addition begins, and the nodes are in **Adding** state.

Follow-up Operations

View Execution Progress

On the node list page, you can view the list information of added nodes. For nodes in **Adding** state, you can view the execution progress.

Procedure

1. Click **View Execution Progress** on the right side of nodes in **Adding** state.
2. In the pop-up execution progress dialog, you can view the node execution progress (`status.conditions`).
   Tip: When a certain type is executing or has a failed state with a reason, you can view detailed information about the reason (`status.conditions.reason`) by hovering the cursor over the corresponding reason (displayed in blue text).

Re-add Failed Nodes

After adding nodes, if some nodes fail to be added, a prompt will appear above the node list. Click the **Re-add** button in the prompt box to re-add the failed nodes.

Manage Nodes

TOC

Update Node Labels

Labels↗ are key-value pairs attached to nodes that can define node attributes. After setting labels for nodes, you can easily filter or select nodes by labels. For example: directing Pods to

be scheduled to specific nodes.

Supports updating node labels for nodes in normal state, adding or removing custom node labels.

Procedure

1. In the left navigation bar, click **Cluster Management** > **Clusters**.
2. Click the ***cluster name*** where the node with labels to be updated is located.
3. Under the **Nodes** tab, click **Update Node Labels** on the right side of the node with labels to be updated.
4. Add, modify, or delete node labels.
5. Click **OK**.
   After successfully updating node labels, the number of node labels changes. You can view all label information of the node in the **Node Labels** item in the **Node** information bar.

Stop/Resume Node Scheduling

By setting the scheduling state of nodes, you can control whether newly created Pods in the cluster are allowed to be scheduled to the node.

- **Stop Scheduling**: Newly created Pods are not allowed to be scheduled to the node, but existing Pods running on the node are not affected.
- **Resume Scheduling**: Newly created Pods are allowed to be scheduled to the node.

Procedure

1. In the left navigation bar, click **Clusters** > **Clusters**.
2. Click the ***cluster name*** where the node to stop/resume scheduling is located.
3. Under the **Nodes** tab, click **Stop Scheduling/Resume Scheduling** on the right side of the node to set scheduling state.
4. Click **OK**.

Evict Pods

Evict all Pods except those managed by DaemonSet (daemon set) from nodes in normal state to other nodes in the cluster, and set the node to unschedulable state.

Note: Data from locally stored Pods will be lost after eviction. Please proceed with caution.

Procedure

1. In the left navigation bar, click **Cluster Management** > **Clusters**.
2. Click the ***cluster name*** where the node to evict Pods is located.
3. Under the **Nodes** tab, click the ***node name*** to evict Pods.
4. In the upper right corner, click **Actions** > **Evict Pods**.
5. Review the information of Pods to be evicted, then click **Evict**.

Set Taints

Set taint information for nodes in normal state.

Taints are a property of nodes that allow nodes to refuse to run certain types of Pods or even evict Pods. Taints work together with tolerations on Pods to prevent Pods from being assigned to inappropriate nodes. One or more taints can be applied to each node, and Pods that cannot tolerate these taints will not be accepted by the node.

For example: For a node where we find its memory utilization has reached 91%, it is not recommended to continue scheduling new Pods to this node. We can set a taint for it. After setting the taint, Kubernetes will not schedule Pods to this node.

Learn more...↗

Procedure

1. In the left navigation bar, click **Cluster Management** > **Clusters**.
2. Click the ***cluster name*** where the node to set taints is located.

3. Under the **Nodes** tab, click **Set Taints** on the right side of the node to set taints.
4. Refer to the following description to set the key, value, and effect of taints. Multiple taints can be added to a node.
   Taint attributes consist of `key=value [effect]`.
   `key=value` is used to match Pod tolerations. The taint indicates that the node has been contaminated by `key=value`, and Pod scheduling is not allowed or should avoid scheduling to this node, unless the Pod can tolerate (Tolerations) the `key=value` taint.
   effect is the effect of the taint, with the following three options:

   - **NoSchedule**: Indicates scheduling is not allowed, and already scheduled resources are not affected.
   - **PreferNoSchedule**: Indicates try not to schedule.
   - **NoExecute**: Indicates scheduling is not allowed, and already scheduled resources will be deleted after `tolerationSeconds`.
5. Click **OK**.

Label and Taint Management

The platform supports batch setting of labels and taints for nodes.

Constraints and Limitations

- Before setting device labels, you need to deploy device plugins on the cluster first, such as NVIDIA GPU MPS device plugin, NVIDIA GPU device plugin, GPU Manager device plugin, etc.
  Tip: Device labels are actually node labels. For your convenience, the platform categorizes node labels that device plugins depend on as device labels for quick configuration.

Procedure

1. In the left navigation bar, click **Clusters** > **Clusters**.
2. Click the ***cluster name*** where you want to manage labels and taints.

3. Under the **Nodes** tab, multi-select the nodes you want to manage, and click the **Label and Taint Management** button.
   Tip: You can enter the node labels you care about in the search box on the node list page to quickly filter out the list of nodes you want to manage labels and taints for.
4. In **Batch Operations**, add and fill in the operations you want to perform, then click OK to submit the batch operations to the cluster.

   - **Node Labels**: You can **add/update** specified labels for selected nodes, or **delete** specified labels. When selecting delete, the platform will filter out all label lists on the selected nodes. When the value is set to **Any**, it represents deleting labels on all nodes containing the specified label key.
   - **Taints**: You can **add/update** specified taints for selected nodes, or **delete** specified taints. When selecting delete, the platform will filter out all taint lists on the selected nodes. When the value is set to **Any**, it represents deleting taints on all nodes containing the specified taint key.
   - **Device Labels**: You can set the devices you want to use for selected nodes, where the device list comes from device plugins you have deployed in this cluster.

Enable/Disable Virtualization Switch

When nodes in an on-premises cluster are physical machines, you can control whether Kubernetes is allowed to schedule virtual machines (VMI, VirtualMachineInstance) to the node by enabling/disabling the node virtualization switch.

When the switch is enabled, newly created virtual machines are allowed to be scheduled to the physical machine node; when the switch is disabled, newly created virtual machines are prohibited from being scheduled to the physical machine node, but this does not affect virtual machines already running on the node.

Tip: For related operations and precautions, please refer to Prepare Virtualization Environment.

Delete On-Premises Cluster Nodes

Supports deleting nodes in clusters of type on-premises. For example: deleting failed nodes in on-premises clusters.

Constraints and Limitations

- Nodes in imported clusters are not supported for deletion.
- When there is only one control plane node in the cluster, deleting this control plane node is not supported.

Procedure

1. In the left navigation bar, click **Cluster Management** > **Clusters**.
2. Click the ***cluster name*** of type **On-Premises** where the node to be deleted is located.
3. Under the **Nodes** tab, click **Delete** on the right side of the node to be deleted.
   Tip: If you need to clean up resources under the node after deleting a Linux node, click **Download Cleanup Script** at the bottom of the dialog to download the cleanup script to local. After the node is successfully deleted, log in to the node and execute the cleanup script.
4. Enter the node name, then click **Delete**.

Node Monitoring

View node monitoring data on the node details page.

TIP

- When a cluster has more than 1 node, you can click the ***current node name*** in the resource path area on the node details page to expand the node dropdown list, then click to select a node for quick switching to other node details pages.
- When monitoring components are configured for the cluster, you can view node monitoring data including resource runtime status, resource usage, and resource trend statistics.

TOC

Procedure

1. In the left navigation bar, click **Clusters** > **Clusters**.
2. Click the ***cluster name*** where the target node is located.
3. Under the **Nodes** tab, click the target ***node name***.
4. Click the **Monitoring** tab to enter the node monitoring data display page and view relevant node monitoring data.

TIP

- Hover over a card and click the **Details** icon to view PromQL expressions; click the **Export** icon to export PromQL expressions for all charts on the current page.
- When a cluster has more than 1 node, you can click the ***current node name*** in the resource path area on the node details page to expand the node dropdown list, then click to select a node for quick switching to other node details pages.

TIP

In the storage space statistics display area, when a node has more than 4 storage partitions:

- In the partition total usage pie chart, the top 3 partitions with the highest usage are displayed separately, while remaining partitions are shown as **Others** with their total usage data displayed when hovering over the area;
- In the partition usage bar chart, the top 3 partitions with the highest usage are displayed separately, while remaining partitions are shown as **Others** with their total usage and individual usage rates displayed when hovering over the bars.

The monitoring trend statistics are described in the following table.

Parameter	Description
CPU	**Usage rate**, **request rate**, and **limit rate** of CPU within the specified time range. **Usage rate** = CPU usage of all pods on the node / Total CPU of the node. **Note**: If the CPU usage rate of a node spikes during a certain period, you must first identify the process consuming the most CPU resources. For example, for Java custom applications, memory leaks or infinite loops in the code may cause high CPU usage. **Request rate** = CPU requests of all pods on the node / Total CPU of the node.

Parameter	Description
	Note: If the CPU request rate of a node spikes during a certain period, it may be due to unreasonable cluster oversubscription ratio settings or excessively high request values for pods running on the node, which may cause resource waste. **Limit rate** = CPU limits of all pods on the node / Total CPU of the node. **Note**: If the CPU limit rate of a node spikes during a certain period, it indicates that the limit values for pods running on the node are set too high, which may cause CPU resource waste.
Memory	**Usage rate**, **request rate**, and **limit rate** of memory within the specified time range. **Usage rate** = Memory usage of all pods on the node / Total memory of the node. Memory is one of the important components on a server and serves as a bridge for CPU communication. Therefore, memory performance has a significant impact on the machine. When programs run, data loading, thread concurrency, and I/O buffering all depend on memory. The available memory size determines whether programs can run normally and how they run. **Request rate** = Memory requests of all pods on the node / Total memory of the node. **Note**: If the memory request rate of a node spikes during a certain period, it may be due to unreasonable cluster oversubscription ratio settings or excessively high request values for pods running on the node, which may cause resource waste. **Limit rate** = Memory limits of all pods on the node / Total memory of the node. **Note**: If the memory limit rate of a node spikes during a certain period, it indicates that the limit values for pods running on the node are set too high, which may cause memory resource waste.

Parameter	Description
Storage	**Space usage rate** and **inode usage rate** within the specified time range. **Space usage rate** = Storage space used / Total storage space. By monitoring historical disk space data, you can evaluate disk usage during a given time period. When disk usage is high, you can free up disk space by cleaning up unnecessary images or containers. **Inode usage rate** = Inode storage used / Total inode storage. **Note**: Every file must have an inode to store file metadata such as file creator and creation date. Inodes also consume disk space, and many small cache files can easily lead to inode resource exhaustion. Additionally, when inodes are exhausted but the disk is not full, new files cannot be created on the disk.
System Load	Average CPU load over 1 minute, 5 minutes, and 15 minutes. The value is the ratio of the total number of processes currently being executed by the CPU and waiting to be executed by the CPU to the maximum number of processes the CPU can execute, which is an important indicator of system busy/idle status. **Note**: If the 1-minute/5-minute/15-minute curves are similar over a certain period, it indicates that the cluster's CPU load is relatively stable. If the 1-minute value is much greater than the 15-minute value at a certain time period or specific time point, it indicates that the load in the recent 1 minute is increasing and needs continued observation. Once the 1-minute value exceeds the number of CPUs, it may indicate system overload. You need to further analyze the root cause of the problem. If the 1-minute value is much smaller than the 15-minute value at a certain time period or specific time point, it indicates that the system load is decreasing in the recent 1 minute and generated high load in the previous 15 minutes.

Parameter	Description
Disk Throughput	Disk throughput within the specified time range refers to the speed of data flow transmission by the disk, where transmission data is the sum of read and write data.
Disk IOPS	Disk IOPS within the specified time range is the sum of continuous reads and writes per second, representing a performance metric of the number of read and write operations per second by the disk.
Network Traffic Rate	Network traffic inflow and outflow rates within the specified time range, counted by the node's physical network interface.
Network Packet Rate (packets/sec)	Network packet receive and send rates within the specified time range, counted by the node's physical network interface.

Managed Clusters

overview

overview

Import Clusters

Overview

Import Standard Kubernetes Cl

Import Ope

Import Amazon EKS Cluster

Import Huawei Cloud CCE Cluster (Public Clo

Import an existing CCE (Cloud Container E

zur

Import Alibaba Cloud ACK Cluster

Import Tenc

Register Cluster

Register Cluster

Public Cloud Cluster Initialization

Network Initialization

Public cloud cluster network initialization.

Storage Initialization

Public cloud cluster storage initialization.

How to

Network Configuration for Impc

Fetch import cluster informatio

Trust an ins

Collect Network Data from Custom Named Network Cards

Collect network data from custom named network cards

overview

The platform supports managing existing standard Kubernetes clusters, OpenShift, Amazon EKS (Elastic Kubernetes Service), and Huawei Cloud CCE (Cloud Container Engine) clusters.

TOC

What is a managed cluster?

A managed cluster refers to consolidating existing clusters into a centralized platform for unified governance. It allows enterprises to bring various cluster types—including standard Kubernetes clusters and certain public cloud clusters—under a single control plane. Centralized management improves scalability, availability, and maintainability, enabling better utilization of compute resources and a more efficient cloud environment. You can onboard clusters to the platform via **Access a cluster** or **Register a cluster**.

What's the difference between the two onboarding methods?

They differ only in how onboarding is performed; day-to-day operations are consistent.

- **Import a cluster**: The platform first obtains information about the target cluster and then actively sends access instructions to it. Using this information, the platform establishes a stable connection for centralized monitoring and management, helping administrators oversee the environment and ensure efficient, secure resource utilization.
- **Register a cluster**: Deploy a reverse proxy in the target cluster, which initiates a registration request to the platform. The cluster uses the CLI to automatically establish a tunnel and communicate securely with the platform. Because no cluster details need to be disclosed, security is enhanced and the process is simpler and more efficient.

Import Clusters

Overview

Choose a provider to connect an existing managed cluster to the platform.

- Standard Kubernetes
- OpenShift
- AWS EKS
- Google GKE
- Azure AKS
- Alibaba Cloud ACK
- Tencent Cloud TKE

Import Standard Kubernetes Cluster

Supports integrating standard native Kubernetes clusters deployed with **kubeadm** into the platform for unified management.

TOC

Terminology

Term	Description
Managed Kubernetes Cluster	A type of Kubernetes cluster provided by cloud vendors, where the Master nodes and their components are managed by the vendor. Users cannot log in or manage the Master nodes.
Unmanaged Kubernetes Cluster	In contrast, some cloud vendors provide clusters where users manage the Master nodes, such as Alibaba Cloud ACK Dedicated Edition or Tencent Cloud TKE Independent Cluster.

Prerequisites

- Kubernetes and related components in the cluster must meet the version and parameter requirements.
- If the runtime is Containerd, update the Containerd configuration before integration to ensure distributed storage can be deployed successfully.

Notes

By default, the platform monitors NIC traffic matching `eth.*|en.*|wl.*|ww.*`. If your NIC uses a different naming convention, update the configuration after integration following [Custom NIC Monitoring].

Obtain Registry Address

- To use the registry deployed by the platform during **global cluster** installation, run the following on a global control node:

```
if [ "$(kubectl get productbase -o jsonpath='{.items[].spec.registry.preferPlatformURL}')" = 'false' ]; then
    REGISTRY=$(kubectl get cm -n kube-public global-info -o jsonpath='{.data.registryAddress}')
else
    REGISTRY=$(kubectl get cm -n kube-public global-info -o jsonpath='{.data.platformURL}' | awk -F // '{print $NF}')
fi
echo "Registry address: $REGISTRY"
```

- To use an **external registry**, set **REGISTRY** manually:

```
REGISTRY=<external-registry-address>  # e.g., registry.example.cn:60080 or 192.168.134.43
echo "Registry address: $REGISTRY"
```

Check if Extra Registry Config is Needed

1. Run the following to check if the registry supports HTTPS with a trusted CA certificate:

```
REGISTRY=<registry-address-from-previous-step>

if curl -s -o /dev/null --retry 3 --retry-delay 5 -- "https://${REGISTRY}/v2/"; then
    echo 'Pass: Registry uses a trusted CA certificate. No extra config needed.'
else
    echo 'Fail: Registry does not support HTTPS or uses an untrusted certificate. Follow "Trust Insecure Registry".'
fi
```

2. If check fails, see How to trust an insecure registry?

Get Cluster Info

Refer to How to fetch cluster information?.

Integrate Cluster

1. In the left navigation, go to **Cluster Management** > **Clusters**.
2. Click **Import Cluster**.
3. Configure parameters as below:

Parameter	Description
Registry	Registry storing required platform component images. Options: **Platform Default** (configured during global setup), **Private Registry** (requires address, port, username, password), **Public Registry** (requires cloud credential update).
Cluster Info	Can be entered manually or parsed from a KubeConfig file. Required fields: **Cluster Address**, **CA Certificate** (Base64 decoded if entered manually), and **Authentication** (token or client certificate with cluster-admin rights).

4. Click **Check Connectivity**. The platform verifies network access and auto-detects cluster type.
5. If successful, click **Import** to complete.
   *Progress can be viewed via the **execution progress** dialog (status.conditions). Once integrated, the cluster appears as healthy in the list.*

Network Configuration

Ensure connectivity between the global cluster and the imported cluster.

FAQ

Why is the "Add Node" button disabled?

For both managed and unmanaged clusters, adding nodes through the platform UI is not supported. Add nodes directly or via the vendor.

Which certificates are supported?

1. **Kubernetes Certificates**: Only API Server certificates can be viewed; other certificates are unsupported and will not auto-rotate.
2. **Platform Component Certificates**: Viewable and auto-rotatable.

Which features are unsupported?

- **Managed clusters**: Audit logs are not available.
- **Managed clusters**: ETCD, Scheduler, Controller Manager monitoring not supported (only API Server metrics available).
- **All clusters**: Certificates other than API Server are not supported.

