Clusters

Overview

Overview

Immutable Infrastructure

Immutable Infrastructure

Managed Clusters

Managed Clusters

Import Third-Party Clusters

Register Cl

Public Cloud Cluster Initialization

Public cloud cluster initialization.

How to

Creating an On-Premise Cluster

Creating an On-Premise Cluster

How to

Add External Address for Built-

Updating Public Repository Cre

Access a Cl

Overview

Start here to choose whether to create a platform-managed cluster, evaluate Hosted Control Plane, or onboard an existing Kubernetes environment. For the platform-level relationship between cluster responsibility, control-plane topology, and onboarding boundaries, see Cluster Management Models.

TOC

- Choose A Cluster Path
- Platform-Managed Cluster Creation
- Installer-Provisioned Infrastructure
- User-Provisioned Infrastructure
- Hosted Control Plane
- Third-Party Cluster Onboarding
- Version Compatibility
  - Upgrade Prerequisite Behavior
  - ACP 4.2 And Earlier

Choose A Cluster Path

If you need to	Start with	Key boundary
Let the platform provision machines and manage Immutable OS for supported providers.	About Immutable Infrastructure	Platform-owned lifecycle applies only within the supported provider and workflow scope.
Create a cluster on machines that you prepare and maintain.	Creating an On-Premise Cluster	The platform manages Kubernetes after node preparation; node OS lifecycle remains user-owned.
Evaluate Hosted Control Plane (HCP).	About Hosted Control Plane	Check the HCP documentation for current maturity, operating system, connectivity, and production-use boundaries.
Onboard an existing Kubernetes environment.	Third-Party Cluster Onboarding	The external owner, distribution, or provider usually owns Kubernetes, node, and infrastructure lifecycle.
Choose between direct platform access and reverse-connect onboarding.	Import Third-Party Clusters and Register Cluster	Import and register are onboarding methods, not separate day-2 capability models.

Platform-Managed Cluster Creation

For clusters that ACP creates and lifecycle-manages, choose the creation path by infrastructure responsibility.

Infrastructure responsibility	Use when	Continue with
Installer-Provisioned Infrastructure (IPI)	The platform should provision machines and manage node operating systems through Immutable OS for a supported provider integration.	About Immutable Infrastructure
User-Provisioned Infrastructure (UPI)	You prepare physical or virtual machines, and the platform installs and manages Kubernetes	Creating an On-Premise Cluster

Infrastructure responsibility	Use when	Continue with
	on those nodes.	

Installer-Provisioned Infrastructure

In Installer-Provisioned Infrastructure (IPI), the platform provisions machines and manages node operating systems through Immutable OS. This model is used when the platform owns the supported cluster provisioning, scaling, and upgrade lifecycle for the selected provider integration.

Currently, the platform supports Alauda OS for immutable node management. For immutable infrastructure concepts and provider-specific workflows, see About Immutable Infrastructure.

User-Provisioned Infrastructure

In User-Provisioned Infrastructure, users prepare physical or virtual machines before cluster creation. The platform installs and manages Kubernetes on those nodes, while node operating system management, including provisioning, patching, and replacement, remains under the user's control.

This model is suitable when organizations already have established infrastructure or operating system management procedures. For the cluster creation workflow, see Creating an On-Premise Cluster.

To create and manage User-Provisioned Infrastructure clusters through APIs, see Immutable Infrastructure API Reference ↗ and go to **Provider APIs > User-Provisioned Infrastructure Provider APIs**.

Hosted Control Plane

Hosted Control Plane is a control-plane topology. Each hosted cluster has its own control plane, and multiple hosted control planes run as workloads on a management cluster.

In ACP, HCP is implemented through Kamaji (`TenantControlPlane`). For current support scope and evaluation details, see About Hosted Control Plane.

Third-Party Cluster Onboarding

Third-party clusters are existing Kubernetes environments provided outside Alauda. They can include standard Kubernetes environments, external Kubernetes distributions, or public cloud Kubernetes services. ACP can provide centralized governance and operations within documented prerequisites and caveats, but onboarding does not make ACP the owner of the external cluster's Kubernetes lifecycle, node lifecycle, or provider infrastructure lifecycle.

Third-party clusters are onboarded through import or register workflows:

Onboarding method	Connection model	Use when
Import cluster	The `global` cluster connects to the target cluster API server with supplied address, CA, and credentials.	The platform can reach the target cluster API server and the operator can provide the required cluster information.
Register cluster	A reverse proxy service in the target cluster initiates registration and establishes a tunnel to the platform.	The target cluster should initiate the connection, or direct platform access to the target API server is restricted.

Provider-specific caveats can apply to certificates, audit data, control-plane metrics, ingress, storage, node operations, connectivity, and Extension compatibility. Use the import and register workflow pages for detailed prerequisites.

For onboarding workflows, see Third-Party Cluster Onboarding, Import Third-Party Clusters, and Register Cluster.

Version Compatibility

Use the Kubernetes Support Matrix to choose platform-managed cluster creation targets, verify workload-cluster upgrade prerequisites, and validate third-party cluster onboarding versions. These are separate ranges for separate decisions.

Upgrade Prerequisite Behavior

- For ACP 4.3 and later, workload clusters only need to remain within the Compatible Versions before the `global` cluster upgrade.
- The third-party onboarding range is independent from the Compatible Versions used for this upgrade prerequisite.

ACP 4.2 And Earlier

- Upgrade workload clusters to the latest documented compatible Kubernetes version before upgrading the `global` cluster.
- Use the Kubernetes Support Matrix as the numerical authority for the documented version mapping.

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.

ACP supports Immutable Infrastructure on the following providers: Huawei DCS, VMware vSphere, and Huawei Cloud Stack. Bare-metal support is planned.

> **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 ↗.

TOC

Tasks

API Reference

Tasks

The following scenarios cover Immutable Infrastructure across the cluster lifecycle.

