Menu

Install

This document will provide all the information regarding the installation of ACP.

Overview

Overview

Prepare for Installation

Prerequisites

Download

Node Preprocessing

Installing

Installing

Global Cluster Disaster Recovery

Global Cluster Disaster Recovery

Overview

By following this guide, you will complete the installation of **ACP Core**. If you need to understand the concept of **ACP Core**, refer to Architecture.

Installing **ACP Core** refers to the process of deploying the `global` cluster.

After the installation, you can **create new workload clusters** or **connect existing ones**, and install additional **Extensions** to enhance the platform's capabilities.

INFO

Before installation, please ensure that you have completed capacity planning, environment preprocessing, and prerequisite checks to ensure that the hardware, network, and OS of each node meet the requirements. The following content covers platform architecture design, installation methods, and key term explanations to help you grasp the core points during the actual installation process.

TOC

Installation Method

The installation process of the `global` cluster is mainly divided into three stages:

1. **Preparation Stage**

   - **Prerequisite Check**: Ensure that all node hardware, network, and OS meet the requirements, such as kernel version, CPU architecture, and network configuration.
   - **Installation Package Download**: Log in to the Alauda Customer Portal to obtain the latest installation package.
   - **Node Preprocessing**: Complete the preprocessing work of all nodes.

2. **Execution Stage**

   - **Installation Package Upload and Extraction**: Upload the installation package to the target control plane node (recommended directory: `/root/cpaas-install`) and extract the installation resources.
   - **Start the Installer**: Execute the installation script (such as `bash setup.sh`) on the control plane node, and select the network plugin (Kube-OVN or Calico), IP protocol mode (IPv4/IPv6/dual stack), and VIP configuration according to the actual environment.
   - **Parameter Configuration**: Access the Web UI provided by the installer, and set the Kubernetes version, cluster network, node name, access address, and other key parameters in sequence to complete the installation of the `global` cluster.

3. **Verification Stage**

   - **System Status Check**: After the installation is complete, log in to the platform Web UI to check the cluster status and the operating status of each component.
   - **CLI Verification**: Use command-line tools to check the cluster resource status to ensure that all services are running normally and there are no exceptions or failures.

The following chapters will further explain the detailed operations, configuration parameters, and verification methods of each installation stage. Please read carefully and complete the corresponding preparatory work before the formal installation.

Appendix — Alauda Customer Portal

Alauda Customer Portal is Alauda's unified customer service and delivery platform that provides centralized access to all product-related resources and support services. It serves as the official entry point for customers, partners, and delivery teams to obtain software packages, documentation, support, and license management in a secure and consistent manner.

Purpose and Overview

The Alauda Customer Portal streamlines the end-to-end product lifecycle—from installation and configuration to maintenance and support—by consolidating all essential resources into one platform. It ensures that every deployment is based on verified software versions and official technical guidance.

Key Features

- **Product Downloads** Provides access to verified installation and upgrade packages, ensuring that deployments are consistent with the latest supported product versions.
- **Knowledge Base** Offers comprehensive product documentation, technical articles, troubleshooting guides, and best practices to assist with installation, configuration, and operations.
- **Support Tickets** Enables users to submit, track, and manage support requests directly online, ensuring timely issue resolution and full visibility into support progress.
- **Application Marketplace** Delivers a curated collection of official and third-party extensions that can be installed to extend or customize the platform's capabilities.
- **License Management** Supports the application, activation, and renewal of software licenses, providing traceable and compliant license usage across all environments.

Usage Guidance

Before starting installation or upgrade activities, users should log in to the **Alauda Customer Portal** using their authorized account to download the required installation packages and verify license status. For customer delivery and production environments, the versions and documentation published on the Alauda Customer Portal should always be regarded as the **official baseline** for deployment and maintenance.

Menu

Prepare for Installation

Prerequisites

Download

Node Preprocessing

Prerequisites

Before installing the `global` cluster, you need to prepare hardware, network, and OS that meet the requirements.

INFO

1. The platform currently does not support direct installation of the `global` cluster in an existing Kubernetes environment. If your environment already has a Kubernetes cluster, please back up your data and clean the environment before installation.
2. If you plan to use global Cluster Disaster Recovery, please first read global Cluster Disaster Recovery.
3. Make sure all of the new nodes meet the Nodes Requirements.
4. The disks' performance and capacity also have to meet the Disk Configuration Requirements.