How to fix Containerd runtime causing distributed storage deployment failures?

When using Containerd, distributed storage deployment fails unless you adjust Containerd settings on **all nodes**:

1. Edit `/etc/systemd/system/containerd.service`, set `LimitNOFILE=1048576`.
2. Run `systemctl daemon-reload`.
3. Restart Containerd: `systemctl restart containerd`.
4. On control nodes, restart distributed storage pods:

```
kubectl delete pod --all -n rook-ceph
```

Import OpenShift Cluster

Supports integrating deployed OpenShift clusters into the platform for unified management.

TOC

Prerequisites

- The Kubernetes version and parameters of the cluster must meet the Standard Kubernetes Cluster Requirements.
- During integration, `kubectl` commands are required. Please install the CLI tool on the bastion host that can access the cluster.
- To enable real-time monitoring of metrics such as nodes, workloads (Deployment, StatefulSet, DaemonSet), Pods, and containers, ensure **Prometheus** is already deployed in the target cluster.

Obtain Registry Address

- To use the registry deployed by the platform during **global cluster** installation, run the following command on a global control node:

```
if [ "$(kubectl get productbase -o jsonpath='{.items[].spec.registry.prefe
rPlatformURL}')" = 'false' ]; then
    REGISTRY=$(kubectl get cm -n kube-public global-info -o jsonpath='{.
data.registryAddress}')
else
    REGISTRY=$(kubectl get cm -n kube-public global-info -o jsonpath='{.
data.platformURL}' | awk -F // '{print $NF}')
fi
echo "Registry address is: $REGISTRY"
```

- To use an **external registry**, manually set the **REGISTRY** variable:

```
REGISTRY=<external-registry-address>  # e.g., registry.example.cn:60080 o
r 192.168.134.43
echo "Registry address is: $REGISTRY"
```

Check if Extra Registry Config is Needed

1. Run the following command to check if the registry supports HTTPS and uses a trusted CA certificate:

```
REGISTRY=<registry-address-from-previous-step>

if curl -s -o /dev/null --retry 3 --retry-delay 5 -- "https://${REGISTRY}/v2/"; then
    echo 'Pass: Registry uses a trusted CA certificate. No extra config needed.'
else
    echo 'Fail: Registry does not support HTTPS or uses an untrusted certificate. Follow "Trust Insecure Registry".'
fi
```

2. If the check fails, follow the steps below.

Trust Insecure Registry

1. **Log in to all OCP cluster nodes.**

2. On each node, configure the registry settings:

```
sudo -i
sudo chattr -i /

sudo mkdir -p /etc/systemd/system/crio.service.d/
cat | sudo tee /etc/systemd/system/crio.service.d/99-registry-cpaas-system.conf << 'EOF'
[Service]
ExecStart=
ExecStart=/usr/bin/crio \
          --insecure-registry='<registry-address>' \  # e.g., registry.example.cn:60080 or 192.168.134.43
          $CRIO_CONFIG_OPTIONS \
          $CRIO_RUNTIME_OPTIONS \
          $CRIO_STORAGE_OPTIONS \
          $CRIO_NETWORK_OPTIONS \
          $CRIO_METRICS_OPTIONS
EOF
```

3. Restart `crio`:

```
sudo systemctl daemon-reload && sudo systemctl restart crio
```

Configure DNS for the Cluster

Modify the CoreDNS `ConfigMap` in the global cluster to configure DNS.

1. From the bastion host, get the OCP cluster base domain:

   ```
   oc get dns cluster -o jsonpath='{.spec.baseDomain}'
   ```

   Example output:

   ```
   ocp.example.com
   ```

2. Log in to the platform management console, switch to the **global** cluster, then go to **Cluster Management** > **Resource Management**.

3. Edit the `cpaas-coredns` `ConfigMap` in the `kube-system` namespace.
   Add a new block using the OCP base domain and DNS server address (from `/etc/resolv.conf` on a cluster node).
   Example:

   ```
   Corefile: |
   ocp.example.com:1053 {
       log
       forward . 192.168.31.220
   }
   .:1053 {
       log
       forward . 192.168.31.220
   }
   ```

Get Cluster Info

Choose one of the following:

Method 1 (Recommended): Get the KubeConfig File

1. On the bastion host, search for the `kubeconfig` file and verify it contains an admin context.
2. Copy the kubeconfig file from the bastion host to your local machine:

```
scp root@<bastion-ip>:</path/to/kubeconfig> <local-path>
```

Method 2: Use Token, API Server Address, and CA Certificate

See How to fetch cluster information?.

Import Cluster

1. In the left navigation, go to **Cluster Management** > **Clusters**.
2. Click **Import Cluster**.
3. Configure the parameters:

Parameter	Description
Registry	Registry storing platform component images.**Platform Default**: registry configured during global setup.**Private Registry**: requires registry address, port, username, and password.**Public Registry**: requires updating cloud credentials.
Cluster Info	Either upload the KubeConfig file or enter manually.**Cluster Address**: API Server address.**CA Certificate**: decoded Base64 CA certificate.**Authentication**: token or client certificate with cluster-admin permissions.

4. Click **Check Connectivity**.
5. If successful, click **Import**. Progress can be viewed in the execution log. Once imported, the cluster appears healthy in the list.

Network Configuration

Ensure network connectivity between the global cluster and the imported cluster. See Network Configuration for Imported Clusters.

Deploy Add-ons

After successful integration, go to **Marketplace** to deploy required add-ons such as monitoring, log collection, and log storage.

Before deploying log collection, ensure `/var/cpaas/` *has more than 50GB free space:*

```
df -h /var/cpaas
```

Update Audit Policy

You can modify the audit policy (`spec.audit.profile`) of the cluster:

- **Default**: logs metadata of read/write requests (OAuth access token creation logs the body).
- **WriteRequestBodies**: logs metadata for all requests and bodies of write requests.
- **AllRequestBodies**: logs metadata and bodies of all requests.

Sensitive resources (e.g., Secrets, Routes, OAuthClient) only log metadata.

Update with:

```
oc edit apiserver cluster
```

FAQ

Why is the "Add Node" button disabled?

Adding nodes via the platform UI is not supported. Use the vendor's method.

Which certificates are supported?

1. **Kubernetes Certificates**: Only API Server certificates are visible, no auto-rotation.
2. **Platform Component Certificates**: Visible and auto-rotated.

Which features are unsupported for OpenShift clusters?

- Audit data collection.
- ETCD, Scheduler, Controller Manager monitoring (only API Server metrics available).
- Certificates other than API Server.

Import Amazon EKS Cluster

Connect an existing Amazon EKS (Elastic Kubernetes Service) cluster to the platform for unified management.

TOC

Prerequisites

- The cluster's Kubernetes version and settings meet the requirements in Version compatibility for importing standard Kubernetes clusters.

- The image registry must support HTTPS and provide a valid TLS certificate issued by a public CA.

Prepare the environment

To comply with AWS EKS security practices, perform the following steps in AWS CloudShell.

1. Ensure network connectivity to the AWS Management Console.
2. Search for `cloudshell`, then open CloudShell↗.
3. Verify that the selected region matches your target cluster's region; switch if needed.
4. After CloudShell is ready, clear the terminal and run:

```
# List clusters in the current region and verify your permissions
aws eks list-clusters

# <region-code> is the region of the cluster, e.g., us-west-1
# <my-cluster> is the cluster name from the previous output
aws eks update-kubeconfig --region <region-code> --name <my-cluster>

# The kubeconfig file is saved to "${HOME}/.kube/config"
# Save its content to a file, then upload it to the platform for parsin
g
cat "${HOME}/.kube/config"
```

5. The environment is now ready. For subsequent steps such as **Get cluster information** and **Import cluster**, run any commands against the target cluster from within CloudShell.

Get cluster information

Get the import token

KubeConfig from public-cloud clusters cannot be used directly for import.

Refer to How do I get cluster information? to obtain the cluster import token.

Import the cluster

1. In the left navigation, go to **Cluster Management** > **Clusters**.
2. Click **Import Cluster**.
3. Configure the parameters as follows.

Parameter	Description
Image registry	Registry that stores platform component images required by the cluster. **- Platform default**: the registry configured when the global cluster was deployed. **- Private registry**: a pre-provisioned registry hosting required images. Provide the **private registry address**, **port**, **username**, and **password**. **- Public registry**: a public internet registry. Before use, obtain credentials as described in Update public registry cloud credentials.
Cluster information	**Tip**: Upload the kubeconfig file and let the platform parse it automatically. **Cluster endpoint**: the external API server address exposed by the target cluster. **CA certificate**: the cluster's CA certificate. **Authentication**: use the **token** created in the previous step with **cluster administrator** privileges.

4. Click **Check connectivity** to verify network connectivity and automatically detect the cluster type. The detected type appears as a badge in the top-right of the form.
5. After the connectivity check passes, click **Import**, then confirm.
   Tips:

   - For clusters in the **Importing** state, click the details icon to view progress in the **Execution progress** dialog (`status.conditions`).
   - After a successful import, the cluster list shows key information. The cluster status is Normal and cluster operations are available.

Network configuration

Ensure the global cluster and the imported cluster have network connectivity. See Network Configuration for Imported Clusters.

Next steps

Initialize Ingress and storage

If you need Ingress and storage capabilities, see Initialize Ingress for AWS EKS and Initialize storage for AWS EKS.

FAQ

The Add Node button is disabled after import. How can I add nodes?

Adding nodes from the platform UI is not supported. Please add nodes through your cluster provider.

Which certificates are supported by certificate management for imported clusters?

1. **Kubernetes certificates**: You can view the API server certificate only. Other Kubernetes certificates are not visible and are not auto-rotated.
2. **Platform component certificates**: Visible in the platform and support automatic rotation.

What features are not supported for imported AWS EKS clusters?

- Audit data is not available.
- ETCD, Scheduler, and Controller Manager metrics are not supported; a subset of API server charts is available.
- Certificate details other than the Kubernetes API server certificate are not available.

Import GKE Cluster

The platform supports importing Google GKE clusters.

TOC

Prerequisites

- The Kubernetes version and components on the cluster meet the version requirements for importing public cloud clusters.

- Ensure the cluster type is a standard cluster and the account has permissions to maintain the control plane. Autopilot clusters are not currently supported.
- The image repository must support HTTPS access and provide a valid TLS certificate authenticated by a public certification authority.

Preparing the Operating Environment

To comply with GKE security standards, the following steps must be performed using Cloud Shell.

1. Ensure network connectivity with Google.
2. Access the **Clusters** page↗ in the Kubernetes Engine feature; find the cluster to be imported, click on cluster details, and select the **Connect** button.
3. In the popup dialog, copy the command for configuring kubectl command-line access permissions and click the **Run in Cloud Shell** button.
4. Wait for Cloud Shell to be ready, clear the command line, paste the content copied in the previous step, and execute it.
5. The environment is now ready. All subsequent commands executed in the importing cluster environment for steps such as **Obtaining Cluster Information** and **Importing Cluster** should be executed in Cloud Shell.

Obtaining Cluster Information

Obtaining the API Server Address and CA Certificate of the Target Cluster

1. Access the **Clusters** page↗ in the Kubernetes Engine feature and click to enter the details page of the target cluster.
2. The API Server address can be found in the **External endpoints** section.
3. To obtain the CA certificate, use one of the following methods in Cloud Shell:
   Method A: Get the CA certificate from your kubeconfig:

```
gcloud container clusters get-credentials <cluster-name> --zone <zone>
kubectl config view --raw -o jsonpath='{.clusters[0].cluster.certificat
e-authority-data}' | base64 -d
```

Method B: Get the CA certificate directly from the cluster:

```
gcloud container clusters describe <cluster-name> --zone <zone> --forma
t='get(masterAuth.clusterCaCertificate)' | base64 -d
```

Note: The certificate must be Base64-decoded before pasting into the import form.

Obtaining the Target Cluster Token

The KubeConfig file of public cloud clusters cannot be directly used for importing clusters.

Please refer to the FAQ How to obtain cluster information? to obtain the target cluster token.

Importing the Cluster

1. In the left navigation bar, click **Clusters** > **Clusters**.
2. Click **Manage Cluster** > **Import Cluster**.
3. Configure the relevant parameters according to the following instructions.

Parameter	Description
Image Repository	Repository for storing platform component images required by the cluster. **- Platform Default**: Image repository configured during global deployment. **- Private Repository**: Pre-built repository storing platform required components. Requires input of **Private Image Repository Address**, **Port**, **Username**, and **Password** for accessing the image repository. **- Public Repository**: Use public image repository services on the internet. Before use, you must first refer to Update Public Repository Cloud Credentials to obtain repository authentication permissions.

Parameter	Description
Cluster Information	**Cluster Information**: Includes the target cluster token and the API Server address and CA certificate of the target cluster. **Cluster Address**: The access address where the target cluster exposes the API Server for platform access to the cluster's API Server. **CA Certificate**: CA certificate of the target cluster. **Note**: When manually inputting, you need to enter the Base64 decoded certificate. **Authentication Method**: Authentication method for the target cluster, requires using the **token** (Token) with **cluster management permissions** created in the previous step for authentication.

4. Click **Check Connectivity** to verify network connectivity with the target cluster and automatically identify the cluster type, which will be displayed as a badge in the top-right corner of the form.
5. After connectivity check passes, click **Import** and confirm.

> **TIP**
>
> - Click the **Details** icon on the right side of clusters in **Importing** status to view the cluster execution progress (status.conditions) in the popup **Execution Progress** dialog.
> - After successful cluster import, you can view key cluster information in the cluster list, the cluster status shows as normal, and you can perform cluster-related operations.

Network Configuration

Ensure network connectivity between the global cluster and the imported cluster. See Network Configuration for Imported Clusters.

Post-Import Operations

Ingress and Storage Initialization

After importing the cluster, if you need to use Ingress and storage-related features, please refer to Google GKE Ingress Controller Configuration and Google GKE Storage Configuration.

Frequently Asked Questions

How to add nodes when the "Add Node" button is grayed out after importing the cluster?

Adding nodes through the platform interface is not supported. Please contact the cluster provider to add nodes.

What certificates are supported by the certificate management functionality for imported clusters?

1. **Kubernetes Certificates**: All imported clusters only support viewing APIServer certificate information in the platform certificate management interface. Other Kubernetes certificates cannot be viewed and automatic rotation is not supported.
2. **Platform Component Certificates**: All imported clusters can view platform component certificate information in the platform certificate management interface and support automatic rotation.

Import Huawei Cloud CCE Cluster (Public Cloud)

Import an existing CCE (Cloud Container Engine) cluster (public cloud) into the platform for unified management.

TOC

Prerequisites

- The Kubernetes version and parameters on the cluster meet the Standard Kubernetes Cluster Component Version and Parameter Requirements.
- Ensure the cluster type is Huawei Cloud CCE cluster and the account has permissions to maintain the control plane. Turbo clusters are not currently supported.
- Huawei Cloud CCE clusters do not have the ability to access external network resources by default after creation. Before importing the cluster, ensure that the cluster to be imported can access the platform access address.

Obtain Image Registry Address

- To use the **platform-deployed** image registry from the global cluster deployment, execute the following command on the **control node of the global cluster** to obtain the address:

```
if [ "$(kubectl get productbase -o jsonpath='{.items[].spec.registry.pr
eferPlatformURL}')" = 'false' ]; then
    REGISTRY=$(kubectl get cm -n kube-public global-info -o jsonpath
='{.data.registryAddress}')
else
    REGISTRY=$(kubectl get cm -n kube-public global-info -o jsonpath
='{.data.platformURL}' | awk -F \// '{print $NF}')
fi
echo "Image registry address is: $REGISTRY"
```

- To use an **external image registry**, manually set the **REGISTRY** variable.

```
REGISTRY=<external image registry address>  # Valid examples: registry.
example.cn:60080 or 192.168.134.43
echo "Image registry address is: $REGISTRY"
```

Determine if Image Registry Requires Additional Configuration

1. Execute the following command to determine whether the specified image registry supports HTTPS access and uses certificates issued by trusted CA authorities:

```
REGISTRY=<image registry address obtained from the "Obtain Image Regist
ry Address" section>

if curl -s -o /dev/null --retry 3 --retry-delay 5 -- "https://${REGISTR
Y}/v2/"; then
    echo 'Test passed: The image registry uses certificates issued by t
rusted CA authorities. You do not need to execute the content in the "T
rust Insecure Image Registry" section.'
else
    echo 'Test failed: The image registry does not support HTTPS or the
certificate is not trusted. Please refer to the "Trust Insecure Image R
egistry" section for configuration.'
fi
```

2. If the test fails, please refer to the FAQ How to trust an insecure image registry?.

Obtain Cluster Information

1. Ensure network connectivity with the Huawei Cloud console.
2. Access the Cluster Management page ↗ of the `Cloud Container Engine CCE` feature; find the cluster to be imported and click the cluster name to enter the details page.
3. As shown in the figure below, follow the navigation to find the download KubeConfig file button: `Cluster Information - Connection Information - kubectl - Configuration`, and download the KubeConfig file.

Obtain Import Cluster Token

The KubeConfig file of public cloud clusters cannot be directly used for cluster import.

Please refer to the FAQ How to obtain cluster information? to obtain the import cluster token.

Import Cluster

1. In the left navigation bar, click **Cluster Management** > **Clusters**.
2. Click **Import Cluster**.

3. Configure the `Image Registry` related parameters according to the following instructions.

Parameter	Description
Image Registry	Repository for storing platform component images required by the cluster. **- Platform Default**: Image registry configured during global cluster deployment. **- Private Registry**: Pre-built registry storing platform required components. You need to enter the **private image registry address**, **port**, **username**, and **password** for accessing the image registry. **- Public Registry**: Use image registry services located on the public network. Before use, you need to first refer to Update Public Image Registry Cloud Credentials to obtain registry authentication permissions.
Cluster Information	**Tip**: Please upload the KubeConfig file for automatic parsing and filling by the platform. **Cluster Address**: The access address of the API Server exposed by the imported cluster, used for the platform to access the API Server of the imported cluster. **CA Certificate**: The CA certificate of the imported cluster. **Authentication Method**: The authentication method of the imported cluster, which requires using a **token** with **cluster management permissions** created in the previous step for authentication.