Scenario	Documentation
Install the `global` cluster	Installing the global Cluster on Immutable Infrastructure ↗
Upgrade the `global` cluster	Upgrading the global Cluster on Immutable Infrastructure ↗
Plan disaster recovery for the `global` cluster	Global Cluster Disaster Recovery on Immutable Infrastructure ↗
Install provider plugins	Installation ↗
Configure infrastructure resources	Infrastructure Resources ↗
Create workload clusters	Creating Clusters ↗
Upgrade workload clusters	Upgrading Clusters ↗
Manage cluster nodes	Managing Nodes ↗
Configure machines (OS or kernel level)	Machine Configuration ↗

API Reference

For Cluster API and provider CRD references, see API Reference ↗.

Managed Clusters

Managed Clusters

Managed Clusters

Import Third-Party Clusters

Import Third-Party Clusters

Import Standard Kubernetes Cl

Import Ope

Import GKE Cluster

Import Huawei Cloud CCE Cluster (Public Clo

Import an existing CCE cluster as a third-party cluster.

Import Azure AKS Cluster

Import Aliba

Import Tencent Cloud TKE Cluster

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 Cust

Collect network data from custom named r

Configure audit collection for imported standa

Enable Kubernetes audit logging on imported standard Kuberr collect audit data.

Third-Party Cluster Onboarding

Third-party clusters are existing Kubernetes environments whose Kubernetes distribution, node lifecycle, or provider infrastructure are managed outside Alauda. In the console, these environments appear under **Managed Clusters**.

ACP can provide centralized governance, resource visibility, Extension installation, application operations, and observability integration for third-party clusters when connectivity, credentials, component prerequisites, provider requirements, and Extension compatibility are satisfied. Onboarding does not make ACP the owner of the external cluster's Kubernetes lifecycle, node lifecycle, or provider infrastructure lifecycle.

TOC

Choose An Onboarding Method

Choose An Onboarding Method

Method	Connection model	Use when
Import cluster	The `global` cluster connects to the target cluster API server with supplied address, CA, and credentials.	The platform can reach the target API server, and the administrator can provide the required cluster information.

Method	Connection model	Use when
Register cluster	A reverse proxy service in the target cluster initiates registration and establishes a tunnel to the platform.	The target cluster should initiate the connection, or direct access from the `global` cluster to the target API server is restricted.

After onboarding, high-level day-2 operations are treated consistently in the cluster list. Provider-specific limitations can still apply to node operations, audit data, control-plane metrics, certificates, ingress, storage, and Extension compatibility.

For import procedures, see Import Third-Party Clusters. For reverse-connect onboarding, see Register Cluster.

Import Third-Party Clusters

Import Third-Party Clusters

Import Standard Kubernetes Cl

Import Ope

Import GKE Cluster

Import Huawei Cloud CCE Cluster (Public Clo

Import an existing CCE cluster as a third-party cluster.

Import Azure AKS Cluster

Import Aliba

Import Tencent Cloud TKE Cluster

Import Third-Party Clusters

Use import when the `global` cluster can reach the target cluster API server and the administrator can provide the required cluster address, CA, and credentials.

Target environment	Start with
Existing standard Kubernetes environment	Import Standard Kubernetes Cluster
External Kubernetes distribution	Import OpenShift Cluster
Public cloud Kubernetes service	Import Amazon EKS Cluster, Import GKE Cluster, Import Azure AKS Cluster, Import Alibaba Cloud ACK Cluster, Import Tencent Cloud TKE Cluster, or Import Huawei Cloud CCE Cluster

Provider-specific procedures include provider console or CLI steps where required. The provider name identifies the procedure path; it is not a separate ACP cluster model.

Import Standard Kubernetes Cluster

Use this procedure to import an existing standard Kubernetes cluster, such as a cluster deployed with **kubeadm**, as a third-party cluster.

TOC

- Terminology
- Prerequisites
- Notes
- Obtain Registry Address
- Check if Extra Registry Config is Needed
- Get Cluster Info
- Integrate Cluster
- Network Configuration
- Post-import Configuration
- FAQ
  - Why is the "Add Node" button disabled?
  - Which certificates are supported?
  - Which features are unsupported?
  - How to fix Containerd runtime causing distributed storage deployment failures?

Terminology

Term	Description
Provider-managed Kubernetes service	A Kubernetes service where the provider owns the control plane nodes and control plane components. Users usually cannot log in to or directly maintain the control plane nodes.
Self-managed Kubernetes cluster	A Kubernetes cluster where the cluster owner maintains the control plane nodes and cluster components.

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 Collect Network Data from Custom Named Network Cards.

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

See 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.

Post-import Configuration

If you need the platform to collect audit data from an imported standard Kubernetes cluster, configure Kubernetes API server audit logging on the cluster after import. See How to configure audit collection for imported standard Kubernetes clusters?.

FAQ

Why is the "Add Node" button disabled?

Adding nodes through the platform UI is not supported for imported standard Kubernetes clusters. Add nodes directly in the target cluster or through the cluster provider.

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?

- **Provider-managed Kubernetes services**: Audit logs are not available.
- **Provider-managed Kubernetes services**: ETCD, Scheduler, and Controller Manager monitoring are not supported. Only API Server metrics are 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

Use this provider-specific procedure to import an existing OpenShift cluster as a third-party cluster. The external distribution owner remains responsible for the Kubernetes distribution, node lifecycle, and provider infrastructure lifecycle unless a documented provider-specific capability provides that lifecycle.

TOC

- Prerequisites
- Obtain Registry Address
- Check if Extra Registry Config is Needed
  - Trust Insecure Registry
- Configure DNS for the Cluster
- Get Cluster Info
  - Method 1 (Recommended): Get the KubeConfig File
  - Method 2: Use Token, API Server Address, and CA Certificate
- Import Cluster
- Network Configuration
- Deploy Add-ons
- Update Audit Policy
- FAQ
  - Why is the "Add Node" button disabled?
  - Which certificates are supported?
  - Limitations for this provider-specific import path

Prerequisites

- The Kubernetes version and parameters of the cluster must meet the Standard Kubernetes Cluster Requirements.
- During integration, `kubectl` commands are required. 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 [ "$(kubectl get productbase -o jsonpath='{.items[].spec.registry.pr
eferPlatformURL}')" = 'false' ]; then
    REGISTRY=$(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 "Registry address is: $REGISTRY"
```

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

```
REGISTRY=<external-registry-address>  # e.g., registry.example.cn:60080
or 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 through the platform UI is not supported. Use the distribution provider'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.