TOC

Resource Planning

This section provides guidelines for resource planning before installing ACP. Choose the appropriate deployment scenario based on your environment and usage needs, and prepare resources accordingly.

INFO

The following recommendations only cover the minimum resources required for a successful installation of the **Global cluster**.

They **do not** include the resources required for any **additional extensions or components** deployed on the Global cluster.

For detailed requirements of each extension, refer to the corresponding component documentation.

Deployment Architectures

WARNING

For ARM architectures (such as Kunpeng 920), it is recommended to increase the configuration to **2 times** that of the x86 minimum configuration, but not less than **1.5 times**.

For example: If x86 requires 8 cores 16GB, then ARM should reach at least 12 cores 24GB, and the recommended configuration is 16 cores 32GB.

Before installation, you must determine which deployment architecture best fits your use case. ACP supports the following three common deployment architectures:

- **Multi-Cluster**

  Select this architecture if you need to centrally manage multiple Kubernetes clusters. In this mode, ACP consists of one **Global cluster** and multiple **workload clusters**. Running non-platform workloads on the Global cluster may degrade platform stability and performance, and should be avoided.

- **Single Cluster**

Select this architecture if you plan to install only one cluster and run workloads directly on it. In this mode, the Global cluster also acts as a workload cluster, and therefore requires **more resources** compared with a pure Global-only setup in Multi-Cluster mode.

- **Single Node**

> **WARNING**
>
> This architecture is intended solely for testing or proof-of-concept purposes and should not be used in production.

Single Node

The following table lists the **minimum hardware requirements** for installing ACP in **Single Node** mode.

Resource	Minimum Requirement
CPU	12 cores
Memory	24GB
Storage	Storage Capacity

Single Cluster

In this mode, the Global cluster serves both as the control plane and as the workload cluster. The number of control plane nodes of global cluster MUST be 3.

The total resource requirement consists of two parts:

- **Base resources** for the Global cluster itself
- **Additional resources** for running workloads on the same cluster

The resources required for a **highly available Global cluster** are as follows:

Resource	Minimum Requirement
CPU	8 cores
Memory	16GB
Storage	Storage Capacity

To estimate the additional resources required for your workloads, refer to **Evaluating Resources for Workload Cluster**

Multi-Cluster

When managing multiple workload clusters, the resource usage of the Global cluster increases proportionally with the number of managed clusters. The additional overhead mainly comes from cluster registration, monitoring, and control-plane synchronization.

To estimate the resources required for your Global cluster based on the number of managed clusters, refer to **Evaluating Resources for Global Cluster**

Network

Before installation, the following network resources must be pre-configured. If a hardware LoadBalancer cannot be provided, the installer supports configuring **haproxy + keepalived** as a software load balancer, but you need to understand:

- **Poorer Performance**: Software load balancing performance is lower than hardware LoadBalancer.
- **Higher Complexity**: If you are not familiar with keepalived, it may cause the `global` cluster to be unavailable, problem troubleshooting will take a long time, and seriously affect platform reliability.

Network Resources

Resource	Mandatory	Quantity	Description
`global` VIP	Mandatory	1	Used for nodes in the cluster to access kube-apiserver, configured in the load balancing device to ensure high availability. This IP can also be used as the access address for the platform Web UI. Workload clusters in the same network as the `global` cluster can also access the `global` cluster through this IP.
External IP	Optional	On Demand	When there are workload clusters that are not in the same network as the `global` cluster, such as a hybrid cloud scenario, it must be provided. Workload clusters in other networks access the `global` cluster through this IP. This IP needs to be configured in the load balancing device to ensure high availability. This IP can also be used as the access address for the platform Web UI.
Domain Name	Optional	On Demand	If you need to access the `global` cluster or platform Web UI through a domain name, please provide it in advance and ensure that the domain name resolution is correct.
Certificate	Optional	On Demand	It is recommended to use a trusted certificate to avoid browser security warnings; if not provided, the installer will generate a self-signed certificate, but there may be security risks when using HTTPS.