4. Click the `Parse KubeConfig File` button and submit the KubeConfig file downloaded in the previous step. The platform will automatically parse and fill in the `Cluster Information` related parameters.
5. Click **Check Connectivity** to check network connectivity with the imported cluster and automatically identify the type of the imported cluster. The cluster type will be displayed as a badge in the upper right corner of the form.
6. After connectivity check passes, click **Import** and confirm.
   Tips:

   - Click the

icon on the right side of a cluster in **Importing** status to view the cluster's execution progress (status.conditions) in the popup **Execution Progress** dialog.

- After successful cluster import, you can view the cluster's key information in the cluster list. The cluster status displays as normal and you can perform cluster-related operations.

Network Configuration

To ensure network connectivity between the global cluster and the imported cluster, you must refer to Imported Cluster Network Configuration.

Follow-up Operations

Ingress (Inbound Rules) and Storage Initialization

After importing the cluster, if you need to use Ingress (inbound rules) and storage-related features, please refer to Huawei Cloud CCE Cluster Ingress Initialization Configuration and Huawei Cloud CCE Cluster Storage Initialization Configuration.

FAQ

After importing the cluster, the add node button is grayed out. How to add nodes?

Adding nodes through the platform interface is not supported. Please contact the cluster provider to add nodes.

What certificates does the certificate management feature support for imported clusters?

1. **Kubernetes Certificates**: All imported clusters only support viewing APIServer certificate information in the platform certificate management interface. Viewing other Kubernetes certificates and automatic rotation are not supported.
2. **Platform Component Certificates**: All imported clusters can view platform component certificate information in the platform certificate management interface and support automatic rotation.

What other features are not supported for imported Huawei Cloud CCE clusters?

- Audit data retrieval is not supported.
- ETCD, Scheduler, and Controller Manager related monitoring information are not supported. APIServer partial monitoring charts are supported.
- Cluster certificate related information other than Kubernetes APIServer certificates cannot be retrieved.

Import Azure AKS Cluster

Import an existing Azure AKS cluster into the platform for unified management.

TOC

Prerequisites

- The Kubernetes version and parameters on the cluster must meet the Standard Kubernetes Cluster Component Version and Parameter Requirements.

> **TIP**
>
> - If AKS nodes cannot access the global cluster, refer to the FAQ: How to configure AKS node external IP security group rules.

- The image registry must support HTTPS access and provide a valid TLS certificate authenticated by a public certification authority.

Prepare the Operating Environment

To comply with Azure AKS security standards, the following steps must be performed using Cloud Shell.

1. Ensure network connectivity with Azure Console.
2. Open the Kubernetes Services page ↗, locate the cluster you want to import, and click to enter the cluster overview page.
3. Click the `Connect` button, which will open a floating window titled `Connect to <import cluster name>`. Follow the instructions to open Cloud Shell and configure the operating environment.

Obtain Cluster Information

Obtain Import Clusters Token

The KubeConfig file of public cloud clusters cannot be directly used for cluster import.

Please refer to the FAQ How to obtain cluster information? to obtain the import cluster token.

Import Cluster

1. In the left navigation bar, click **Cluster Management** > **Clusters**.
2. Click **Import Cluster**.
3. Configure the relevant parameters according to the following instructions.

Parameter	Description
Image Registry	The registry that stores platform component images required by the cluster. **- Platform Default**: The image registry configured when deploying the global cluster. **- Private Registry**: A pre-built registry that stores platform-required component images. You need to enter the **Private Image Registry Address**, **Port**, **Username**, and **Password** for accessing the image registry. **- Public Registry**: Use a public image registry service on the internet. Before use, you must first refer to Update Public Image Registry Cloud Credentials to obtain registry authentication permissions.
Cluster Information	**Tip**: Please upload a KubeConfig file, and the platform will automatically parse and fill in the information. **Cluster Address**: The access address of the API Server exposed by the import cluster, used by the platform to access the import cluster's API Server. **CA Certificate**: The CA certificate of the import cluster. **Authentication Method**: The authentication method of the import cluster, which requires using a **Token** with **cluster management permissions** created in the previous step for authentication.

4. Click **Check Connectivity** to verify network connectivity with the import cluster and automatically identify the import cluster type. The cluster type will be displayed as a badge in the upper right corner of the form.
5. After connectivity check passes, click **Import** and confirm.

TIP

- Click the **Details** icon on the right side of a cluster in **Importing** status to view the cluster's execution progress (status.conditions) in the popup **Execution Progress** dialog.
- After the cluster is successfully imported, you can view the cluster's key information in the cluster list. The cluster status will show as normal, and you can perform cluster-related

operations.

Network Configuration

Ensure the global cluster and the imported cluster have network connectivity. See Network Configuration for Imported Clusters.

Post-Import Operations

Ingress (Inbound Rules) and Storage Initialization

After importing the cluster, if you need to use Ingress (inbound rules) and storage-related features, please refer to Azure AKS Cluster Ingress Initialization Configuration and Azure AKS Cluster Storage Initialization Configuration.

Frequently Asked Questions

How to configure AKS node external IP security group rules

Nodes only have internal IPs by default. The external IP is configured on a frontend load balancer (LB), which is used for outbound traffic by default. This LB is controlled by the AKS principal. Direct manual modification of this configuration may cause issues. You can allow traffic through **Kubernetes** > **Properties** > **Infrastructure Resource Group** > **Network Security Group** > **Add Outbound/Inbound All Rules**.

How to access AKS node

To view logs of system components such as Kubelet, CNI, and kernel, you need to SSH into the node first. It is recommended to use the `kubectl-node-shell` plugin instead of assigning public IP addresses to each node.

Option 1: Using kubectl node-shell

Official Link ↗

Option 2: Using debug

Official Link ↗

> **NOTE**
>
> This example requires kubectl version 1.25 or later, which includes the GA `kubectl debug` command.

```
kubectl debug node/aks-newadd-41368356-vmss000002 -it --image=mcr.microsoft.com/dotnet/runtime-deps:6.0
chroot /host
```

Azure ALB using internal load balancer

Refer to Official Link ↗

```yaml
apiVersion: v1
kind: Service
metadata:
  name: internal-app
  namespace: cpaas-system
  annotations:
    service.beta.kubernetes.io/azure-load-balancer-internal: "true"
spec:
  type: LoadBalancer
  ports:
  - name: http-port
    port: 80
    protocol: TCP
  - name: https-port
    port: 443
    protocol: TCP
  selector:
    service.cpaas.io/name: deployment-aks-alb
    service_name: alb2-aks-alb
```

Azure ALB using external load balancer

Deploy a highly available ALB with the access address configured as the external LB.

```
apiVersion: v1
kind: Service
metadata:
  name: azure-alb
  namespace: cpaas-system
spec:
  type: LoadBalancer
  ports:
  - name: http-port
    port: 80
    protocol: TCP
  - name: https-port
    port: 443
    protocol: TCP
  - name: prom-port
    port: 11780
    protocol: TCP
  - name: prom2-port
    port: 11781
    protocol: TCP
  - name: prom3-port
    port: 15012
    protocol: TCP
  selector:
    service_name: alb2-cpaas-system
```

If it has been deployed in advance, you can use the following command to modify it.

```
kubectl edit helmrequest -n cpaas-system uat-cluster-aks-alb
```

The add node button is grayed out after importing the cluster. How to add nodes?

Adding nodes through the platform interface is not supported. Please contact the cluster provider to add nodes.

What certificates are supported by the certificate management feature for imported clusters?

1. **Kubernetes Certificates**: All imported clusters only support viewing APIServer certificate information in the platform certificate management interface. Other Kubernetes certificates cannot be viewed and automatic rotation is not supported.
2. **Platform Component Certificates**: All imported clusters can view platform component certificate information in the platform certificate management interface and support automatic rotation.

What other features are not supported for imported AKS clusters?

- Audit data retrieval is not supported.
- ETCD, Scheduler, and Controller Manager related monitoring information is not supported. APIServer partial monitoring charts are supported.
- Cluster certificate-related information other than Kubernetes APIServer certificates cannot be retrieved.

Import Alibaba Cloud ACK Cluster

Import existing Alibaba Cloud ACK managed clusters (Managed Kubernetes) or Alibaba Cloud ACK dedicated clusters (Dedicated Kubernetes) for unified platform management.

TIP

For product information about ACK managed clusters (Managed Kubernetes) or Alibaba Cloud ACK dedicated clusters (Dedicated Kubernetes), refer to the official documentation ↗.

TOC

Prerequisites

- The Kubernetes version and parameters on the cluster meet the component version and parameter requirements for importing standard Kubernetes clusters.

Get Image Registry Address

- To use the **platform-deployed** image registry from the global cluster deployment, execute the following command on the **control node of the global cluster** to get the address:

```
if [ "$(kubectl get productbase -o jsonpath='{.items[].spec.registry.prefe
rPlatformURL}')" = 'false' ]; then
    REGISTRY=$(kubectl get cm -n kube-public global-info -o jsonpath='{.
data.registryAddress}')
else
    REGISTRY=$(kubectl get cm -n kube-public global-info -o jsonpath='{.
data.platformURL}' | awk -F \// '{print $NF}')
fi
echo "Image registry address is: $REGISTRY"
```

- To use an **external image registry**, manually set the **REGISTRY** variable.

```
REGISTRY=<external image registry address>  # Valid examples: registry.e
xample.cn:60080 or 192.168.134.43
echo "Image registry address is: $REGISTRY"
```

Determine if Image Registry Requires Additional Configuration

1. Execute the following command to determine if the specified image registry supports HTTPS access and uses certificates issued by trusted CA authorities:

```
REGISTRY=<image registry address obtained from the "Get Image Registry Address" section>

if curl -s -o /dev/null --retry 3 --retry-delay 5 -- "https://${REGISTRY}/v2/"; then
    echo 'Test passed: The image registry uses certificates issued by trusted CA authorities. You do not need to execute the content in the "Trust Insecure Image Registry" section.'
else
    echo 'Test failed: The image registry does not support HTTPS or the certificate is not trusted. Please refer to the "Trust Insecure Image Registry" section for configuration.'
fi
```

2. If the test fails, refer to the FAQ How to trust insecure image registries?.

Get KubeConfig

1. Log in to the Alibaba Cloud Container Service management platform.
2. In the left navigation bar of the console, click **Clusters**.
3. On the **Cluster List** page, click the target cluster name or **Details** under the **Actions** column on the right side of the target cluster.
4. On the **Cluster Information** page, click the **Connection Information** tab, then click **Generate Temporary KubeConfig**.
5. In the **Temporary KubeConfig** dialog, set the validity period of the temporary credentials and the method to access the cluster (including public network access and internal network access).
6. Click **Generate Temporary KubeConfig**, then click **Copy** to copy the content and save it to the **KubeConfig** file on your local computer.
7. After the cluster is successfully imported, you can revoke the temporary credentials.

Import Cluster

1. In the left navigation bar, click **Cluster Management** > **Clusters**.

2. Click **Import Cluster**.
3. Configure the relevant parameters according to the following instructions.

Parameter	Description
Image Registry	Repository for storing platform component images required by the cluster. **- Platform Default**: Image registry configured during global cluster deployment. **- Private Registry**: Pre-built registry that stores platform-required component images. You need to enter the **private image registry address**, **port**, **username**, and **password** for accessing the image registry. **- Public Registry**: Use public image registry services on the internet. Before use, you need to refer to Update Public Repository Cloud Credentials to obtain repository authentication permissions.
Cluster Information	**Tip**: Can be filled manually or uploaded via KubeConfig file for automatic parsing and filling by the platform. **Parse KubeConfig File**: After uploading the obtained KubeConfig file, the platform will automatically parse and fill the **Cluster Information**. You can modify the automatically filled information. **Cluster Address**: The access address of the cluster's externally exposed API Server, used by the platform to access the cluster's API Server. **CA Certificate**: The cluster's CA certificate. **Note**: When entering manually, you need to enter the Base64-decoded certificate. **Authentication Method**: Authentication method for accessing the cluster. You need to use a **token** or **certificate authentication (client certificate and key)** with **cluster management permissions** for authentication.

4. Click **Check Connectivity** to check network connectivity with the cluster to be imported and automatically identify the type of cluster to be imported. The cluster type will be displayed as a badge in the upper right corner of the form.
5. After connectivity check passes, click **Import** and confirm.

TIP

- Click the **Details** icon on the right side of a cluster in **Importing** status to view the cluster's execution progress (status.conditions) in the popup **Execution Progress** dialog.
- After the cluster is successfully imported, you can view the cluster's key information in the cluster list. The cluster status shows as normal and you can perform cluster-related

operations.

Network Configuration

Ensure network connectivity between the global cluster and the cluster to be imported. See Network Configuration for Imported Clusters.

FAQ

How to handle port conflicts between Alibaba Cloud monitoring and platform monitoring components?

When Alibaba Cloud's built-in monitoring and platform monitoring components coexist, port conflicts will occur. It is recommended to uninstall Alibaba Cloud monitoring and keep only platform monitoring.

How to use public network access for Alibaba Cloud clusters?

If using public network access for Alibaba Cloud clusters, you can bind a public IP on Alibaba Cloud.

After importing a cluster, the add node button is grayed out. How to add nodes?

Both **Alibaba Cloud ACK managed clusters** and **ACK dedicated clusters** do not support adding nodes through the platform interface. Please add them in the backend or contact the cluster provider to add them.

Which certificates are supported by the certificate management function for imported clusters?

1. **Kubernetes Certificates**: All imported clusters only support viewing APIServer certificate information in the platform certificate management interface. They do not support viewing other Kubernetes certificates and do not support automatic rotation.
2. **Platform Component Certificates**: All imported clusters can view platform component certificate information in the platform certificate management interface and support automatic rotation.

What other features are not supported for imported Alibaba Cloud ACK managed clusters and ACK dedicated clusters?

- **Alibaba Cloud ACK managed clusters** do not support obtaining audit data.
- **Alibaba Cloud ACK managed clusters** do not support ETCD, Scheduler, Controller Manager related monitoring information, but support some APIServer monitoring charts.
- Both **Alibaba Cloud ACK managed clusters** and **ACK dedicated clusters** do not support obtaining cluster certificate-related information except for Kubernetes APIServer certificates.

Import Tencent Cloud TKE Cluster

Import existing Tencent Cloud TKE Dedicated clusters or Tencent Cloud TKE Managed clusters into the platform for unified management.

TIP

For product introduction of TKE Dedicated clusters or Tencent Cloud TKE Managed clusters, please refer to the official documentation ↗.

TOC

Prerequisites

- The Kubernetes version and parameters on the cluster meet the component version and parameter requirements for importing standard Kubernetes clusters.
- The image registry must support HTTPS access and provide a valid TLS certificate issued by a public certificate authority.

Obtain Image Registry Address

- To use the **platform-deployed** image registry configured during global cluster deployment, execute the following command on the **control node of the global cluster** to obtain the address:

```
if [ "$(kubectl get productbase -o jsonpath='{.items[].spec.registry.prefe
rPlatformURL}')" = 'false' ]; then
    REGISTRY=$(kubectl get cm -n kube-public global-info -o jsonpath='{.
data.registryAddress}')
else
    REGISTRY=$(kubectl get cm -n kube-public global-info -o jsonpath='{.
data.platformURL}' | awk -F \// '{print $NF}')
fi
echo "Image registry address is: $REGISTRY"
```

- To use an **external image registry**, manually set the **REGISTRY** variable.

```
REGISTRY=<external image registry address>  # Valid examples: registry.e
xample.cn:60080 or 192.168.134.43
echo "Image registry address is: $REGISTRY"
```

Determine if Image Registry Requires Additional Configuration

1. Execute the following command to determine whether the specified image registry supports HTTPS access and uses a certificate issued by a trusted CA:

```
REGISTRY=<image registry address obtained from the "Obtain Image Regist
ry Address" section>

if curl -s -o /dev/null --retry 3 --retry-delay 5 -- "https://${REGISTR
Y}/v2/"; then
    echo 'Verification passed: The image registry uses a certificate is
sued by a trusted CA. It is not necessary to execute the content in the
"Trust Unsafe Image Registry" section.'
else
    echo 'Verification failed: The image registry does not support HTTP
S or the certificate is not trusted. Please refer to the "Trust Unsafe
Image Registry" section for configuration.'
fi
```

2. If verification fails, please refer to the FAQ How to trust an unsafe image registry?.

Obtain KubeConfig

1. Log in to the Tencent Cloud Container Service management platform.
2. In **Cluster Details** > **Basic Information**, view the **Cluster APIServer** information.
3. Select **Internet Access** or **Intranet Access** based on the actual customer network, then download **Kubeconfig** and save it to your local computer.

Import Cluster

1. In the left navigation bar, click **Cluster Management** > **Clusters**.
2. Click **Import Cluster**.
3. Configure the relevant parameters according to the following instructions.

Parameter	Description
Image Registry	Registry for storing platform component images required by the cluster. - **Platform Default**: Image registry configured during global deployment. - **Private Registry**: Pre-built registry that stores platform-required component images. You need to input the **private image registry address**, **port**,

Parameter	Description
	username, and **password** for accessing the image registry. **- Public Registry**: Use image registry services located on the public network. Before use, you need to first refer to Update Public Registry Cloud Credentials to obtain registry authentication permissions.
Cluster Information	**Tip**: Can be filled manually or uploaded via KubeConfig file for automatic parsing and filling by the platform. **Parse KubeConfig File**: After uploading the obtained KubeConfig file, the platform will automatically parse and fill in the **Cluster Information**, and you can modify the automatically filled information. **Cluster Address**: The access address of the cluster's externally exposed API Server, used by the platform to access the cluster's API Server. **CA Certificate**: The cluster's CA certificate. **Note**: When manually inputting, you need to input the Base64-decoded certificate. **Authentication Method**: Authentication method for accessing the cluster, requires using a **token** (Token) or **certificate authentication (client certificate and key)** with **cluster management permissions**.