Limitations for this provider-specific import path

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

Import Amazon EKS Cluster

Use this provider-specific procedure to import an existing Amazon EKS cluster as a third-party cluster. The external provider remains responsible for provider infrastructure and the provider-managed Kubernetes and node lifecycle unless a documented provider-specific capability provides that lifecycle.

TOC

- Prerequisites
- Prepare the environment
- Get cluster information
  - Get the import token
- Import the cluster
- Network configuration
- Next steps
  - Initialize Ingress and storage
- FAQ
  - The Add Node button is disabled after import. How can I add nodes?
  - Which certificates are supported by certificate management for imported clusters?
  - Limitations for this provider-specific import path

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.

To obtain the cluster import token, see How do I get cluster information?.

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. 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.

Limitations for this provider-specific import path

- 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

Use this provider-specific procedure to import an existing Google GKE cluster as a third-party cluster. The external provider remains responsible for provider infrastructure and the provider-managed Kubernetes and node lifecycle unless a documented provider-specific capability provides that lifecycle.

TOC

- Prerequisites
- Preparing the Operating Environment
- Obtaining Cluster Information
  - Obtaining the API Server Address and CA Certificate of the Target Cluster
  - Obtaining the Target Cluster Token
- Importing the Cluster
- Network Configuration
- Post-Import Operations
  - Ingress and Storage Initialization
- Frequently Asked Questions
  - How to add nodes when the "Add Node" button is grayed out after importing the cluster?
  - What certificates are supported by the certificate management functionality for imported clusters?

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.

See 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. Enter **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, update

Parameter	Description
	public repository cloud credentials to obtain repository authentication permissions.
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, see 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. 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)

Use this provider-specific procedure to import an existing Huawei Cloud CCE cluster as a third-party cluster. The external provider remains responsible for provider infrastructure and the provider-managed Kubernetes and node lifecycle unless a documented provider-specific capability provides that lifecycle.

TOC

- Prerequisites
- Obtain Image Registry Address
- Determine if Image Registry Requires Additional Configuration
- Obtain Cluster Information
  - Obtain Import Cluster Token
- Import Cluster
- Network Configuration
- Follow-up Operations
  - Ingress (Inbound Rules) and Storage Initialization
- FAQ
  - After importing the cluster, the add node button is grayed out. How to add nodes?
  - What certificates does the certificate management feature support for imported clusters?
  - Limitations for this provider-specific import path

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. See the "Trust Insecure Image Registry" sec
tion for configuration.'
fi
```

2. If the test fails, see 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.

See 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. 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, update public image registry cloud credentials to obtain registry authentication permissions.
Cluster Information	**Tip**: 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, see 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, see 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. 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.

Limitations for this provider-specific import path

- 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

Use this provider-specific procedure to import an existing Azure AKS cluster as a third-party cluster. The external provider remains responsible for provider infrastructure and the provider-managed Kubernetes and node lifecycle unless a documented provider-specific capability provides that lifecycle.

TOC

- Prerequisites
- Prepare the Operating Environment
- Obtain Cluster Information
  - Obtain Import Token
- Import Cluster
- Network Configuration
- Post-Import Operations
  - Ingress (Inbound Rules) and Storage Initialization
- Frequently Asked Questions
  - How to configure AKS node external IP security group rules
  - How to access AKS node
  - Azure ALB using internal load balancer
  - Azure ALB using external load balancer
  - The add node button is grayed out after importing the cluster. How to add nodes?
  - What certificates are supported by the certificate management feature for imported clusters?
  - Limitations for this provider-specific import path

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, see 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 Token

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

See 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. 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, update public image registry cloud credentials to obtain registry authentication permissions.
Cluster Information	**Tip**: 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, see 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

See the kubectl-node-shell project ↗.

Option 2: Using debug

See the provider guidance for connecting to AKS nodes ↗.

> **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

For internal load balancer behavior, see the provider guidance ↗.

```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.

```yaml
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. 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.

Limitations for this provider-specific import path

- 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

Use this provider-specific procedure to import existing Alibaba Cloud ACK managed clusters or dedicated clusters as third-party clusters. The external provider remains responsible for provider infrastructure and the provider-managed Kubernetes and node lifecycle unless a documented provider-specific capability provides that lifecycle.

TIP

For provider cluster type details, see the provider guide ↗.

TOC

- Prerequisites
- Get Image Registry Address
- Determine if Image Registry Requires Additional Configuration
- Get KubeConfig
- Import Cluster
- Network Configuration
- FAQ
  - How to handle port conflicts between Alibaba Cloud monitoring and platform monitoring components?
  - How to use public network access for Alibaba Cloud clusters?
  - After importing a cluster, the add node button is grayed out. How to add nodes?
  - Which certificates are supported by the certificate management function for imported clusters?
  - Limitations for this provider-specific import path

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.exa
mple.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. See the "Trust Insecure Image Registry" section for configuration.'
fi
```

2. If the test fails, see 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, 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. Add nodes in the provider backend or contact the cluster provider.

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.

Limitations for this provider-specific import path

- **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

Use this provider-specific procedure to import existing Tencent Cloud TKE dedicated clusters or managed clusters as third-party clusters. The external provider remains responsible for provider infrastructure and the provider-managed Kubernetes and node lifecycle unless a documented provider-specific capability provides that lifecycle.

TIP

For provider cluster type details, see the provider guide ↗.