4. Click **Check Connectivity** to verify network connectivity with the cluster to be imported and automatically identify the type of cluster to be imported. The cluster type will be displayed as a badge in the upper right corner of the form.
5. After connectivity check passes, click **Import** and confirm.
   Tip:

   - Click the **Details** icon on the right side of a cluster in **Importing** status to view the cluster's execution progress (status.conditions) in the popup **Execution Progress** dialog.
   - After successful cluster import, you can view the cluster's key information in the cluster list. The cluster status displays as normal, and cluster-related operations can be performed.

Network Configuration

Ensure network connectivity between the global cluster and the cluster to be imported. You must refer to Network Configuration for Importing Clusters.

FAQ

After importing the cluster, the "Add Node" button is grayed out. How to add nodes?

Both **TKE Dedicated clusters** and **TKE Managed clusters** do not support adding nodes through the platform interface. Please add them in the backend or contact the cluster provider to add them.

What certificates does the certificate management function for imported clusters support?

1. **Kubernetes Certificates**: All imported clusters only support viewing APIServer certificate information in the platform certificate management interface. They do not support viewing other Kubernetes certificates and do not support automatic rotation.
2. **Platform Component Certificates**: All imported clusters can view platform component certificate information in the platform certificate management interface and support automatic rotation.

What other features are not supported for imported TKE Managed clusters and TKE Dedicated clusters?

- **TKE Managed clusters** do not support obtaining audit data.
- **TKE Managed clusters** do not support ETCD, Scheduler, Controller Manager related monitoring information, but support partial APIServer monitoring charts.
- Both **TKE Managed clusters** and **TKE Dedicated clusters** do not support obtaining cluster certificate-related information except for Kubernetes APIServer certificates.

Register Cluster

This is a method of deploying a reverse proxy service in the managed cluster, where the managed cluster actively initiates registration requests to the platform.

TOC

Prerequisites

- Depending on the type of managed cluster, the versions and parameters of Kubernetes and other components on the managed cluster must meet the Version and Parameter Requirements for Managed Clusters.
- The image registry must support HTTPS access and provide a valid TLS certificate authenticated by a public certification authority. If this cannot be met, refer to the FAQ How to Trust Insecure Image Registries?

  Note: The **Public Registry** provided by the platform on the public network already meets HTTPS access requirements. You only need to verify whether the **Platform Default** and

Private Registry support HTTPS access.

- If the runtime component of the cluster to be connected is Containerd, you need to modify the Containerd configuration before connecting the cluster to ensure successful deployment of distributed storage.

Important Notes

The platform's network card traffic monitoring recognizes network cards with names matching `eth\.\|en\.\|wl\.*\|ww\.*` by default. Therefore, if you use network cards with other naming conventions, please refer to the Collecting Network Data from Custom Named Network Cards documentation to modify the corresponding resources after cluster connection, ensuring the platform can properly monitor network card traffic.

Register Cluster

1. In the left navigation bar, click **Clusters** > **Clusters**.
2. Click **Managed Clusters** > **Register Cluster**.
3. Configure the registry parameters for storing platform component images required by the registered cluster according to the following instructions.

Parameter	Description
Platform Default	Image registry used when deploying global components.
Private Registry	External image registry that you have set up in advance. You need to enter the **Private Image Registry Address**, **Port**, **Username**, and **Password** for accessing the image registry.
Public Registry	Pull required images through the public image registry provided by the platform. You need to ensure that your cluster can access the public network. Before use, you need to first refer to Update Public Network Image Registry Cloud Credentials to obtain registry authentication permissions.

4. Click **Create**, obtain the registration command on the **Registration Command** page and run the command in the cluster to be registered.
   Note: The registration command is valid for 24 hours. Please re-obtain it after expiration.

View Registration Command

You can find the cluster waiting for registration in the cluster list and click **View Registration Command**. Please perform the registration operation before the expiration time.

FAQ

How to Resolve Distributed Storage Deployment Failure When the Runtime Component of the Connected Cluster is Containerd?

When the runtime component of the connected cluster is Containerd, distributed storage deployment will fail. To resolve this issue, you need to manually modify the Containerd configuration information on **all nodes** of the cluster and restart Containerd.

Note: If you modify the Containerd configuration by following the steps below before deploying distributed storage, you do not need to execute step four.

1. Log in to the cluster node and edit the `/etc/systemd/system/containerd.service` file, changing the `LimitNOFILE` parameter value to `1048576`.
2. Execute the command `systemctl daemon-reload` to reload the configuration.
3. Execute the command `systemctl restart containerd` to restart Containerd.
4. Execute the command `kubectl delete pod --all -n rook-ceph` on the cluster control node to restart all Pods in the rook-ceph namespace to make the configuration effective.

Public Cloud Cluster Initialization

Network Initialization

AWS EKS Cluster Network Initialization Configuration

TOC

Support Overview

Feature	Support Status	Requirements
LoadBalancer Service	Supported	Optionally deploy AWS Load Balancer Controller. Without this controller, LoadBalancer capabilities are limited.

Feature	Support Status	Requirements
Ingress	Supported	Optionally deploy AWS Load Balancer Controller. Optionally enable **Ingress Class** functionality (once enabled, you can manually select ingress classes when creating ingress through the form interface).

Prerequisites

- Prepare two subnets with the **kubernetes.io/role/elb** tag. For shared subnets, add the **kubernetes.io/cluster/<cluster-name>: shared** tag. See Adding Tags to Subnets.
- If you have created an EKS cluster, import the Amazon EKS cluster.
- Ensure kubectl, Helm, AWS CLI, and eksctl tools are available before deploying AWS Load Balancer Controller.
  Note: After installing the tools, configure login information using the user who created the cluster via AWS CLI, and test if AWS CLI and eksctl tools are correctly installed.
- Obtain ACCOUNT_ID, **REGION**, and **CLUSTER_NAME** in advance, and replace `<ACCOUNT_ID>`, `<REGION>`, and `<CLUSTER_NAME>` in the documentation with the actual values.
  Note: **ACCOUNT_ID** is the Account ID of the user who created the cluster, **REGION** is the cluster region, and **CLUSTER_NAME** is the cluster name.
- Update and verify the Kubeconfig configuration file.

Configuration Steps

Deploy AWS Load Balancer Controller

Note: For detailed information on deploying AWS Load Balancer Controller, see official documentation ↗.

Configure OIDC Provider

Kubernetes clusters use OpenID Connect (OIDC) for identity management and are associated with an OIDC issuer URL. To enable AWS Identity in the cluster and allow IAM roles for Service Accounts, create an IAM OIDC Provider associated with the cluster's OIDC issuer URL.

Execute the following command in eksctl to configure the OIDC Provider:

```
eksctl utils associate-iam-oidc-provider --region=<REGION> --cluster=<CLU
STER_NAME> --approve
```

Configure Service Account

Execute the following commands to create an IAM policy and create a Service Account named `aws-load-balancer-controller`, associating it with an IAM role:

```
curl -o aws-load-balancer-controller-iam-policy.json https://raw.githubus
ercontent.com/kubernetes-sigs/aws-load-balancer-controller/v2.4.7/docs/in
stall/iam_policy.json
aws iam create-policy \
    --policy-name <CLUSTER_NAME>-AWSLoadBalancerControllerIAMPolicy \
    --policy-document file://aws-load-balancer-controller-iam-policy.json

eksctl create iamserviceaccount \
  --cluster=<CLUSTER_NAME> \
  --namespace=kube-system \
  --name=aws-load-balancer-controller \
  --role-name AmazonEKSLoadBalancerControllerRole \
  --attach-policy-arn=arn:aws:iam::<ACCOUNT_ID>:policy/<CLUSTER_NAME>-AWS
LoadBalancerControllerIAMPolicy \
  --approve
```

Deploy AWS Load Balancer Controller to Cluster

Execute the following commands in eksctl to deploy AWS Load Balancer Controller:

1. Add the eks-charts repository:

   ```
   helm repo add eks https://aws.github.io/eks-charts
   ```

2. Update the local repository:

```
helm repo update eks
```

3. Deploy the AWS Load Balancer Controller Helm Chart to the cluster:
   Note: `aws-load-balancer-controller` is the **Service Account** created in Configure Service Account.

```
helm install aws-load-balancer-controller eks/aws-load-balancer-control
ler \
  -n kube-system \
  --version=v2.4.7 \
  --set ingressClassConfig.default=true \
  --set clusterName=<CLUSTER_NAME> \
  --set serviceAccount.create=false \
  --set serviceAccount.name=aws-load-balancer-controller
```

Create Ingress and LoadBalancer Services

You can create ingress and LoadBalancer services simultaneously or choose one based on your needs.

Create Ingress

1. In **Container Platform**, click **Network** > **Ingress** in the left navigation.
2. Click **Create Ingress** and select **EKS Ingress Class** for **Ingress Class**.
3. Select **Protocol**. Default is **HTTP**. For **HTTPS**, first create a certificate and select it.
4. Switch to **YAML** and add the following annotations. For details, see annotation documentation ↗:

```
alb.ingress.kubernetes.io/scheme: internet-facing ## Specify public acc
ess
alb.ingress.kubernetes.io/target-type: ip ## Route traffic directly to
pods
```

5. Click **Create**.

Create LoadBalancer Service

1. In **Container Platform**, click **Network** > **Services** in the left navigation.
2. Click **Create Service** and select **LoadBalancer** for **External Access**.
3. Expand **annotations** and fill in LoadBalancer service annotations as needed.
4. Click **Create**.

Related Operations

Test AWS CLI and eksctl Installation

- Execute the following command. If it returns a cluster list, AWS CLI is correctly installed:

```
aws eks list-clusters
```

- Execute the following command. If it returns a cluster list, eksctl is correctly installed:

```
eksctl get clusters
```

Get ACCOUNT_ID

Execute `aws sts get-caller-identity` to get **ACCOUNT_ID**. The `651168850570` in the response is the **ACCOUNT_ID**:

```
{
  "ARN": "arn:aws:iam::651168850570:user/jwshi"
}
```

Kubeconfig Configuration File

1. Execute the following command to update the Kubeconfig file for the specified region:

```
aws eks --region <REGION>  update-kubeconfig --name <CLUSTER_NAME>
```

2. Execute the following command to verify the Kubeconfig file. If it returns information normally, the configuration is correct:

```
kubectl get svc -n cpaas-system
```

Add Tags to Subnets

1. Execute the following command to get cluster subnets:

```
eksctl get cluster --name <CLUSTER_NAME>
```

2. Execute the following command to get subnet details:

```
aws ec2 describe-subnets
```

3. Execute the following commands to add tags to subnets. Replace `<subnet-id>` with actual values. See Subnet auto-discovery ↗:

   - Add the `kubernetes.io/role/elb` tag to subnets:

   ```
   aws ec2 create-tags --resources <subnet-id> --tags Key=kubernetes.io/
   role/elb,Value="1"
   ```

   - Add the `kubernetes.io/cluster/<CLUSTER_NAME>: shared` tag to shared subnets:

   ```
   aws ec2 create-tags --resources <subnet-id> --tags Key=kubernetes.io/
   cluster/<CLUSTER_NAME>,Value="shared"
   ```

Create Certificate

When using HTTPS protocol, save HTTPS certificate credentials as a Secret (TLS type) in advance.

1. In **Container Platform**, click **Configuration** > **Secrets** in the left navigation.
2. Click **Create Secret**.
3. Select **TLS** type and import or fill in **Certificate** and **Private Key** as needed.
4. Click **Create**.

AWS EKS Supplementary Information

TOC

Terminology

Abbreviation	Full Name	Description
eks-clb	Classic Load Balancer	AWS default load balancer. Has issues in certain situations and is not recommended.
eks-nlb	Network Load Balancer	AWS Layer 4 load balancer that performs load balancing at TCP/UDP level, suitable for scenarios requiring higher-level network control.
eks-alb	Application Load Balancer	AWS Layer 7 load balancer. Compared to eks-nlb, eks-alb can parse HTTP/HTTPS protocols and distribute requests more intelligently, suitable for web applications.

Abbreviation	Full Name	Description
aws-lb	AWS Load Balancer	Load balancer installed on Kubernetes that can automatically create eks-nlb and eks-alb based on LoadBalancer Services and Ingress in Kubernetes to meet application load balancing needs.
Platform Load Balancer	-	Platform's proprietary Layer 7 load balancer.
Service Annotations	-	Metadata attached to objects in key-value pairs. This additional information can be recognized and utilized to enhance and simplify management of various aspects of Kubernetes resources. Annotations can be explanatory text without specific functionality, specify cloud provider configurations or behaviors, or specify configuration parameters and tools. Very powerful functionality.

Important Notes

When creating load balancers, it's recommended to manually configure service annotations to ensure the platform load balancer correctly uses aws-lb. If the appropriate service annotations are not configured correctly, the platform will default to using eks-clb, which has UDP-related issues that may cause unexpected situations.

EKS Using aws-lb to Provide External Access for Container Network Load Balancers

Service Annotation Configuration Instructions

1. In the corresponding cluster, execute the following command using kubectl to find all Pods in the kube-system namespace with names containing "aws-load":

```
kubectl get pod -n kube-system |grep aws-load
```

2. Create a load balancer; for detailed creation steps and parameters, see the Load Balancer creation section in AWS EKS Service Annotation Instructions.

   - If the above command returns no related Pods, it means the cluster does not have AWS Load Balancer Controller installed. No service annotations are needed; create the load balancer directly.
   - If the above command returns related Pods, it means the cluster has AWS Load Balancer Controller installed. When creating a load balancer in the corresponding cluster, add the following service annotations. For annotation details, see AWS EKS Service Annotation Instructions:
     - `service.beta.kubernetes.io/aws-load-balancer-type: external //Required`
     - `service.beta.kubernetes.io/aws-load-balancer-nlb-target-type: ip //Required`
     - `service.beta.kubernetes.io/aws-load-balancer-scheme: internet-facing //Optional. Add this annotation if public network support is needed.`

Access Address Acquisition Method

- When creating container network type load balancers, the filled service annotations will be set on the LoadBalancer Service corresponding to the platform load balancer.
- In public clouds, LoadBalancer Services with appropriate service annotations will be recognized by the public cloud and assigned addresses. The platform load balancer will read this address and set it as its own access address.

Huawei Cloud CCE Cluster Network Initialization Configuration

TOC

Support Overview

Feature	Support Status	Requirements
LoadBalancer Service	Default Support	No additional deployment required.
Ingress	Default Support	Optionally enable **Ingress Class** functionality (once enabled, you can manually select ingress classes when creating ingress through the form interface). No additional deployment required.

Prerequisites

If you have created a CCE cluster, import the CCE cluster (Public Cloud).

Configuration Steps

You can create ingress and LoadBalancer services simultaneously or choose one based on your needs.

Create Ingress

There are two methods to create ingress. **Method 1: Manual Ingress Class Selection** is recommended.

Note: Avoid creating two ingress resources with the same **path**.

(Recommended) Method 1: Manual Ingress Class Selection

1. In **Container Platform**, click **Network** > **Ingress** in the left navigation.
2. Click **Create Ingress** and select **CCE Ingress Class** for **Ingress Class**.
3. Select **Protocol**. Default is **HTTP**. For **HTTPS**, first create a certificate and select it.
4. Switch to **YAML** and add the following annotations based on your default Ingress Controller type. For annotation details, see Using Annotations to Configure Load Balancers ↗:
   Note: Replace the **values** in the annotations below with actual environment values.

Default Ingress Controller Type	Annotations
Shared (Auto-create)	`kubernetes.io/elb.autocreate: '{"type":"public","bandwidth_name":"{random}","bandwidth_chargemode":"traffic","bandwidth_size":5,"bandwidt` `kubernetes.io/elb.class: union`

Default Ingress Controller Type	Annotations

Shared (Reuse)	`kubernetes.io/elb.class: union`
	`kubernetes.io/elb.id: <Load Balancer Instance ID>`
	`kubernetes.io/elb.port: '80'`

Dedicated (Auto-create)	`kubernetes.io/elb.autocreate: '{"type":"public","bandwidth_name":"<Bar`
	`Name>","bandwidth_chargemode":"traffic","bandwidth_size":5,"bandwidth_s`
	`["<AZ A>","<AZ B>","<AZ C>"],"elb_virsubnet_ids":["<ELB Virtual Subnet`
	`ID>"],"l7_flavor_name":"L7_flavor.elb.s1.small","l4_flavor_name":"L4_fl`
	`kubernetes.io/elb.class: performance`
	`kubernetes.io/elb.port: "80"`

Dedicated (Reuse)	`kubernetes.io/elb.class: performance`
	`kubernetes.io/elb.id: <Load Balancer Instance ID>`
	`kubernetes.io/elb.port: "80"`

5. Click **Create**. Once created, you can access cluster services through ELB.

Method 2: Use Default Ingress Class

1. Create an IngressClass YAML file with the following content. For details, see Default Ingress Class ↗:

```yaml
apiVersion: networking.k8s.io/v1
kind: IngressClass
metadata:
  annotations:
    ingressclass.kubernetes.io/is-default-class: "true"
  name: cce
spec:
  controller: alauda/cce
```

2. Save the file and apply it to the imported cluster. Replace `<filename.yaml>` with your actual YAML filename:

```
kubectl apply -f <filename.yaml>
```

3. In **Container Platform**, click **Network** > **Ingress** in the left navigation.
4. Select **Protocol**. Default is **HTTP**. For **HTTPS**, first create a certificate and select it.
5. Click **Create**. Once created, you can access cluster services through ELB.

Create LoadBalancer Service

1. In **Container Platform**, click **Network** > **Services** in the left navigation.
2. Click **Create Service** and select **LoadBalancer** for **External Access**.
3. Expand **annotations** and fill in LoadBalancer service annotations as needed.
4. Click **Create**.

Related Operations

Create Certificate

When using HTTPS protocol, save HTTPS certificate credentials as a Secret (TLS type) in advance.

1. In **Container Platform**, click **Configuration** > **Secrets** in the left navigation.
2. Click **Create Secret**.
3. Select **TLS** type and import or fill in **Certificate** and **Private Key** as needed.
4. Click **Create**.

Azure AKS Cluster Network Initialization Configuration

TOC

Support Overview

Feature	Support Status	Requirements
LoadBalancer Service	Default Support	No additional deployment required.
Ingress	Supported	Optionally deploy Ingress Controller. Optionally enable **Ingress Class** functionality (once enabled, you can manually select ingress classes when creating ingress through the form interface).

Prerequisites

If you have created an AKS cluster, import the Azure AKS cluster.

Configuration Steps

Deploy Ingress Controller

AKS uses **container network mode** and leverages **Nginx Ingress Controller** to manage load balancers, while providing external access addresses for virtual IP addresses (VIPs) in the container internal network through **LoadBalancer** type **Services**.

1. Log in to Microsoft Azure and access your created AKS cluster.
2. In the left navigation, click **Kubernetes Resources** > **Services and Ingresses**.
3. Click **Create**, select **Ingress (Preview)** from the dropdown, and it will prompt and automatically create an Ingress Controller.
4. Click **Enable** and wait for completion.

Create Ingress and LoadBalancer Services

You can create ingress and LoadBalancer services simultaneously or choose one based on your needs.

Create Ingress

1. In **Container Platform**, click **Network** > **Ingress** in the left navigation.
2. Click **Create Ingress** and select **webapprouting.kubernetes.azure.com** for **Ingress Class**.
3. Select **Protocol**. Default is **HTTP**. For **HTTPS**, first create a certificate and select it.
4. Click **Create**.

Create LoadBalancer Service

1. In **Container Platform**, click **Network** > **Services** in the left navigation.

2. Click **Create Service** and select **LoadBalancer** for **External Access**.
3. Expand **annotations** and fill in LoadBalancer service annotations as needed.
4. Click **Create**.

Related Operations

Create Certificate

When using HTTPS protocol, save HTTPS certificate credentials as a Secret (TLS type) in advance.

1. In **Container Platform**, click **Configuration** > **Secrets** in the left navigation.
2. Click **Create Secret**.
3. Select **TLS** type and import or fill in **Certificate** and **Private Key** as needed.
4. Click **Create**.

Google GKE Cluster Network Initialization Configuration

TOC

Support Overview

Feature	Support Status	Requirements
LoadBalancer Service	Default Support	No additional deployment required.
Ingress	Default Support	Optionally enable **Ingress Class** functionality (once enabled, you can manually select ingress classes when

Feature	Support Status	Requirements
		creating ingress through the form interface). No additional deployment required.

Prerequisites

If you have created a GKE cluster, import the GKE cluster.

Configuration Steps

Deploy Ingress Controller

No manual deployment is required. GKE provides a managed built-in Ingress controller called GKE Ingress. This controller maps Ingress resources to Google Cloud Load Balancers to handle HTTP(S) workloads in GKE, making configuration simpler and more automated.

Create Ingress and LoadBalancer Services

You can create ingress and LoadBalancer services simultaneously or choose one based on your needs.

Create Ingress

1. In **Container Platform**, click **Network** > **Ingress** in the left navigation.
2. Click **Create Ingress** and select **GKE Ingress Class** for **Ingress Class**.
3. Select **Protocol**. Default is **HTTP**. For **HTTPS**, first create a certificate and select it.
4. Click **Create**. Wait approximately 5 minutes for GKE platform to automatically assign a public IP address to the ingress.
   Note: Different ingress resources will be assigned different public IP addresses.

Create LoadBalancer Service

1. In **Container Platform**, click **Network** > **Services** in the left navigation.
2. Click **Create Service** and select **LoadBalancer** for **External Access**.
3. Expand **annotations** and fill in LoadBalancer service annotations as needed.
4. Click **Create**.

Related Operations

View Ingress Resources in Google Cloud

1. Go to **Google Cloud > Kubernetes Engine** and click **Services and Ingress** in the left navigation.
2. Click **INGRESS**.
3. View information about corresponding Ingress resources in the list.

Create Certificate

When using HTTPS protocol, save HTTPS certificate credentials as a Secret (TLS type) in advance.

1. In **Container Platform**, click **Configuration** > **Secrets** in the left navigation.
2. Click **Create Secret**.
3. Select **TLS** type and import or fill in **Certificate** and **Private Key** as needed.
4. Click **Create**.

Storage Initialization

Overview

- Amazon Elastic Kubernetes Service (Amazon EKS) is Amazon's managed Kubernetes service for running Kubernetes on AWS Cloud and on-premises data centers. In the cloud, Amazon EKS automatically manages the availability and scalability of Kubernetes control plane nodes responsible for scheduling containers, managing application availability, storing cluster data, and other critical tasks, providing a consistent and fully supported Kubernetes solution.
- Huawei Cloud Container Engine (CCE) provides highly reliable, high-performance enterprise-level container application management services, supporting Kubernetes community native applications and tools, simplifying the construction of automated container runtime environments on the cloud.
- Azure Kubernetes Service (AKS) provides the fastest way to start developing and deploying cloud-native applications on Azure, data centers, or edge using built-in code-to-cloud pipelines and guardrails, with unified management and governance for on-premises, edge, and multi-cloud Kubernetes clusters.
- Google Kubernetes Engine (GKE) provides an extremely scalable, fully automated Kubernetes service that can be used with almost no Kubernetes expertise required. Its advantages include increased speed, reduced risk, and lower total cost of ownership, with built-in security posture and observability tools, and industry-leading autoscaling solutions that can scale up to 15,000 nodes.

TOC

Google GKE Clusters

Storage Class Support

AWS EKS Clusters

Storage Type	Default Storage Class	Create PVC with RWO Access Mode	Create PVC with RWX Access Mode	PVC Expansion	PVC Snapshots
File Storage	efs-sc	Supported	Supported	Not Supported	Not Supported
Block Storage	ebs-sc	Supported	Not Supported	Supported	Not Supported

Huawei Cloud CCE Clusters

Storage Type	Default Storage Class	Create PVC with RWO Access Mode	Create PVC with RWX Access Mode	PVC Expansion	PVC Snapshots
File Storage	csi-nas	Not Supported	Supported	Supported	Not Supported
Block Storage	csi-disk	Supported	Not Supported	Supported	Not Supported

Azure AKS Clusters

Storage Type	Default Storage Class	Create PVC with RWO Access Mode	Create PVC with RWX Access Mode	PVC Expansion	PVC Snapshots
File Storage	azurefile	Supported	Supported	Supported	Not Supported
Block Storage	default	Supported	Not Supported	Supported	Not Supported

Google GKE Clusters

Storage Type	Default Storage Class	Create PVC with RWO Access Mode	Create PVC with RWX Access Mode	PVC Expansion	PVC Snapshots
File Storage	standard-rwx	Supported	Supported	Supported	Not Supported
Block Storage	standard-rwo	Supported	Not Supported	Supported	Not Supported

AWS EKS Cluster Storage Initialization Configuration

Platform integration with AWS EKS and storage initialization configuration.

TOC

Constraints and Limitations

- The default efs-sc file storage class may not support permission modifications after mounting, which may cause some applications like PostgreSQL and Jenkins to fail to run properly.
- A1 series instances are not supported by AL2023 AMIs, which prevents the EBS block storage plugin (Amazon EBS CSI Driver) from deploying properly. The EBS CSI driver has GA multi-architecture/ARM support, so the limitation is with AMI/instance support rather

than the driver itself. If you need to use EBS block storage classes, avoid using the following instance types and consider Graviton2/3 alternatives instead:

- a1.medium
- a1.large
- a1.xlarge
- a1.2xlarge
- a1.4xlarge

Recommended alternatives: Use Graviton2/3 instance families such as m6g, c6g, r6g, t4g, etc., which provide better performance and full EBS CSI driver support.

Prerequisites

- Ensure kubectl ↗ and AWS CLI tools are available.
- If you have created an EKS cluster, import the Amazon EKS cluster; if not, create an AWS EKS cluster.
- Deploy the EFS file storage plugin **Amazon EFS CSI Driver** and EBS block storage plugin **Amazon EBS CSI Driver** in the EKS cluster.
  Note: If using EFS file storage, create file storage in the EKS region and record the **File System ID** from the **File System**.

Configuration Steps

Create Storage Classes

1. Go to **Platform Management** and click **Storage Management** > **Storage Classes** in the left navigation.
2. Click the dropdown next to **Create Storage Class** > **Create from YAML**.
3. Add the following content to the YAML file to create default storage classes as needed. The default storage class name for file storage ↗ is **efs-sc**, and for block storage ↗ is **ebs-sc**.

   - EFS File Storage

Note: Replace `<File System ID>` with the actual **File System ID**, e.g., `fileSystemId: fs-05aef9e1edd309f2b`.

```
kind: StorageClass
apiVersion: storage.k8s.io/v1
metadata:
  name: efs-sc
provisioner: efs.csi.aws.com
parameters:
  provisioningMode: efs-ap
  fileSystemId: <File System ID>
  directoryPerms: "755"
```

- EBS Block Storage

```
allowVolumeExpansion: true
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: ebs-sc
provisioner: ebs.csi.aws.com
reclaimPolicy: Delete
volumeBindingMode: WaitForFirstConsumer
```

4. Click **Create**.

   Note: If the default storage classes don't meet requirements, create new storage classes following the above steps and modify parameters as needed. See Available Storage Class Parameters.

Modify Storage Class Project Assignment

1. In the left navigation, click **Storage Management** > **Storage Classes**.
2. Click the three dots next to the storage class named **efs-sc** or **ebs-sc** > **Update Project**.
3. Select the **Project Assignment** method as needed and click **Update** to assign the storage class to projects.

Related Operations

Configure Available Storage Class Parameters

- EFS File Storage Available Parameters

Parameter	Optional Values	Default Value	Optional	Description
az		""	true	Used for cross-account mounting. If specified, uses the mount target associated with az for cross-account mounting; if not specified, randomly selects a mount target for cross-account mounting.
basePath			true	Path for creating dynamically provisioned access points. If not specified, access points are created under the file system root directory.
directoryPerms			false	Directory permissions for creating Access Point root directory ↗.
uid			true	POSIX user ID for creating Access

Parameter	Optional Values	Default Value	Optional	Description
				Point root directory ↗.
gid			true	POSIX group ID for creating Access Point root directory ↗.
gidRangeStart		50000	true	Starting range of POSIX group IDs to apply when creating access point root directory ↗. Not needed if uid/gid are set.
gidRangeEnd		7000000	true	Ending range of POSIX group IDs. Not needed if uid/gid are set.
subPathPattern			true	Template for constructing subpaths where each access point created under dynamic provisioning is located. Can consist of fixed strings and limited variables, similar to the "subPathPattern"

Parameter	Optional Values	Default Value	Optional	Description
				variable in nfs-subdir-external-provisioner chart. Optional parameters are .PVC.name, .PVC.namespace, and .PV.name.
ensureUniqueDirectory		true	true	Used when dynamic provisioning is enabled. When set to true, appends UID to the pattern specified in subPathPattern to ensure access points don't accidentally point to the same directory.**Note**: Only set to false if you're certain this is the desired behavior.
provisioningMode	efs-ap		false	EFS volume type, currently supports access points.
fileSystemId			false	File system ID of the created access point.

- EBS Block Storage Available Parameters

  Note: For performance parameters of different volume types, see Amazon EBS Volume Types ↗.

Parameter	Optional Values	Default Value	Description
"allowAutoIOPSPerGBIncrease"	true, false	false	When set to "true", the CSI driver increases volume IOPS when iopsPerGB * <volume size> is too low to meet AWS supported IOPS range. This ensures dynamic provisioning always succeeds even when user-specified PVC capacity or iopsPerGB values are too small, but may incur additional costs as such volumes have higher IOPS than required by iopsPerGB.
"blockExpress"	true, false	false	Creates io2 Block Express volumes by raising IOPS limits for io2 volumes to 256000, but volumes created with IOPS exceeding 64000 cannot be mounted on instances that don't support io2 Block Express.
"blockSize"			Block size used when formatting the underlying filesystem. Only applies

Parameter	Optional Values	Default Value	Description
			to Linux nodes with ext2, ext3, ext4, or xfs filesystem types.
"bytesPerINode"			Bytes per inode used when formatting the underlying filesystem. Only applies to Linux nodes with ext2, ext3, or ext4 filesystem types.
"csi.storage.k8s.io/fstype"	xfs, ext2, ext3, ext4	ext4	Filesystem type to format when creating volumes. Case-sensitive.
"encrypted"	true, false	false	Whether the volume needs encryption.
"inodeSize"			Inode size used when formatting the underlying filesystem. Only applies to Linux nodes with ext2, ext3, ext4, or xfs filesystem types. Inodes are data structures in filesystems that store file and directory metadata.
"iops"			I/O operations per second, applicable to IO1, IO2, and GP3 volumes.
"iopsPerGB"			I/O operations per GiB per second, applicable to

Parameter	Optional Values	Default Value	Description
			IO1, IO2, and GP3 volumes.
"kmsKeyId"			Full ARN of the key to use for encrypting volumes. If not specified, AWS uses the default KMS key for the volume's region and automatically generates a key named /aws/ebs.
"numberOfINodes"			Number of inodes specified when formatting the underlying filesystem. Only applies to Linux nodes with ext2, ext3, or ext4 filesystem types.
"throughput"		125	Throughput in MiB/s. Only valid when specifying gp3 volume type. If empty, defaults to 125 MiB/s. See Amazon EBS Volume Types↗.
"type"	io1, io2, gp2, gp3, sc1, st1, standard, sbp1, sbg1	gp3	EBS volume type.

Huawei Cloud CCE Cluster Storage Initialization Configuration

Platform integration with Huawei Cloud CCE and storage initialization configuration.

TOC

Constraints and Limitations

Cluster PVC quantity has limits, and account storage capacity has quotas. You can request increases through support tickets.

Prerequisites

- If you have created a CCE cluster, import the CCE cluster (Public Cloud).

Configuration Steps

1. Go to **Platform Management** and click **Storage Management** > **Storage Classes** in the left navigation.
2. Click the three dots next to the storage class named **csi-nas** or **csi-disk** > **Update Project**. **Note**: After importing a CCE cluster, default storage classes are generated. **csi-disk** is recommended for block storage, and **csi-nas** for file storage. See Default Storage Class Description.
3. Select the **Project Assignment** method as needed and click **Update** to assign **csi-nas** or **csi-disk** storage classes to projects.

Default Storage Class Description

Storage Class Name	Storage Class Type	Description
(Recommended) csi-disk	Block Storage	Recommended for use.
(Recommended) csi-nas	File Storage	Recommended for use. Only available in regions that support csi-nas service.
csi-disk-topology	Block Storage	Automatic cloud disk topology when nodes span AZs.
csi-local	Local Storage	
csi-local-topology	Local Storage	
csi-obs	Object Storage	
csi-sfsturbo	Ultra-fast File Storage	Ultra-fast file storage cannot dynamically create persistent volumes.

Common Issues

PVC Creation Failure

The following error occurs because the PVC quantity limit has been reached. You can request an increase through support tickets:

```
message: "ShareLimitExceeded: Maximum number of shares allowed (10) excee
ded."
```

Account Overdue

PVC creation fails with the following error due to overdue payments. Please pay promptly:

```
message: "Your account is suspended and resources cannot be used"
```

Azure AKS Cluster Storage Initialization Configuration

Platform integration with Azure AKS and storage initialization configuration.

TOC

Constraints and Limitations

The default azurefile file storage class may not support permission modifications after mounting, which may cause some applications like PostgreSQL and Jenkins to fail to run properly.

Prerequisites

- If you have created an AKS cluster, import the Azure AKS cluster.

Configuration Steps

1. Go to **Platform Management** and click **Storage Management** > **Storage Classes** in the left navigation.
2. Click the three dots next to the storage class named **default** or **azurefile** > **Update Project**.
   Note: After importing an AKS cluster, the following default storage classes are generated. **default** is recommended for block storage, and **azurefile** for file storage. See Default Storage Class Description.
3. Select the **Project Assignment** method as needed and click **Update** to assign **default** or **azurefile** storage classes to projects.
   Note: If the default storage classes don't meet requirements, create new storage classes following the above steps and modify parameters as needed. See Available Storage Class Parameters.

Related Information

Default Storage Class Description

Storage Class Name	Storage Class Type	Description
(Recommended) azurefile	File Storage	Creates Azure file shares using Azure standard storage.
(Recommended) default	Block Storage	Creates managed disks using Azure StandardSSD storage.
azurefile-csi	File Storage	Creates Azure file shares using Azure standard storage.
azurefile-csi-nfs	File Storage	Creates Azure file shares using Azure standard storage, NFS protocol.

Storage Class Name	Storage Class Type	Description
azurefile-csi-premium	File Storage	Creates Azure file shares using Azure premium storage.
azurefile-premium	File Storage	Creates Azure file shares using Azure premium storage.
managed	Block Storage	Creates managed disks using Azure StandardSSD storage.
managed-csi	Block Storage	Creates managed disks using Azure StandardSSD locally redundant storage (LRS).
managed-csi-premium	Block Storage	Creates managed disks using Azure premium locally redundant storage (LRS).
managed-premium	Block Storage	Creates managed disks using Azure premium storage.

Available Storage Class Parameters

- For block storage optional parameters and meanings, see Create and use volumes with Azure disks in Azure Kubernetes Service (AKS)↗.
- For file storage optional parameters and meanings, see Create and use volumes with Azure Files in Azure Kubernetes Service (AKS)↗.