TOC

Prerequisites

Obtain Image Registry Address

Determine if Image Registry Requires Additional Configuration

Obtain KubeConfig

Import Cluster

Network Configuration

FAQ

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

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

Limitations for this provider-specific import path

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. See the "Trust Unsafe Image Regist
ry" section for configuration.'
fi
```

2. If verification fails, see 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. Enter the **private image registry address**, **port**, **username**, and

Parameter	Description
	password for accessing the image registry. **- Public Registry**: Use image registry services located on the public network. Before use, 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. See 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. Add nodes in the provider backend or contact the cluster provider.

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.

Limitations for this provider-specific import path

- **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

Use register when the target third-party cluster should initiate the connection to the platform through a reverse proxy tunnel.

TOC

Prerequisites

Important Notes

Register Cluster

View Registration Command

FAQ

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

Prerequisites

- Depending on the target environment, the Kubernetes version and component parameters in the target cluster must meet the version and parameter requirements.
- The image registry must support HTTPS access and provide a valid TLS certificate authenticated by a public certification authority. If this requirement cannot be met, see 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

Network interface traffic monitoring recognizes network interfaces with names matching `eth\.\|en\.\|wl\.*\|ww\.*` by default. If the target cluster uses other naming conventions, see Collecting Network Data from Custom Named Network Cards and update the corresponding resources after registration.

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.

Parameter	Description
Public Registry	Pull required images through the public image registry provided by the platform. Ensure that the target cluster can access the public network. Before use, see 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. Obtain a new command after it expires.

View Registration Command

Find the cluster waiting for registration in the cluster list and click **View Registration Command**. Run the registration command before it expires.

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
Prerequisites
Configuration Steps
Deploy AWS Load Balancer Controller
Create Ingress and LoadBalancer Services
Related Operations
Test AWS CLI and eksctl Installation
Get ACCOUNT_ID
Kubeconfig Configuration File
Add Tags to Subnets
Create Certificate

Support Overview

Feature	Support Status	Requirements
LoadBalancer Service	Supported	Optionally deploy AWS Load Balancer Controller. Without this controller, LoadBalancer capabilities are limited.
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>` 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 the provider guide ↗.

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 the annotation reference ↗:

```
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

Important Notes

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

Service Annotation Configuration Instructions

Access Address Acquisition Method

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
Prerequisites
Configuration Steps
Create Ingress
Create LoadBalancer Service
Related Operations
Create Certificate

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 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,"bandwi` `kubernetes.io/elb.class: union`

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":"<E`
	`Name>","bandwidth_chargemode":"traffic","bandwidth_size":5,"bandwidth`
	`["<AZ A>","<AZ B>","<AZ C>"],"elb_virsubnet_ids":["<ELB Virtual Subne`
	`ID>"],"l7_flavor_name":"L7_flavor.elb.s1.small","l4_flavor_name":"L4_`
	`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
Prerequisites
Configuration Steps
Deploy Ingress Controller
Create Ingress and LoadBalancer Services
Related Operations
Create Certificate

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

Feature	Support Status	Requirements
		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
Prerequisites
Configuration Steps
Deploy Ingress Controller
Create Ingress and LoadBalancer Services
Related Operations
View Ingress Resources in Google Cloud
Create Certificate

Support Overview

Feature	Support Status	Requirements
LoadBalancer Service	Default Support	No additional deployment required.

Feature	Support Status	Requirements
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 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

Public Cloud Storage Initialization Overview

Use the storage initialization procedures after importing a supported public cloud Kubernetes service and before relying on persistent storage features in workloads. The external provider remains responsible for provider infrastructure and provider-managed storage backends; ACP configures and uses the storage classes exposed to the imported cluster.

The following tables summarize the default storage classes and operations documented for each provider-specific initialization path.

TOC

- Storage Class Support
  - AWS EKS Clusters
  - Huawei Cloud CCE Clusters
  - Azure AKS Clusters
  - 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	P S
File Storage	efs-sc	Supported	Supported	Not Supported	N
Block Storage	ebs-sc	Supported	Not Supported	Supported	N

Huawei Cloud CCE Clusters

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

Azure AKS Clusters

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

Google GKE Clusters

Storage Type	Default Storage Class	Create PVC with RWO Access Mode	Create PVC with RWX Access Mode	PVC Expansion	P S
File Storage	standard-rwx	Supported	Supported	Supported	N

Storage Type	Default Storage Class	Create PVC with RWO Access Mode	Create PVC with RWX Access Mode	PVC Expansion	P S
Block Storage	standard-rwo	Supported	Not Supported	Supported	N

AWS EKS Cluster Storage Initialization Configuration

Platform integration with AWS EKS and storage initialization configuration.

TOC

Constraints and Limitations
Prerequisites
Configuration Steps
Create Storage Classes
Modify Storage Class Project Assignment
Related Operations
Configure Available Storage Class Parameters

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 cr account mounting. I specified, u the mount t associated az for cross account mounting; i specified, randomly s a mount tar cross-accou mounting.
basePath			true	Path for cre dynamically provisioned access poir not specifie access poir created unc file system directory.
directoryPerms			false	Directory permissions

Parameter	Optional Values	Default Value	Optional	Descriptio
				creating Ac Point root directory ↗.
uid			true	POSIX use creating Ac Point root directory ↗.
gid			true	POSIX grou for creating Access Poi directory ↗.
gidRangeStart		50000	true	Starting ran POSIX grou to apply wh creating ac point root directory ↗. needed if ui are set.
gidRangeEnd		7000000	true	Ending rang POSIX grou Not needed uid/gid are s
subPathPattern			true	Template fo constructing subpaths w each acces created unc dynamic

Parameter	Optional Values	Default Value	Optional	Description
				provisioning located. Ca consist of fi strings and variables, s to the "subPathPa variable in r subdir-exte provisioner Optional parameters .PVC.name .PVC.name and .PV.nar
ensureUniqueDirectory		true	true	Used when dynamic provisioning enabled. W set to true, appends UI the pattern specified in subPathPat ensure acc points don't accidentally to the same directory.**No** Only set to you're certa is the desire behavior.

Parameter	Optional Values	Default Value	Optional	Description
provisioningMode	efs-ap		false	EFS volume currently su access poir
fileSystemId			false	File system the created access poir

- 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

Parameter	Optional Values	Default Value	Description
			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 to Linux nodes with ext2, ext3, ext4, or xfs filesystem types.
"bytesPerINode"			Bytes per inode used when formatting the underlying filesystem. Only

Parameter	Optional Values	Default Value	Description
			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.