Google GKE Cluster Storage Initialization Configuration

Platform integration with Google GKE and storage initialization configuration.

TOC

Constraints and Limitations

- The default file storage type PVC has a minimum capacity of 1T. If the capacity is set to less than 1T during creation, it will automatically expand to 1T.
- Default file storage has capacity limits. You can request expansion through support tickets.

- Default file storage creation and deletion operations take considerable time. If it remains in creating status for a long time, please be patient.

Prerequisites

- When creating clusters, on the Google Cloud Platform **Cluster > Features** page under the **Other** section, check **Enable Compute Engine Persistent Disk CSI Driver** and **Enable Filestore CSI Driver** options.
- Enable **Cloud Filestore API** and **Google Kubernetes Engine API** on Google Cloud Platform. See Access Filestore instances using the Filestore CSI driver↗.
- Adjust regional file storage quotas on Google Cloud Platform. See Resource quotas and limits↗.
- If you have created a GKE cluster, import the GKE cluster.

Configuration Steps

1. Go to **Platform Management** and click **Storage Management** > **Storage Classes** in the left navigation.
2. Click the three dots next to the storage class named **standard-rwx** or **standard-rwo** > **Update Project**.
   Note: After importing a GKE cluster, default storage classes are generated. **standard-rwx** is recommended for file storage, and **standard-rwo** for block storage. See Default Storage Class Description.
3. Select the **Project Assignment** method as needed and click **Update** to assign **standard-rwx** or **standard-rwo** storage classes to projects.
   Note: If the default storage classes don't meet requirements, create new storage classes following the above steps and modify parameters as needed. See Available Storage Class Parameters.

Related Information

Default Storage Class Description

Storage Class Name	Storage Class Type	Description
(Recommended) standard-rwx	File Storage	Uses Basic HDD Filestore service tier ↗.
(Recommended) standard-rwo	Block Storage	Uses balanced persistent disks.
premium-rwx	File Storage	Uses Basic SSD Filestore service tier ↗.
premium-rwo	Block Storage	SSD persistent disks.
enterprise-rwx	File Storage	Uses Enterprise Filestore tier ↗.
enterprise-multishare-rwx	File Storage	Uses Enterprise Filestore tier ↗. See Filestore multishares for Google Kubernetes Engine ↗.

Available Storage Class Parameters

- For block storage optional parameters and meanings, see Storage options ↗.
- For file storage optional parameters and meanings, see Service tiers ↗.

Common Issues

File Storage Type Storage Class PVC Creation Failure

- The following error occurs because Cloud Filestore API is not enabled in the project or lacks appropriate permissions. See Prerequisites to resolve:

```
failed to provision volume with StorageClass "standard-rwx": rpc error:
code = PermissionDenied desc = googlespi: Error 403: Cloud Filestore AP
I has not been used in project alauda-proj-1234 before or it is disable
d.
...
resion: SERVICE_DISABLED
```

- The following error occurs due to exceeding storage quotas. See Prerequisites to resolve:

```
failed to provision volume with StorageClass "standard-rwx": rpc error:
code = ResourceExhausted desc = googlespi: Error 429: Quora limit 'Stan
dardStorageGbPerRegion' has been exceeded. Limit 2048 in region asia-ea
st1.
 rateLimitExceeded
```

Block Storage Type Storage Class PVC Cannot Bind Properly

The following error occurs because the node's CSINode lacks configuration for the pd.csi.storage.gke.io driver. You can resolve this by restarting the **Compute Engine Persistent Disk CSI Driver**.

Note: Updating this plugin will make the cluster unavailable. The update process takes approximately 5-10 minutes.

```
Warning ProvisioningFailed 18m (x14 over 39m) pd.csi.storage.gke.io_gke-5
cb9bddae4d1430eb8ad-01f4-2084-vm_4b4e70bd-e2db-4779-9102-fee83a657ced fai
led to provision volume with StorageClass "standard": error generating ac
cessibility requirements: no available topology found
Normal ExternalProvisioning 4m35s (x143 over 39m) persistentvolume-contro
ller waiting for a volume to be created, either by external provisioner
"pd.csi.storage.gke.io" or manually created by system administrator
Normal Provisioning 3m19s (x18 over 39m) pd.csi.storage.gke.io_gke-5cb9bd
dae4d1430eb8ad-01f4-2084-vm_4b4e70bd-e2db-4779-9102-fee83a657ced External
provisioner is provisioning volume for claim "acp-gke-test/standard"
```

How to

Network Configuration for Impc Fetch import cluster informatio Trust an ins

Collect Network Data from Custom Named Network Cards

Collect network data from custom named network cards

Network Configuration for Import Clusters

TOC

Scenario Description

Before cluster import, only unidirectional connectivity is ensured, allowing the global cluster to access the imported cluster. After cluster import, to ensure the imported cluster can properly access the global cluster and achieve bidirectional connectivity, additional network configuration is required to ensure normal operation of platform functional components.

Prerequisites

Please prepare in advance the **domain name**, **IP address** that the domain name points to, and the corresponding **valid certificate** that the imported cluster can access.

Note:

- This domain name must not be the same as the **platform access address**.
- Ensure that the domain's port (HTTPS port, which is the same port as the platform access address) can forward traffic to all control nodes of the global cluster.

Adding Annotation Information for Imported Clusters

Specifically, to ensure that alerting and log collection components can properly access the platform, you must add annotation information to the imported cluster.

1. In the left navigation bar, click **Cluster Management** > **Clusters**.
2. Click **global**.
3. Click **Operations** > **CLI Tools**, and replace the relevant parameters using the following command:

   ```
   kubectl annotate --overwrite -n cpaas-system clusters.cluster.x-k8s.io
   <imported cluster name> \
           cpaas.io/platform-url=<prepared domain address, e.g.: https://w
   ww.domain.cn>
   ```

   Code example:

   ```
   kubectl annotate --overwrite -n cpaas-system clusters.cluster.x-k8s.io
   cluster-imported \
           cpaas.io/platform-url=https://www.domain.cn
   ```

How to fetch import cluster information?

TOC

Problem description

Obtain the configuration required to connect to the import cluster so that the platform can be authorized to access and manage it. This section provides the steps to retrieve the import cluster information.

Prerequisites

- A working `kubectl` environment. **For public cloud clusters, it is strongly recommended to use the provider's CloudShell.** If CloudShell is not available, Linux or macOS is recommended.
- You have obtained the import cluster's KubeConfig file and copied it to the environment where `kubectl` is installed. **To avoid operating on the wrong environment, it is strongly recommended to use one of the following non-destructive approaches:**

- **Backup approach**: Copy your existing kubeconfig to a safe location first: `cp "${HOME}/.kube/config" "${HOME}/.kube/config.backup"`
- **Isolated approach**: Set the `KUBECONFIG` environment variable to point to the imported kubeconfig: `export KUBECONFIG="/path/to/imported/kubeconfig"`
- **Merge approach**: Use kubectl's merge/flatten without losing existing contexts:
  1. `export KUBECONFIG="/path/to/imported/kubeconfig:${HOME}/.kube/config"`
  2. `kubectl config view --flatten > "${HOME}/.kube/merged.kubeconfig"`
  3. `export KUBECONFIG="${HOME}/.kube/merged.kubeconfig"`

Get cluster information

Get cluster token

1. Run the following commands:

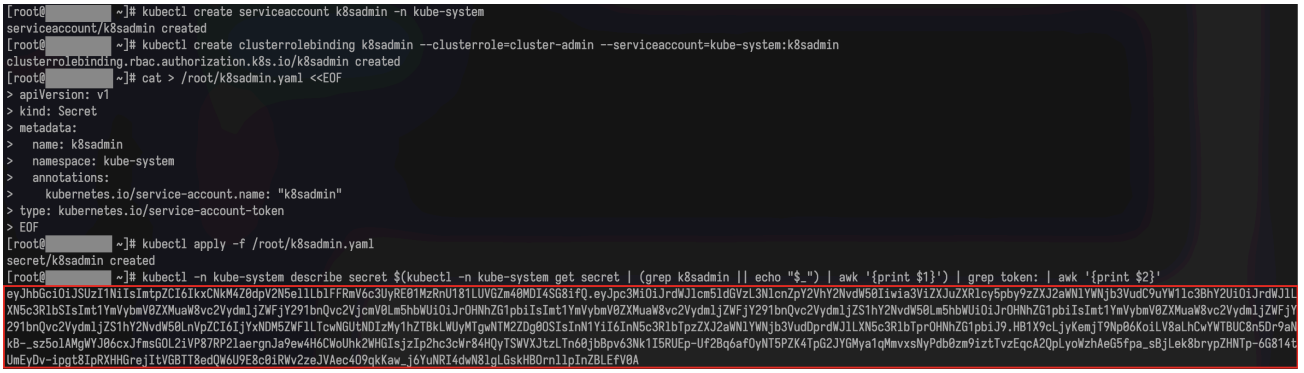
```
# [Important] The following operations support bash only

# Manually create a secret, bind a service account, and generate a non-expiring token
kubectl get ns cpaas-system > /dev/null 2>&1 || kubectl create namespace cpaas-system
kubectl create serviceaccount k8sadmin -n cpaas-system
kubectl create clusterrolebinding k8sadmin --clusterrole=cluster-admin --serviceaccount=cpaas-system:k8sadmin

cat | kubectl apply -f - <<EOF
apiVersion: v1
kind: Secret
metadata:
  name: k8sadmin
  namespace: cpaas-system
  annotations:
    kubernetes.io/service-account.name: "k8sadmin"
type: kubernetes.io/service-account-token
EOF

kubectl -n cpaas-system describe secret \
    $(kubectl -n cpaas-system get secret | (grep k8sadmin || echo "$_") | awk '{print $1}') \
    | grep -F 'token:' | awk '{print $2}'
```

WARNING

This procedure creates a cluster-admin credential intended to be non-expiring.

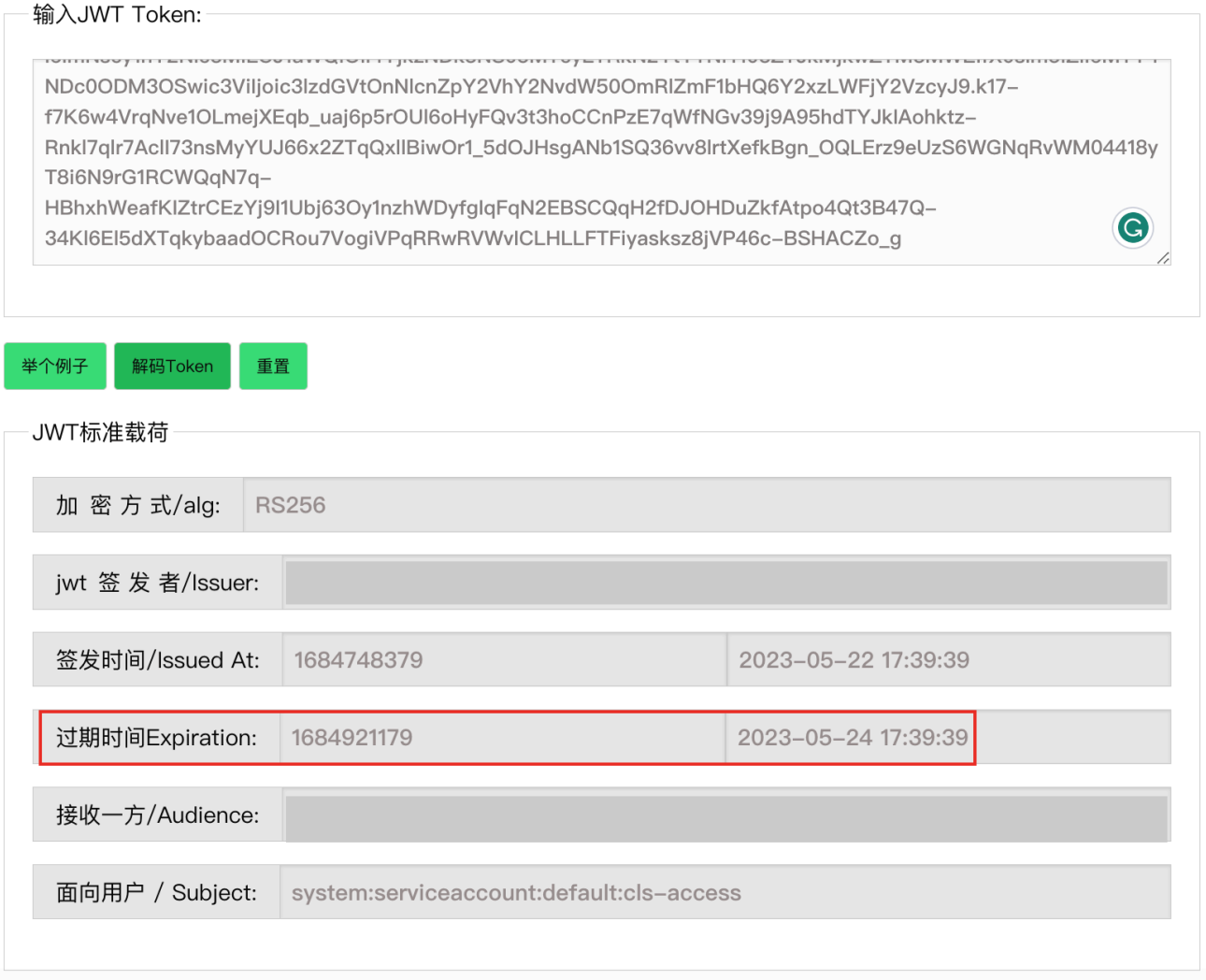
- Prefer least-privilege RBAC scoped to required resources if possible.
- Store the token securely; rotate/revoke when no longer needed.
- Limit who can read `Secret` objects in `cpaas-system`.

2. An example of the token obtained in the previous step is shown below.



3. Validate the token expiration.

   Use any tool that supports parsing JWT tokens to analyze the token and confirm its expiration time. If you can find an expiration field in the parsed result (a key containing "exp", as shown below), the platform will be unable to manage the import cluster after that time. In this case, stop and contact technical support.



TIP

The expiration is recorded as `"exp": 1684486916,` in the original JWT payload. The value is a UNIX timestamp and can be converted to UTC time.

Cleanup (revoke access) when done:

```
kubectl delete clusterrolebinding k8sadmin
kubectl -n cpaas-system delete secret k8sadmin
kubectl -n cpaas-system delete serviceaccount k8sadmin
```

Get the import cluster API server address and CA certificate

> **TIP**
>
> If you have already obtained the API server address and CA certificate using the platform's `Parse KubeConfig File` feature on the import cluster page, skip this step.

1. Run the following commands:

```
# View the import cluster API server addresses. There may be multiple a
ddresses; choose the one that fits your environment.
kubectl --kubeconfig "${KUBECONFIG:-${HOME}/.kube/config}" config view
--show-managed-fields=false --flatten --raw -ojsonpath='{$.clusters..cl
uster.server}'
addr_apiserver='<Selected API server address>'

# Get the CA certificate for the API server specified above
kubectl --kubeconfig "${KUBECONFIG:-${HOME}/.kube/config}" config view
--show-managed-fields=false --flatten --raw \
    -ojsonpath="{$.clusters[?(@.cluster.server == '${addr_apiserve
r}')].cluster.certificate-authority-data}" \
    | { base64 -d 2>/dev/null || base64 -D; }
```

How to trust an insecure image registry?

TOC

Problem description

The image registry hosting platform component images may not provide HTTPS service or may not have a valid TLS certificate issued by a public certificate authority. If you trust this registry, configure your container runtime by following the steps below.

Configure trust for an insecure image registry

Notes:

- All nodes that need to use images, including newly added nodes, must be configured and have Containerd restarted.
- The configuration differs slightly between Containerd v1.4/v1.5 and v1.6. Follow the appropriate steps for your version.

1. Run the following on every node in the import cluster:

    - Back up the configuration file

```
mkdir -p '/var/backup-containerd-confs/'
if ! [ -f /etc/containerd/config.toml ]; then
    echo 'Containerd config not found. Please check if containerd is
correctly installed. If you still cannot resolve the issue, contact t
echnical support.'
    exit 1
else
    cp /etc/containerd/config.toml /var/backup-containerd-confs/confi
g.toml_$(date +%F_%T)
fi
```

- Get the Containerd runtime version

```
# Get the containerd version
# Compare this version to v1.6. Choose steps accordingly
ctr --version | grep -Eo 'v[0-9]+\.[0-9]+\.[0-9]+'
```

Containerd v1.4 v1.5 configuration for insecure registries

2. Run the following on every node in the import cluster:

- Edit `/etc/containerd/config.toml`

```
# Example content to add to the config file
# Lines in brackets are sections. If the file already has sections wi
th the same name, merge their contents.
[plugins."io.containerd.grpc.v1.cri".registry]
    [plugins."io.containerd.grpc.v1.cri".registry.mirrors]
        [plugins."io.containerd.grpc.v1.cri".registry.mirrors."<regis
try-address>"]
            endpoint = ["https://<registry-address>", "http://<regist
ry-address>"]
        [plugins."io.containerd.grpc.v1.cri".registry.mirrors."192.16
8.134.43"]
            endpoint = ["https://192.168.134.43", "http://192.168.13
4.43"]
    [plugins."io.containerd.grpc.v1.cri".registry.configs]
        [plugins."io.containerd.grpc.v1.cri".registry.configs."<regis
try-address>".tls]
            insecure_skip_verify = true
        [plugins."io.containerd.grpc.v1.cri".registry.configs."192.16
8.134.43".tls]
            insecure_skip_verify = true
```

- Restart Containerd.

```
systemctl daemon-reload && systemctl restart containerd
```

Containerd v1.6 configuration for insecure registries

2. Run the following on every node in the import cluster:

- Check whether `config_path` exists in the config.

```
if ! grep -qF 'config_path' /etc/containerd/config.toml; then
    if grep -qE '\[plugins."io.containerd.grpc.v1.cri".registry.(mirr
ors|configs)(\.|\])' /etc/containerd/config.toml; then
        echo 'Follow the steps in "Containerd v1.4 v1.5 configuration
for insecure registries".'
    else
        cat >> /etc/containerd/config.toml << 'EOF'
[plugins."io.containerd.grpc.v1.cri".registry]
    config_path = "/etc/containerd/certs.d/"
EOF
    fi
fi

config_path_var=$(grep -F '/etc/containerd/certs.d' /etc/containerd/c
onfig.toml)
if [ -z "$config_path_var" ]; then
   echo 'The value of config_path in the file is unexpected. Please c
heck!'
   exit 1
fi
```

- Create the `hosts.toml` file.