Parameter	Optional Values	Default Value	Description
"iopsPerGB"			I/O operations per GiB per second, applicable to 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

Parameter	Optional Values	Default Value	Description
			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
Prerequisites
Configuration Steps
Default Storage Class Description
Common Issues
PVC Creation Failure
Account Overdue

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. Resolve the payment issue before retrying:

```
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
- Prerequisites
- Configuration Steps
- Related Information
  - Default Storage Class Description
  - Available Storage Class Parameters

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.

Storage Class Name	Storage Class Type	Description
azurefile-csi-nfs	File Storage	Creates Azure file shares using Azure standard storage, NFS protocol.
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
Prerequisites
Configuration Steps
Related Information
Default Storage Class Description
Available Storage Class Parameters
Common Issues
File Storage Type Storage Class PVC Creation Failure
Block Storage Type Storage Class PVC Cannot Bind Properly

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 the resource remains in `Creating` status for a long time, continue monitoring the operation.

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 Cust

Collect network data from custom named r

Configure audit collection for imported standa

Enable Kubernetes audit logging on imported standard Kuberr collect audit data.

Network Configuration for Imported Clusters

Before import, the required connectivity path lets the `global` cluster access the target cluster. After import, platform components in the imported cluster may also need to access the `global` cluster. Add the platform URL annotation when bidirectional connectivity is required for components such as alerting and log collection.

TOC

Prerequisites

Add The Platform URL Annotation

Prerequisites

Prepare the **domain name**, the **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.

Add The Platform URL Annotation

Add the annotation to the imported cluster resource.

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
Prerequisites
Get cluster information
Get cluster token
Get the import cluster API server address and CA certificate

Problem description

Obtain the configuration required to connect to the import cluster so the platform can be authorized to access and manage it.

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:

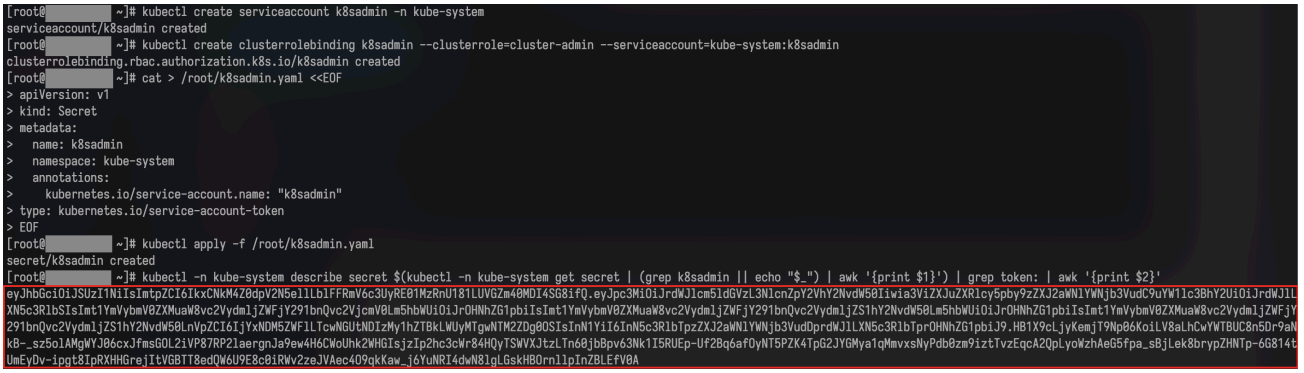
```
# [Important] The following operations support bash only

# 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.

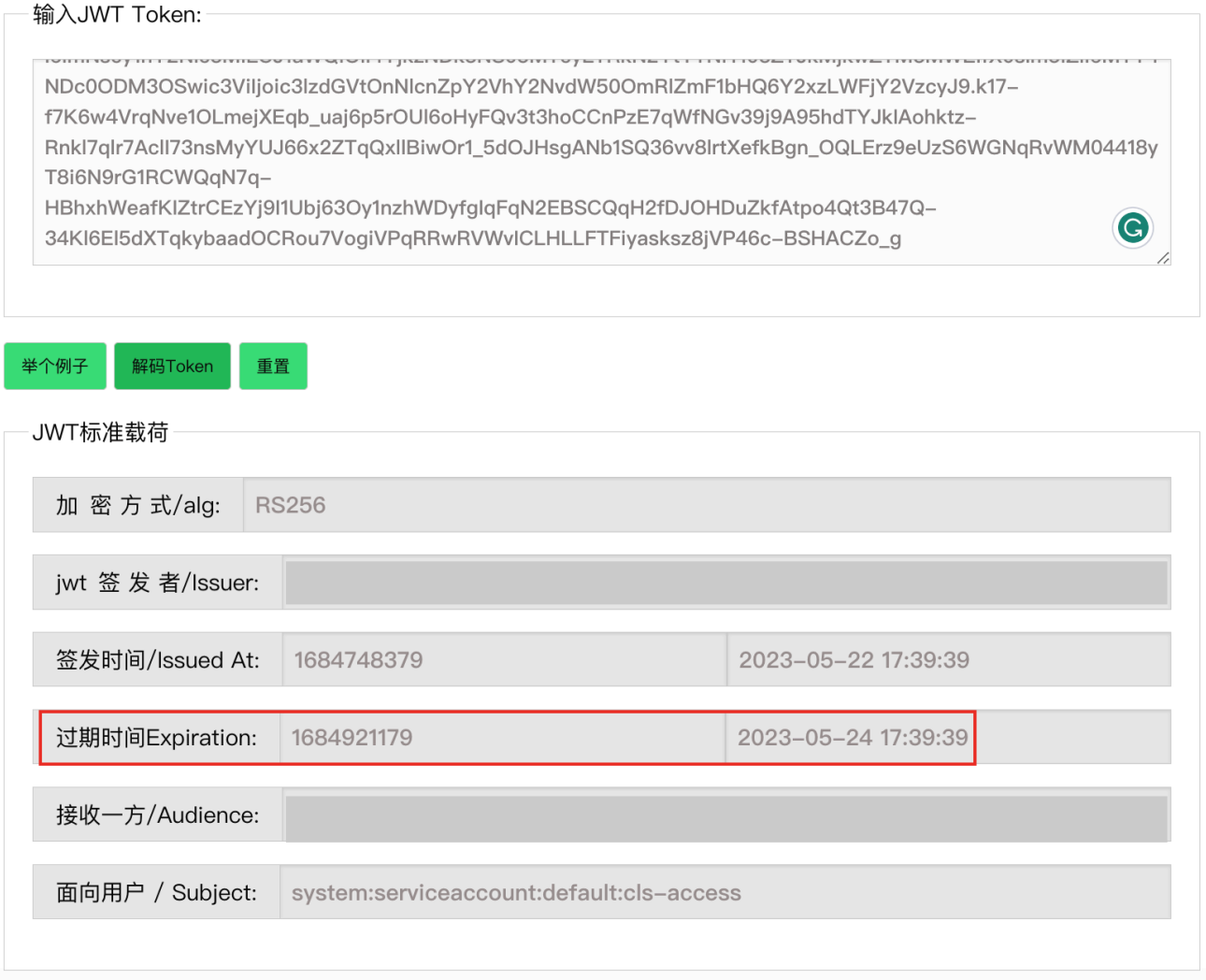
- Prefer least-privilege RBAC scoped to required resources if possible.
- Store the token securely; rotate/revoke when no longer needed.
- Limit who can read `Secret` objects in `cpaas-system`.

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

Configure trust for an insecure image registry

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

Procedure

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.

How to configure audit collection for imported standard Kubernetes clusters?

TOC

Scenario Description

Prerequisites

Procedure

Scenario Description

After a standard Kubernetes cluster is imported into the platform, you must enable Kubernetes API server audit logging on the cluster before the platform can collect audit data from that cluster.

Use this procedure for standard Kubernetes clusters whose control plane nodes are managed by you, such as kubeadm-based clusters. It does not apply to provider-managed cloud Kubernetes services where you cannot log in to or modify control plane nodes.

Prerequisites

- The standard Kubernetes cluster has already been imported into the platform.

- You can log in to every control plane node in the cluster.
- The cluster uses the standard kubeadm-style API server static Pod manifest path: `/etc/kubernetes/manifests/kube-apiserver.yaml`.

Procedure

1. Create a local `policy.yaml` file for the audit policy.
   Set `apiVersion` according to the Kubernetes version:

   - Kubernetes earlier than 1.24: `audit.k8s.io/v1beta1`
   - Kubernetes 1.24 and later: `audit.k8s.io/v1`

   Use the following content:

```
apiVersion: audit.k8s.io/v1
kind: Policy
omitStages:
  - "RequestReceived"
rules:
  - level: None
    users:
      - system:kube-controller-manager
      - system:kube-scheduler
      - system:serviceaccount:kube-system:endpoint-controller
    verbs: ["get", "update"]
    namespaces: ["kube-system"]
    resources:
      - group: ""
        resources: ["endpoints"]
  - level: None
    nonResourceURLs:
      - /healthz*
      - /version
      - /swagger*
  - level: None
    resources:
      - group: ""
        resources: ["events"]
  - level: None
    resources:
      - group: "devops.alauda.io"
  - level: None
    verbs: ["get", "list", "watch"]
  - level: None
    namespaces:
      - kube-system
      - cpaas-system
      - alauda-system
      - istio-system
      - kube-node-lease
    resources:
      - group: "coordination.k8s.io"
        resources: ["leases"]
  - level: None
    resources:
      - group: "authorization.k8s.io"
        resources: ["subjectaccessreviews", "selfsubjectaccessreviews"]
```

```
    - group: "authentication.k8s.io"
      resources: ["tokenreviews"]
  - level: Metadata
    resources:
    - group: ""
      resources: ["secrets", "configmaps"]
  - level: RequestResponse
    resources:
    - group: ""
    - group: "aiops.alauda.io"
    - group: "apps"
    - group: "app.k8s.io"
    - group: "authentication.istio.io"
    - group: "auth.alauda.io"
    - group: "autoscaling"
    - group: "asm.alauda.io"
    - group: "clusterregistry.k8s.io"
    - group: "crd.alauda.io"
    - group: "infrastructure.alauda.io"
    - group: "monitoring.coreos.com"
    - group: "networking.istio.io"
    - group: "networking.k8s.io"
    - group: "portal.alauda.io"
    - group: "rbac.authorization.k8s.io"
    - group: "storage.k8s.io"
    - group: "tke.cloud.tencent.com"
    - group: "devopsx.alauda.io"
    - group: "core.katanomi.dev"
    - group: "deliveries.katanomi.dev"
    - group: "integrations.katanomi.dev"
    - group: "builds.katanomi.dev"
    - group: "operators.katanomi.dev"
    - group: "tekton.dev"
    - group: "operator.tekton.dev"
    - group: "eventing.knative.dev"
    - group: "flows.knative.dev"
    - group: "messaging.knative.dev"
    - group: "operator.knative.dev"
    - group: "sources.knative.dev"
    - group: "operator.devops.alauda.io"
  - level: Metadata
```

TIP

If the cluster version is earlier than 1.24, change only the `apiVersion` field to `audit.k8s.io/v1beta1`. The rest of the policy content stays the same.

2. Upload `policy.yaml` to `/etc/kubernetes/audit/` on every control plane node.

> **WARNING**
>
> - If the cluster has multiple control plane nodes, upload the file to every node.
> - Create the directory manually if it does not exist: `/etc/kubernetes/audit/`

3. Update `/etc/kubernetes/manifests/kube-apiserver.yaml` on every control plane node.
   Add or update the following audit-related flags in `spec.containers[].command`:

Flag	Required	Description
`--audit-policy-file`	Yes	Must be set to `/etc/kubernetes/audit/policy.yaml`.
`--audit-log-format`	Yes	Must be set to `json`.
`--audit-log-path`	Yes	Must be set to `/etc/kubernetes/audit/audit.log`.
`--audit-log-mode`	No	Recommended value: `batch`.
`--audit-log-maxsize`	No	Maximum audit log file size in MiB. Recommended value: `200`.
`--audit-log-maxbackup`	No	Number of retained audit log files. Recommended value: `2`.