  If the previous command printed `Follow the steps in "Containerd v1.4 v1.5 configuration for insecure registries".`, see *Containerd v1.4 v1.5 configuration for insecure registries*.

```
REGISTRY='<registry address obtained in the "Get the registry addres
s" section>'

mkdir -p "/etc/containerd/certs.d/$REGISTRY/"
cat > "/etc/containerd/certs.d/$REGISTRY/hosts.toml" << EOF
server = "$REGISTRY"
[host."http://$REGISTRY"]
  capabilities = ["pull", "resolve", "push"]
  skip_verify = true
[host."https://$REGISTRY"]
  capabilities = ["pull", "resolve", "push"]
  skip_verify = true
EOF
```

- Restart Containerd.

```
systemctl daemon-reload && systemctl restart containerd
```

Collect Network Data from Custom Named Network Cards

TOC

Scenario Description

After creating a workload cluster, the platform monitoring can only recognize network card names matching patterns like `eth.*|en.*|wl.*|ww.*` by default. For user-defined network card names, network traffic data cannot be viewed on the monitoring page. To address this, the platform supports modifying relevant resource parameters to manually capture network card traffic data.

Procedure

1. Log in to the control node of the global cluster and execute the following commands using kubectl.
2. First, find the moduleinfo resource name corresponding to the workload cluster in the global cluster:

```
kubectl get moduleinfo | grep -E 'prometheus|victoriametrics'
```

Example output:

```
global-6448ef7f7e5e3924c1629fad826372e7     global     prometheus
prometheus                          Running   v3.15.0-zz231204040711-9d
1fc12474c2   v3.15.0-zz231204040711-9d1fc12474c2   v3.15.0-zz2312040407
11-9d1fc12474c2
ovn-0954f21f0359720e8c115804376b3e7e        ovn        prometheus
prometheus                          Running   v3.15.0-zz231204040711-9d
1fc12474c2   v3.15.0-zz231204040711-9d1fc12474c2   v3.15.0-zz2312040407
11-9d1fc12474c2
```

3. Edit the moduleinfo resource of the workload cluster, replacing `ovn-0954f21f0359720e8c115804376b3e7e` with the workload cluster moduleinfo resource name from the previous step:

```
kubectl edit moduleinfo ovn-0954f21f0359720e8c115804376b3e7e
```

4. Add the valuesOverride field and modify the field and regular expression according to the comment information:

```
spec:
  valuesOverride: # If this field does not exist, you need to add the v
aluesOverride field and the following parameters under spec
    ait/chart-cpaas-monitor:
      ovn: # Replace with the workload cluster name
        indicator:
          networkDevice: eth.*|em.*|en.*|wl.*|ww.*|[A-Z].*i|custom_inte
rface # Replace custom_interface with a custom regular expression to en
sure correct network card name matching
```

5. Wait 10 minutes, then check the network-related charts on the node monitoring page to ensure the changes take effect.

Creating an On-Premise Cluster

TOC

Prerequisites

Node Requirements

1. If you downloaded a single-architecture installation package from Download Installation Package, ensure your node machines have the same architecture as the package. Otherwise, nodes won't start due to missing architecture-specific images.
2. Verify that your node operating system and kernel are supported. See Supported OS and Kernels for details.
3. Perform availability checks on node machines. For specific check items, refer to Node Preprocessing > Node Checks.
4. If node machine IPs cannot be directly accessed via SSH, provide a SOCKS5 proxy for the nodes. The `global` cluster will access nodes through this proxy service.

Load Balancing

For production environments, a load balancer is required for cluster control plane nodes to ensure high availability.

If a hardware LoadBalancer is available in your environment, it is **recommended** to use it. If not, you can enable `Self-built VIP`, which provides software load balancing using keepalived.

Recommendation: When using a hardware LoadBalancer, configuring the Cluster Endpoint with a domain name is recommended. If the hardware LoadBalancer VIP changes after cluster installation, you can update the DNS record instead of reconfiguring the cluster.

Note: `Self-built VIP` does not support domain name configuration.

The load balancer must correctly forward traffic to ports `6443`, `11780`, and `11781` on all control plane nodes in the cluster.

If your cluster has only one control plane node and you use that node's IP as the Cluster Endpoint parameter, the cluster cannot be scaled from a single node to a highly available multi-node setup later. Therefore, we recommend providing a load balancer even for single-node clusters.

When enabling `Self-built VIP`, you need to prepare:

1. An available VRID
2. A host network that supports the VRRP protocol

3. All control plane nodes and the VIP must be on the same subnet, and the VIP must be different from any node IP.

Connecting `global` Cluster and Workload Cluster

The platform requires mutual access between the `global` cluster and workload clusters. If they're not on the same network, you need to:

1. Provide `External Access` for the workload cluster to ensure the `global` cluster can access it. Network requirements must ensure `global` can access ports `6443`, `11780`, and `11781` on all control plane nodes.
2. Add an additional address to `global` that the workload cluster can access. When creating a workload cluster, add this address to the cluster's annotations with the key `cpaas.io/platform-url` and the value set to the public access address of `global`.

Image Registry

Cluster images support Platform Built-in, Private Repository, and Public Repository options.

- **Platform Built-in**: Uses the image registry provided by the `global` cluster. If the cluster cannot access `global`, see Add External Address for Built-in Registry.
- **Private Repository**: Uses your own image registry. For details on pushing required images to your registry, contact technical support.
- **Public Repository**: Uses the platform's public image registry. Before using, complete Updating Public Repository Credentials.

Container Networking

If you plan to use Kube-OVN's Underlay for your cluster, refer to Preparing Kube-OVN Underlay Physical Network.

Creation Procedure

1. Enter the **Administrator** view, and click **Clusters/Clusters** in the left navigation bar.

2. Click **Create Cluster**.
3. Configure the following sections according to the instructions below: Basic Info, Container Network, Node Settings, and Extended Parameters.

Basic Info

Parameter	Description
Kubernetes Version	All optional versions are rigorously tested for stability and compatibility. **Recommendation:** Choose the latest version for optimal features and support.
Container Runtime	Containerd is provided as the default container runtime.
Cluster Network Protocol	Supports three modes: IPv4 single stack, IPv6 single stack, IPv4/IPv6 dual stack. **Note:** If you select dual stack mode, ensure all nodes have correctly configured IPv6 addresses; the network protocol cannot be changed after setting.
Cluster Endpoint	`IP Address / Domain`: Enter the pre-prepared domain name or VIP if no domain name is available. `Self-built VIP`: Disabled by default. Only enable if you haven't provided a LoadBalancer. When enabled, the installer will automatically

deploy `keepalived` for software load balancing support.

`External Access`: Enter the externally accessible address prepared for the cluster when it's not in the same network environment as the `global` cluster.

Container Network

Kube-OVN

An enterprise-grade Cloud Native Kubernetes container network orchestration system developed by Alauda. It brings mature networking capabilities from the OpenStack domain to Kubernetes, supporting cross-cloud network management, traditional network architecture and infrastructure interconnection, and edge cluster deployment scenarios, while greatly enhancing Kubernetes container network security, management efficiency, and performance.

Parameter	Description
Subnet	Also known as Cluster CIDR, represents the **default subnet** segment. After cluster creation, additional subnets can be added.
Transmit Mode	**Overlay**: A virtual network abstracted over the infrastructure that doesn't consume physical network resources. When creating an Overlay default subnet, all Overlay subnets in the cluster use the same cluster NIC and node NIC configuration. **Underlay**: This transmission method relies on physical network devices. It can directly allocate physical network addresses to Pods, ensuring better performance and connectivity with the physical network. Nodes in an Underlay subnet must have multiple NICs, and the NIC used for bridge networking must be exclusively used by Underlay and not carry other traffic like SSH. When creating an Underlay default subnet, the cluster NIC is

	actually a default NIC for bridge networking, and the node NIC is the node NIC configuration in the bridge network. • **Default Gateway**: The physical network gateway address, which is the gateway address for the Cluster CIDR segment (must be within the Cluster CIDR address range). • **VLAN ID**: Virtual LAN identifier (VLAN number), e.g., `0`. • **Reserved IPs**: Set reserved IPs that won't be automatically allocated, such as IPs in the subnet that are already used by other devices.
Service CIDR	IP address range used by Kubernetes Services of type ClusterIP. Cannot overlap with the default subnet range.
Join CIDR	In Overlay transmission mode, this is the IP address range used for communication between nodes and pods. Cannot overlap with the default subnet or Service CIDR.

Calico

Calico is a layer 3 networking solution that provides secure network connections for containers.

Parameter	Description
Default Subnet	Also known as Cluster CIDR, represents the **default subnet** segment. After cluster creation, additional subnets can be added.
Service CIDR	IP address range used by Kubernetes Services of type ClusterIP. Cannot overlap with the default subnet range.

Flannel

Flannel provides a flat network environment for all containers in the cluster, giving containers created on different node hosts a unique virtual IP address across the entire cluster. The pod subnet is divided evenly among the cluster nodes according to the mask, and pods on each node are assigned IP addresses from the segment allocated to that node. This improves communication efficiency between containers without having to consider IP translation issues.

Parameter	Description
Cluster CIDR	IP address range used by pods created when the cluster starts. Supports setting the maximum number of IP addresses that can be allocated to pods on each node under the current container network. **Note**: The platform will automatically calculate the maximum number of nodes the cluster can accommodate based on the above configuration and display it in the hint below the input field. **Important**: After cluster creation, the cluster network cannot be changed, so please plan the network carefully.
Service CIDR	IP address range used by Kubernetes Services of type ClusterIP. Cannot overlap with the container subnet range.

Custom

If you need to install other network plugins, select **Custom** mode. You can manually install network plugins after the cluster is successfully created.

Parameter	Description
Cluster CIDR	IP address range used by pods created when the cluster starts.
Service CIDR	IP address range used by Kubernetes Services of type ClusterIP. Cannot overlap with the container subnet range.

Node Settings

Parameter	Description
Network Interface Card	The name of the host network interface device used by the cluster network plugin. **Note**: • When selecting Underlay transmission mode for the Kube-OVN default subnet, you must specify the network interface name, which will be the default NIC for bridge networking. - The platform's network interface traffic monitoring by default recognizes traffic on interfaces named like `eth.|en.|wl.|ww.`. If you use interfaces with different naming conventions, please refer to Collect Network Data from Custom-Named Network Interfaces after cluster onboarding to modify the relevant resources and ensure the platform can properly monitor network interface traffic.
Node Name	You can choose to use either the node IP or hostname as the node name on the platform.

	Note: When choosing to use hostname as the node name, ensure that the hostnames of nodes added to the cluster are unique.
Nodes	**Add nodes** to the cluster, or **Recovery from draft** temporarily saved node information. See the detailed parameter descriptions for adding nodes below.
Monitoring Type	Supports **Prometheus** and **VictoriaMetrics**. When selecting **VictoriaMetrics** as the monitoring component, you must configure the **Deploy Type**: **- Deploy VictoriaMetrics**: Deploys all related components, including **VMStorage**, **VMAlert**, **VMAgent**, etc. **- Deploy VictoriaMetrics Agent**: Only deploys the log collection component, **VMAgent**. When using this deployment method, you need to associate with a VictoriaMetrics instance already deployed on another cluster in the platform to provide monitoring services for the cluster.
Monitoring Nodes	Select nodes for deploying cluster monitoring components. Supports selecting compute nodes and control plane nodes that allow application deployment. To avoid affecting cluster performance, it's recommended to prioritize compute nodes. After the cluster is successfully created, monitoring components with storage type **Local Volume** will be deployed on the selected nodes.

Node Addition Parameters

Parameter	Description

Type	**Control Plane Node**: Responsible for running components such as kube-apiserver, kube-scheduler, kube-controller-manager, etcd, container network, and some platform management components in the cluster. When **Application Deployable** is enabled, control plane nodes can also be used as compute nodes. **Worker Node**: Responsible for hosting business pods running on the cluster.
IPv4 Address	The IPv4 address of the node. For clusters created in internal network mode, enter the node's **private IP**.
IPv6 Address	Valid when the cluster has IPv4/IPv6 dual stack enabled. The IPv6 address of the node.
Application Deployable	Valid when **Node Type** is **Control Plane Node**. Whether to allow business applications to be deployed on this control plane node, scheduling business-related pods to this node.
Display Name	The display name of the node.

SSH Connection IP	The IP address that can connect to the node when accessing it via SSH service. If you can log in to the node using `ssh <username>@<node's IPv4 address>`, this parameter is not required; otherwise, enter the node's public IP or NAT external IP to ensure the `global` cluster and proxy can connect to the node via this IP.
Network Interface Card	Enter the name of the network interface used by the node. The priority of network interface configuration effectiveness is as follows (from left to right, in descending order): **Kube-OVN Underlay**: Node NIC > Cluster NIC **Kube-OVN Overlay**: Node NIC > Cluster NIC > NIC corresponding to the node's default route **Calico**: Cluster NIC > NIC corresponding to the node's default route **Flannel**: Cluster NIC > NIC corresponding to the node's default route
Associated Bridge Network	**Note**: When creating a cluster, bridge network configuration is not supported; this option is only available when **adding nodes** to a cluster that already has Underlay subnets created.

	Select an existing Add Bridge Network. If you don't want to use the bridge network's default NIC, you can configure the node NIC separately.
SSH Port	SSH service port number, e.g., `22`.
SSH Username	SSH username, needs to be a user with root privileges, e.g., `root`.
Proxy	Whether to access the node's SSH port through a proxy. When the `global` cluster cannot directly access the node to be added via SSH (e.g., the `global` cluster and workload cluster are not in the same subnet; the node IP is an internal IP that the `global` cluster cannot directly access), this switch needs to be turned on and proxy-related parameters configured. After configuring the proxy, node access and deployment can be achieved through the proxy. **Note**: Currently, only SOCKS5 proxy is supported. **Access URL**: Proxy server address, e.g., `192.168.1.1:1080`. **Username**: Username for accessing the proxy server. **Password**: Password for accessing the proxy server.
SSH Authentication	Authentication method and corresponding authentication information for logging into the added node. Options include:

	Password: Requires a username with root privileges and the corresponding **SSH password**. **Key**: Requires a **private key** with root privileges and the **private key password**.
Save Draft	Saves the currently configured data in the dialog as a draft and closes the **Add Node** dialog. Without leaving the **Create Cluster** page, you can select **Restore from draft** to open the **Add Node** dialog and restore the configuration data saved as a draft. **Note**: The data restored from the draft is the most recently saved draft data.

Extended Parameters

Note:

- Apart from required configurations, it's not recommended to set extended parameters, as incorrect settings may make the cluster unavailable and cannot be modified after cluster creation.
- If a entered **Key** duplicates a default parameter **Key**, it will override the default configuration.

Procedure

1. Click **Extended Parameters** to expand the extended parameter configuration area. You can optionally set the following extended parameters for the cluster:

Parameter	Description

	`kubeletExtraArgs`, additional configuration parameters for Kubelet.
Kubelet Parameters	**Note**: When the **Container Network**'s **Node IP Count** parameter is entered, a default **Kubelet Parameter** configuration with the key `max-pods` and a value of **Node IP Count** is automatically generated. This sets the maximum number of pods that can run on any node in the cluster. This configuration is not displayed in the interface. Adding a new `max-pods: maximum number of runnable pods` key-value pair in the **Kubelet Parameters** area will override the default value. Any positive integer is allowed, but it's recommended to use the default value (Node IP Count) or enter a value not exceeding `256`.
Controller Manager Parameters	`controllerManagerExtraArgs`, additional configuration parameters for the Controller Manager.
Scheduler Parameters	`schedulerExtraArgs`, additional configuration parameters for the Scheduler.
APIServer Parameters	`apiServerExtraArgs`, additional configuration parameters for the APIServer.
APIServer URL	`publicAlternativeNames`, APIServer access addresses issued in the certificate. Only IPs or domain names can be entered, with a maximum of 253 characters.

Cluster Annotations	Cluster annotation information, marking cluster characteristics in metadata in the form of key-value pairs for platform components or business components to obtain relevant information.

4. Click **Create**. You'll return to the cluster list page where the cluster will be in the **Creating** state.

Post-Creation Steps

Viewing Creation Progress

On the cluster list page, you can view the list of created clusters. For clusters in the **Creating** state, you can check the execution progress.

Procedure

1. Click the small icon **View Execution Progress** to the right of the cluster status.
2. In the execution progress dialog that appears, you can view the cluster's execution progress (status.conditions).
   Tip: When a certain type is in progress or in a failed state with a reason, hover your cursor over the corresponding reason (shown in blue text) to view detailed information about the reason (status.conditions.reason).

Associating with Projects

After the cluster is created, you can add it to projects in the project management view.

About Hosted Control Plane

Hosted Control Plane (HCP) is a lightweight multi-cluster management model that separates the control plane from worker nodes. Each cluster's control plane is containerized and hosted within a management cluster, reducing resource consumption, accelerating cluster creation and upgrades, and improving scalability for multi-cluster operations.

> **Note**
>
> Because Hosted Control Plane releases on a different cadence from Alauda Container Platform, the Hosted Control Plane documentation is now available as a separate documentation set at Hosted Control Plane ↗.

Cluster Node Planning

A cluster utilizes the Kubernetes node role labels `node-role.kubernetes.io/<role>` to assign different roles to nodes. For convenience of description, we refer to this type of label as a role label.