Example:

```
- --audit-log-format=json
- --audit-log-maxbackup=2
- --audit-log-maxsize=200
- --audit-log-mode=batch
- --audit-log-path=/etc/kubernetes/audit/audit.log
- --audit-policy-file=/etc/kubernetes/audit/policy.yaml
```

4. Add the audit directory mount configuration to the same `kube-apiserver.yaml` file.
   Add the following item under `spec.containers[].volumeMounts`:

```
- mountPath: /etc/kubernetes/audit
  name: k8s-audit
```

   Add the following item under `spec.volumes`:

```
- hostPath:
    path: /etc/kubernetes/audit
    type: DirectoryOrCreate
  name: k8s-audit
```

   WARNING

   - Update the manifest on every control plane node when the cluster has multiple control plane nodes.
   - The `volumeMounts[].name` value must match the corresponding `volumes[].name` value.
   - Do not change the mount path `/etc/kubernetes/audit`.

5. Save the file and verify that the configuration takes effect.
   Check whether `/etc/kubernetes/audit/audit.log` is generated on each control plane node. If the file exists and contains audit records, the configuration is effective.

```
ls -l /etc/kubernetes/audit/audit.log
tail -n 20 /etc/kubernetes/audit/audit.log
```

Creating an On-Premise Cluster

TOC

- Prerequisites
  - Node Requirements
  - Load Balancing
  - Connecting `global` Cluster and Workload Cluster
  - Image Registry
  - Container Networking
- Creation Procedure
  - Basic Info
  - Container Network
  - Node Settings
  - Extended Parameters
- Post-Creation Steps
  - Viewing Creation Progress
  - Associating with Projects

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, see 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. These choices configure the platform image source used by the workload cluster.

- **Platform Built-in**: Uses the platform image source configured for the `global` cluster. This source can be the Registry deployed with `global` or the external platform image registry selected during installation. If the source is the built-in Registry and the workload cluster cannot access `global`, see Add External Address for Built-in Registry.
- **Private Repository**: Uses a different customer-managed image registry for this workload cluster. Before selecting it, upload the complete payload required by the selected Kubernetes version and architecture, and meet the required capabilities and access conditions described in Prepare an External Platform Image Registry. For production use, also follow the production recommendations for HTTPS, DNS, availability, monitoring, backup, recovery, and retention.
- **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, prepare the 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

Kube-OVN is a cloud native Kubernetes container network orchestration system developed by Alauda. It provides Kubernetes networking capabilities for cross-cloud network management, traditional network architecture integration, infrastructure interconnection, and edge cluster deployment scenarios.

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. Plan the network carefully before creating the cluster.

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, see 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.

How to

Add External Address for Built- Updating Public Repository Cre Access a Cl

Add External Address for Built-in Registry

TOC

Overview

Prerequisites

Procedure

Configure Certificate and Routing Rules for the Platform Registry

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.

Configure an externally accessible address for the platform's default registry so workload clusters can 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
```

Updating Public Repository Credentials

TOC

Overview

Procedure

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.

Access a Cluster with KubeConfig

TOC

- Introduction
- Scenarios
- Choose the Access Path
- Prerequisites
- Procedure
  - Download and Use the Platform-Provided KubeConfig
  - Download the KubeConfig file
  - Select the KubeConfig file
  - Validate cluster access
- Build a Token-Based KubeConfig with Your Own ACP API Token
  - Create a ACP API token
  - Prepare a KubeConfig template
  - Set the ACP API token in the KubeConfig file
  - Validate the effective permissions
- Use a Dedicated Account for Pipeline or Connector Access
- Lifecycle and Revocation
- Verification
- Next Steps

Introduction

Use KubeConfig when you need local cluster access from ACP CLI (`ac`), `kubectl` , `helm` , `client-go` , or similar Kubernetes clients.

A KubeConfig file is a YAML client configuration file. It is not a Kubernetes resource. A KubeConfig file typically defines the target cluster endpoint, the client credential, and the context that combines them.

ACP CLI (`ac`) uses standard kubeconfig and provides a kubectl-like experience. It is compatible with `kubectl` workflows and also adds platform-specific context, cluster, and session operations. If you already use `ac login` , ACP CLI can configure kubeconfig and create contexts automatically. The access paths below include downloaded KubeConfig files and manually assembled token-based KubeConfig files.

Scenarios

Use these access paths for the following scenarios:

- A cluster administrator needs a ready-to-use KubeConfig for cluster administration, troubleshooting, or inspection.
- A developer or project user needs a KubeConfig that matches their own ACP identity and RBAC permissions.
- A pipeline or connector needs an independent credential that is separate from a personal day-to-day account.

Choose the Access Path

Use the following table to choose the appropriate access path:

Access path	Credential source	Effective permissions	Lifetime control	Best fit
Platform-downloaded	Credential embedded in	Determined by the credential	Controlled by the current	Cluster administrators

Access path	Credential source	Effective permissions	Lifetime control	Best fit
KubeConfig	the KubeConfig file downloaded from the platform	embedded in the downloaded file	platform behavior for the downloaded file	and template preparation
Token-based KubeConfig	ACP API token created for a user account	Limited to the identity and RBAC scope of the token owner	Controlled by the token expiration or revocation state	Developers, project users, and automation accounts

Current behavior

Currently, the KubeConfig file downloaded from ACP defaults to a 10-year validity period. ACP does not currently provide CA replacement for this path and does not support user-defined cluster CA for this path.

If you need stricter identity isolation or shorter lifecycle control, use a token-based KubeConfig instead of long-term reuse of the downloaded file.

Prerequisites

Before you begin, ensure the following conditions are met:

- You can sign in to ACP and access the target cluster.
- You have ACP CLI (`ac`) or `kubectl` installed on the local machine where you will use the KubeConfig file.
- If you want to use a token-based KubeConfig, you can create a ACP API token from **Profile** > **API Tokens**. For details, see Introduction.
- If you want to use KubeConfig for a pipeline or connector, prepare a dedicated account and grant only the required permissions before creating its token.

Procedure

Download and Use the Platform-Provided KubeConfig

Use this path when you need direct cluster access from the platform-provided KubeConfig file.

1 Download the KubeConfig file

In the **Administrator** view, go to **Clusters** > **Clusters**, select the target cluster, open the cluster details page, click **Actions**, and select **Download Kubeconfig**.

Save the downloaded file to a local path such as `~/.kube/<cluster-name>.yaml`.

2 Select the KubeConfig file

Point ACP CLI (`ac`) or `kubectl` to the downloaded file:

```
export KUBECONFIG="$HOME/.kube/<cluster-name>.yaml"
kubectl config get-contexts
kubectl config current-context
ac config current-context
```

If you manage multiple clusters, use `kubectl config use-context <context-name>` or `ac config use-context <context-name>` to switch to the correct context before running commands. If you use ACP CLI sessions, you can also use `ac config use-cluster <cluster-name>` to switch clusters within the current platform session.

3 Validate cluster access

Run read-only commands to confirm that the KubeConfig file works as expected:

```
kubectl get nodes
kubectl get pods -A
ac get nodes
ac get pods -A
```

If the commands return the expected cluster resources, the KubeConfig file is valid for direct use.

Use this downloaded file directly when you need cluster administration. You can also use it as a template for a token-based KubeConfig that works with both `ac` and `kubectl`.

Build a Token-Based KubeConfig with Your Own ACP API Token

Use this path when you want the KubeConfig permissions to match a specific ACP user or automation account.

This path does not elevate permissions. The KubeConfig file only carries the connection and authentication settings. The effective access boundary still depends on the token owner and that identity's RBAC assignments.

1 Create a ACP API token

Open **Profile** > **API Tokens**, create a new token, and set an expiration time that matches your use case.

Store the token securely. If you plan to use it for automation, avoid using a personal daily account.

2 Prepare a KubeConfig template

Use one of the following methods:

- Download the platform-provided KubeConfig file and save a separate copy for token-based access.
- Create a standard KubeConfig file manually.

If you create the file manually, use the target cluster endpoint in the following format:

```
https://<platform-access-address>/kubernetes/<cluster-name>
```

If you do not want to build the cluster section from scratch, download the platform-provided KubeConfig first and reuse its cluster and CA fields as the template.

3 Set the ACP API token in the KubeConfig file

Replace the `users[].user.token` value with the ACP API token that you created for the target user or automation account.

The following example shows a token-based KubeConfig:

KubeConfig.yaml

```yaml
apiVersion: v1
kind: Config
clusters:
  - name: <cluster-name>
    cluster:
      server: https://<platform-access-address>/kubernetes/<cluster-name>
      certificate-authority-data: <base64-encoded-ca-data>
users:
  - name: <user-name>
    user:
      token: <platform-api-token>
contexts:
  - name: <context-name>
    context:
      cluster: <cluster-name>
      user: <user-name>
      namespace: <namespace>
current-context: <context-name>
```

`namespace` is optional. It sets the default namespace for commands, but it does not increase the permission scope of the token.

4 Validate the effective permissions

Use the new KubeConfig file and verify that the returned permissions match the intended role:

```
export KUBECONFIG="$HOME/.kube/<user-or-automation>.yaml"
kubectl auth can-i get pods -n <namespace>
kubectl get pods -n <namespace>
ac config current-context
ac get pods -n <namespace>
```

If the command returns `no` for `kubectl auth can-i`, grant the required role or use a different account. Do not treat KubeConfig changes as a substitute for RBAC changes.

Use a Dedicated Account for Pipeline or Connector Access

For pipeline or connector access, use a dedicated account instead of a personal user account:

1. Create a dedicated account for the pipeline or connector.
2. Grant only the required project-level or tenant-level permissions to that account.
3. Create a ACP API token for that account.
4. Build a token-based KubeConfig by using that token.
5. Store the token or KubeConfig file in the secret management system of the automation platform.
6. Rotate the credential by creating a new token, updating the secret, and revoking the old token.

This pattern provides an independent credential for automation, but it is not a native project-level KubeConfig object in ACP.

Lifecycle and Revocation

The lifetime behavior depends on which access path you use:

Access path	What controls validity	Current behavior
Platform-downloaded KubeConfig	The current platform behavior for the downloaded file	The downloaded KubeConfig currently defaults to a 10-year validity period. ACP does not currently provide CA replacement or user-defined cluster CA for this path.
Token-based KubeConfig	The ACP API token in `users[].user.token`	The KubeConfig remains usable only while the token is valid. When the token expires or is revoked, the KubeConfig also stops working.

If you need a shorter validity period or easier revocation, prefer the token-based path.

Verification

After you prepare the KubeConfig file, perform the following checks:

1. Run `kubectl config get-contexts` and confirm that the expected context exists.
2. Run a read-only command such as `kubectl get pods -A` or `kubectl get pods -n <namespace>` and confirm that the response matches the intended access scope.
3. For a token-based KubeConfig, run `kubectl auth can-i <verb> <resource> -n <namespace>` and confirm that the result matches the token owner's role.
4. If you use ACP CLI, run `ac config current-context` or `ac namespace` and confirm that the active context matches the intended cluster and namespace.
5. If you use ACP CLI, run `ac get pods -n <namespace>` or `ac get nodes` and confirm that the response matches the intended access scope.

Next Steps

- For ACP API token management, see Introduction.
- For ACP CLI basics, see Getting Started with CLI.
- For ACP CLI context and session management, see Managing CLI Profiles.

- For ACP CLI and kubectl compatibility, see Usage of ac and kubectl Commands.