By default, a cluster contains two types of nodes: control plane nodes and worker nodes, used to host control plane workloads and application workloads, respectively.

In a cluster:

- The control plane nodes are labeled with the role label `node-role.kubernetes.io/control-plane`.

  > **Note:**
  > Prior to Kubernetes v1.24, the community also used the label `node-role.kubernetes.io/master` to mark control plane nodes. For backward compatibility, both labels are considered valid for identifying control plane nodes.

- The worker nodes, by default, have no role labels. However, you can explicitly assign the role label `node-role.kubernetes.io/worker` to a worker node if desired.

In addition to these default role labels, you can also define custom role labels on worker nodes to further classify them into different functional types. For example:

- You can add the role label `node-role.kubernetes.io/infra` to designate a node as an infra node, intended for hosting infrastructure components.
- You can add the role label `node-role.kubernetes.io/log` to designate a node as a log node, specialized for hosting logging components.

This document will guide you through creating infra nodes and custom role nodes, and migrating workloads to those nodes.

TOC

Creating Infra Nodes on Non-Immutable Cluster

By default, a cluster only includes control plane nodes and worker nodes. If you want to designate certain worker nodes as infra nodes dedicated to hosting infrastructure components, you need to manually add the appropriate role label and taint to those nodes.

> **Note:**
> The operations in this section are only applicable to non-immutable clusters. That is, the following operations are not supported on cloud clusters (such as EKS managed clusters deployed via the `Alauda Container Platform EKS Provider` Cluster Plugin), third-party clusters, or clusters where the nodes use an immutable OS.

Adding Infra Nodes

Step 1: Add the Infra Role Label to the Node resources

```
kubectl label nodes 192.168.143.133 node-role.kubernetes.io/infra="" --ov
erwrite
```

This command adds the infra role label to the Node 192.168.143.133: `node-role.kubernetes.io/infra: ""`, indicating that the node is an infra node.

Step 2: Add a Taint to the Node resources

Add a taint to prevent other workloads from being scheduled onto the infra node.

```
kubectl taint nodes 192.168.143.133 node-role.kubernetes.io/infra=reserve
d:NoSchedule
```

This command adds the taint `node-role.kubernetes.io/infra=reserved:NoSchedule` to Node 192.168.143.133, indicating that only applications that tolerate this taint can be scheduled onto this node.

Step 3: Verify the Label and Taint

Check whether the node has been assigned the infra role label and taint:

```
# kubectl describe node 192.168.143.133
Name:               192.168.143.133
Roles:              infra
Labels:             node-role.kubernetes.io/infra=""
                    ...
Taints:             node-role.kubernetes.io/infra=reserved:NoSchedule
```

The output indicates that the Node 192.168.143.133 has been configured as an infra node and has been tainted with tainted with `node-role.kubernetes.io/infra=reserved:NoSchedule`.

Migrating Pods to Infra Nodes

If you want to schedule specific Pod onto infra nodes, you need to make the following configurations:

- A nodeSelector targeting the infra role label.
- Corresponding tolerations for the infra node's taint.

Below is an example Deployment manifest configured to run on the infra node.

```yaml
apiVersion: apps/v1
kind: Pod
metadata:
  name: infra-pod-demo
  namespace: default
spec:
  ...
  nodeSelector:
    node-role.kubernetes.io/infra: ""
  tolerations:
  - effect: NoSchedule
    key: node-role.kubernetes.io/infra
    value: reserved
    operator: Equal
  ...
```

The nodeSelector ensures the Pod is only scheduled on nodes with the label `node-role.kubernetes.io/infra: ""`, the toleration allows the Pod to tolerate the taint node-role.kubernetes.io/infra=reserved:NoSchedule .

With these configurations, the Pod will be scheduled onto the infra node.

> **Note:**
> Moving pods installed via OLM Operators or Cluster Plugins to an infra node is not always possible. The capability to move these pods is depends on the configuration of each Operator or Cluster Plugin.

Custom Node Planning

Beyond infra nodes, you may want to designate worker nodes for other specialized purposes — such as hosting logging components, storage services, or monitoring agents.

You can achieve this by assigning more custom role labels and corresponding taints to worker nodes, effectively turning them into custom role nodes.

General Steps for Defining Custom Role Nodes

The process is similar to creating infra nodes.

Step 1: Add a Custom Role Label

```
kubectl label nodes <node> node-role.kubernetes.io/<role>="" --overwrite
```

Replace <role> with your desired role name, such as monitoring, storage, or log.

Step 2: Add a Corresponding Taint

```
kubectl taint nodes <node> node-role.kubernetes.io/<role>=<value>:NoSchedule
```

Replace <role> with your custom role name and replace <value> with a meaningful descriptor, such as reserved or dedicated. This value is optional but useful for documentation and clarity.

Step 3: Verify the Configuration

```
kubectl describe node <node>
```

Ensure the Labels and Taints fields reflect your custom role configuration.

Example: Create A Node Dedicated To Logging Components

If you want to create a node specifically for installing logging components, you can add the log role. In this case, create the log node as follows.

Step 1: Add the Log Role Label

```
kubectl label nodes 192.168.143.133 node-role.kubernetes.io/log="" --over
write
```

This label indicates that the node is designated for log-related workloads.

Step 2: Add a Taint to the Node

```
kubectl taint nodes 192.168.143.133 node-role.kubernetes.io/log=reserved:
NoSchedule
```

This taint prevents unscheduled workloads from being deployed to the node.

Step 3: Verify the Label and Taint

```
Name:               192.168.143.133
Roles:              log
Labels:             node-role.kubernetes.io/log=reserved
                    ...
Taints:             node-role.kubernetes.io/log=reserved:NoSchedule
```

This confirms that the node has been successfully configured as a log node with the appropriate label and taint.

By following the above practices, you can effectively partition your Kubernetes nodes based on their intended purpose, improve workload isolation, and ensure that specific components are deployed onto appropriately configured nodes.

etcd Encryption

This guide helps you install, understand, and operate the etcd Encryption Manager in ACP to automate etcd data encryption key rotation within your clusters.

It ensures that sensitive data stored in etcd, such as secrets and configmaps, is encrypted using a secure algorithm, enhancing your cluster's security.

TOC

Installation

See Cluster Plugin for installation instructions.

> **Note:**
>
> - Currently supported:
>   - On-Premises clusters

> - DCS clusters
> - Not supported:
>   - `global cluster`

How it Works

Upon installation, an `etcd-encryption-manager` controller is deployed in the `kube-system` namespace, which:

- Periodically rotates etcd data encryption keys.
- Retains the 8 most recent keys for rollback compatibility.
- Updates encryption configurations on all control nodes.
- Triggers `kube-apiserver` to hot reload new keys.
- Automatically migrates resources to re-encrypt data with new keys.

Cluster stability is maintained throughout these operations.

Default Configuration

Parameter	Value
Encrypted resources	secrets, configmaps
Encryption algorithm	256-bit AES-GCM
Rotation interval	168 hours (7 days)

Operations Guide

Configuration Files

Path	Content
`/etc/kubernetes/encryption-provider.conf`	Current encryption configuration
`/etc/kubernetes/encryption-provider-history.bak`	Historical key records (for recovery)
`/etc/kubernetes/encryption-provider-bak/`	Expired encryption configuration versions

Checking Status

Run the following command to check the current rotation status:

```
kubectl get EtcdEncryptionConfig default -o yaml
```

Example output:

```yaml
apiVersion: cluster.alauda.io/v1alpha1
kind: EtcdEncryptionConfig
metadata:
  name: default
spec:
  resources:
    - secrets
    - configmaps
  rotationInterval: 168h0m0s
  type: aesgcm
status:
  deployStatus:
    192.168.100.1:
      revision: 3
      state: Success
    192.168.100.2:
      revision: 3
      state: Success
    192.168.100.3:
      revision: 3
      state: Success
  migration:
    completeTimestamp: "2025-05-27T05:47:01Z"
    resources:
      - secrets
      - configmaps
    revision: 3
    state: Success
  revision: 3
```

How to

Add External Address for Built-
Optimize Pod Performance with
Updating Pu

Add External Address for Built-in Registry

TOC

Overview

When the `global` cluster uses the `Platform Built-in` registry, workload clusters typically also use this registry to pull images. The registry not only serves components within the `global` cluster but must also be accessible to workload cluster nodes.

In certain scenarios, workload cluster nodes cannot directly access the `global` cluster's registry address - for example, when the `global` cluster is in a private data center while workload clusters are in public clouds or edge environments.

This guide explains how to configure an externally accessible address for the platform's default registry to allow workload clusters to pull images.

Prerequisites

Before you begin, prepare the following:

- A domain name accessible by workload cluster nodes
- The IP address that the domain name points to
- A valid SSL certificate for the domain name

> **WARNING**
>
> - The domain name must be different from the platform access address
> - Ensure the domain's IP address can forward traffic to all control plane nodes of the `global` cluster

Procedure

Configure Certificate and Routing Rules for the Platform Registry

1. Copy the domain's valid certificate to any control plane node of the `global` cluster
2. Create a TLS secret containing the domain certificate:

   ```
   kubectl create secret tls registry-address.tls --cert=<certificate-file
   name> --key=<key-filename> -n kube-system
   ```

   Example:

   ```
   kubectl create secret tls registry-address.tls --cert=custom.crt --key=
   custom.key -n kube-system
   ```

   Note: After creating the certificate, monitor the expiration date of the **registry-address.tls** secret in the **kube-system** namespace of the `global` cluster. Replace the certificate before it expires.
3. Create ingress rules on any control plane node of the `global` cluster:

```
REGISTRY_DOMAIN_NAME=<www.registry.com> # Replace with your accessible
domain name
cat << EOF | kubectl create -f -
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  annotations:
    nginx.ingress.kubernetes.io/backend-protocol: HTTPS
  name: registry-address
  namespace: kube-system
  labels:
    service_name: registry
spec:
  rules:
    - host: $REGISTRY_DOMAIN_NAME
      http:
        paths:
          - backend:
              service:
                name: registry
                port:
                  number: 443
            path: /v2/
            pathType: ImplementationSpecific
          - backend:
              service:
                name: registry
                port:
                  number: 443
            path: /v2/_catalog
            pathType: ImplementationSpecific
          - backend:
              service:
                name: registry
                port:
                  number: 443
            path: /v2/.+/tags/list
            pathType: ImplementationSpecific
          - backend:
              service:
                name: registry
                port:
                  number: 443
```

```
            path: /v2/.+/manifests/[A-Za-z0-9_+.-:]+
            pathType: ImplementationSpecific
          - backend:
              service:
                name: registry
                port:
                  number: 443
            path: /v2/.+/blobls/[A-Za-z0-9-:]+
            pathType: ImplementationSpecific
          - backend:
              service:
                name: registry
                port:
                  number: 443
            path: /v2/.+/blobls/uploads/[A-Za-z0-9-:]+
            pathType: ImplementationSpecific
          - backend:
              service:
                name: registry
                port:
                  number: 443
            path: /auth/token
            pathType: ImplementationSpecific
  tls:
    - secretName: registry-address.tls
      hosts:
        - $REGISTRY_DOMAIN_NAME
EOF
```

A response similar to `... created` indicates successful ingress creation.

4. Check if a Registry Service resource exists:

```
kubectl -n kube-system get svc | grep registry
```

If the Service doesn't exist, create it with:

```
cat << EOF | kubectl create -f -
apiVersion: v1
kind: Service
metadata:
  labels:
    name: registry
    service_name: registry
  name: registry
  namespace: kube-system
spec:
  ports:
    - protocol: TCP
      port: 443
      targetPort: 60080
  selector:
    component: registry
  type: ClusterIP
EOF
```

5. Test the configuration by pulling an image from the registry using the domain name:

```
crictl pull <registry-domain-name>/automation/qaimages:helloworld
```

Or

```
podman pull <registry-domain-name>/automation/qaimages:helloworld
```

Optimize Pod Performance with Manager Policies

This guide provides Kubernetes cluster administrators with a practical, ready-to-apply manual for enabling and validating **CPU ManagerPolicy**, **Memory ManagerPolicy**, and **Topology ManagerPolicy**. By aligning CPU pinning, NUMA affinity, and topology alignment, you can deliver consistent latency and improved performance for critical workloads.

TOC

Scope and Prerequisites

Roles and Permissions

- Requires maintenance window access, `kubectl` admin privileges, and SSH access to nodes.

Workload Requirements

- To achieve dedicated CPU and NUMA affinity, Pods must run in **Guaranteed QoS** class: requests = limits and CPU specified in full cores (e.g., 2, 4).

Not Covered

- HugePages are out of scope. If you need HugePages support, contact your support team.

Quick Start: Sample Kubelet Config

Add the following snippet to `/var/lib/kubelet/config.yaml`, adjusting values for your environment:

```
# ── CPU ManagerPolicy ──
cpuManagerPolicy: "static"              # Options: none | static
cpuManagerPolicyOptions:
  full-pcpus-only: "true"               # Recommended: allocate only full
cores
cpuManagerReconcilePeriod: "5s"
reservedSystemCPUs: ""                  # e.g. "0-1" if reserving specifi
c CPUs for the system

# ── Memory ManagerPolicy ──
memoryManagerPolicy: "Static"           # Options: none | Static
reservedMemory:
  - numaNode: 0
    limits:
      memory: "2048Mi"
  - numaNode: 1
    limits:
      memory: "2048Mi"

# ── Topology ManagerPolicy ──
topologyManagerPolicy: "single-numa-node"     # Options: none | best-effo
rt | restricted | single-numa-node
topologyManagerScope: "pod"                   # Options: container | pod
```

Notes:

- `full-pcpus-only: "true"` improves latency consistency.
- `topologyManagerScope: pod` ensures containers within the same Pod align to a common NUMA topology.
- `reservedMemory` must be calculated based on kubelet config and eviction thresholds (see next section).

How to Calculate `reservedMemory`

Formula:

```
R_total = kubeReserved(memory) + systemReserved(memory) + evictionHard(memory.available)
```

The sum of `reservedMemory` across all NUMA nodes must equal `R_total`.

Steps (for N NUMA nodes):

1. Calculate `R_total` (Mi).
2. Compute division and remainder:
   - base = floor(R_total / N)
   - rem = R_total − base × N
3. Assign values:
   - NUMA node 0 = base + rem
   - Remaining NUMA nodes = base

Example (2 NUMA nodes):

- kubeReserved=512Mi, systemReserved=512Mi, evictionHard=100Mi → R_total = 1124Mi
- base = 562, rem = 0

```
reservedMemory:
- numaNode: 0
  limits:
    memory: "562Mi"
- numaNode: 1
  limits:
    memory: "562Mi"
```

Applying the Configuration

For each node:

1. **Cordon and Drain**

```
kubectl cordon <node>
kubectl drain <node> --ignore-daemonsets --delete-emptydir-data
```

2. **Stop Kubelet and Clear State**

```
sudo systemctl stop kubelet
sudo rm -f /var/lib/kubelet/cpu_manager_state
sudo rm -f /var/lib/kubelet/memory_manager_state
```

3. **Restart Kubelet**

```
sudo systemctl daemon-reload
sudo systemctl start kubelet
```

4. **Reschedule Pods**

```
kubectl uncordon <node>
```

- For DaemonSets and system Pods, restart or delete Pods explicitly.

5. **Verify Recovery**

```
kubectl get nodes
kubectl get pods -A -o wide | grep <node>
```

Verification

CPU ManagerPolicy State

```
sudo cat /var/lib/kubelet/cpu_manager_state | jq .
```

Check:

- `.policyName` = `"static"`
- `.defaultCpuSet` lists non-dedicated CPUs
- `.entries` show container-to-CPU assignments

Memory ManagerPolicy State

```
sudo cat /var/lib/kubelet/memory_manager_state | jq .
```

Check:

- `.policyName` = `"Static"`
- Sum of reserved memory matches `R_total`
- Guaranteed Pods are assigned to NUMA nodes per `single-numa-node` policy

Key Policies and Behaviors

CPU ManagerPolicy

- Purpose: Allocate exclusive physical CPUs to Guaranteed Pods
- Config: `cpuManagerPolicy: static`, `full-pcpus-only: "true"`
- Behavior: Only applies to Guaranteed Pods; Burstable/BestEffort are unaffected

Memory ManagerPolicy

- Purpose: Reserve and align memory at NUMA node level
- Config: `memoryManagerPolicy: "Static"`, `reservedMemory`
- Behavior: Works best with Topology ManagerPolicy for alignment

Topology ManagerPolicy

- Purpose: Align CPU, memory, and device allocation on a single NUMA node
- Config: `topologyManagerPolicy: single-numa-node`, `topologyManagerScope: pod`
- Modes: best-effort, restricted, single-numa-node (strict)

Terminology

- **NUMA node**: Non-Uniform Memory Access domain
- **CPU pinning**: Binding containers to dedicated CPUs
- **NUMA affinity**: Preferring memory from the same NUMA node as CPU
- **Topology alignment**: Co-locating CPU, memory, and devices on one NUMA node
- **Guaranteed Pod**: requests = limits; CPU specified as full cores

Updating Public Repository Credentials

TOC

Overview

The `Public Repository` is a platform-provided image registry service available on the public internet. When you want your clusters to use the `Public Repository` as their image registry, you need to update the built-in `public-registry-credential` Cloud Credentials. This ensures your platform has permission to pull images from the public registry.

Procedure

1. Log in to the **Alauda Customer Portal** and download your organization's authentication file from the **Enterprise Management** section located in the **User Information** dropdown in the upper right corner.
2. Navigate to **Clusters** > **Cloud Credential** in the left navigation bar of the **Administrator** console.
3. Locate the cloud credential named `public-registry-credential` and click **Update** from the dropdown menu on the right.

4. In the **Upload Public Repository Address** section, upload the authentication file you downloaded from the **Alauda Customer Portal**.
5. Click **Update** to apply the changes.