INFO

A domain name must be provided in the following cases:

1. The `global` cluster needs to support IPv6 access.
2. A disaster recovery plan for the `global` cluster is planned.

NOTE

If the platform needs to configure multiple access addresses (for example, addresses for internal and external networks), please prepare the corresponding IP addresses or domain names in advance according to the table above. You can configure them in the installation parameters later, or add them according to the product documentation after installation.

Network Configuration

Type	Requirement Description
Network Speed	Speed of `global` cluster and workload cluster in the same network ≥1Gbps (recommended 10Gbps); cross-network speed ≥100Mbps (recommended 1Gbps). Insufficient speed will significantly reduce data query performance.
Network Latency	Latency ≤10 ms within the same cluster; latency ≤100 ms (recommended ≤30 ms) across clusters.
Network Policy	Please refer to LoadBalancer Forwarding Rules to ensure that the necessary ports are open; when using Calico CNI, ensure that the **IP-in-IP** protocol is enabled.
IP Address Range	The `global` cluster nodes should avoid using the 172.16-32 network segment. If it has been used, please adjust the Docker configuration (add the bip parameter) to avoid conflicts.

LoadBalancer Forwarding Rules

This rule is designed to ensure that the `global` cluster can receive traffic from the LoadBalancer normally. Please check the network policy according to the following table to ensure that the relevant ports are open.

Source IP	Protocol	Destination IP	Destination Port	Description
`global` VIP, External IP	TCP	All control plane node IPs	443	Provides access services for the platform Web UI, image repository, and Kubernetes API Server through the HTTPS protocol. The default port is `443`. If you need to use a custom HTTPS port, please do the following: • Replace the destination port in the port forwarding rule with your custom port number. • Later, in the Web UI installation parameters, fill in your custom port number.

Source IP	Protocol	Destination IP	Destination Port	Description
`global` VIP, External IP	TCP	All control plane node IPs	6443	This port provides access to the Kubernetes API Server for nodes within the cluster.
`global` VIP, External IP	TCP	All control plane node IPs	11443	This port provides access to the image repository for nodes within the cluster. **Note:** If you plan to use an external image repository instead of the default image repository provided by the `global` cluster, you do not need to configure this port.

TIP

- It is recommended to configure health checks on the LoadBalancer to monitor the port status.
- If you plan to implement a disaster recovery plan for the `global` cluster, you need to open port `2379` for all control plane nodes for ETCD data synchronization between the primary and disaster recovery clusters.

- The platform only supports HTTPS by default. If HTTP support is required, you need to open the HTTP port for all control plane nodes.

Download Core Package

Before installation, you need to download the **Core Package**.

> **INFO**
>
> Starting from Alauda Container Platform v4.1, if you download both the **Core Package** and the **Extensions Packages**, you must complete the installation of the **Core Package** before uploading and installing **Extensions Packages**.

Log in to the **Customer Portal** to download the **Core Package**.

Packages are available for **x86**, **ARM**, and **hybrid** architectures. The hybrid package includes images for both x86 and ARM, resulting in a larger package size. Select the package that best matches your environment.

If you do not have a registered account, please contact technical support.

TOC

Migrating from Single-Architecture to Hybrid

If you initially installed with an x86 or ARM Core Package but later need to support another architecture, you must re-download the **hybrid Core Package** and perform the following steps:

1. Upload the newly downloaded hybrid Core Package to any control plane node of the global cluster.

2. Extract the package and use the included `upgrade.sh` script to synchronize the multi-architecture images to your image registry:

   ```
   bash upgrade.sh --only-sync-image=true
   ```

3. After the script completes, check the `cluster.platform.tkestack.io` resource to verify whether the label `cpaas.io/node-arch-constraint` exists. If it does, you must remove it:

   ```
   kubectl get cluster.platform.tkestack.io global -oyaml | grep cpaas.io/node-arch-constraint
   # If there is output, edit the resource to remove the label; otherwise, you can skip this step.
   kubectl edit cluster.platform.tkestack.io global   ### Edit the labels field and delete cpaas.io/node-arch-constraint
   ```

Node Preprocessing

Before installing the `global` cluster, all nodes (control plane nodes and worker nodes) must complete preprocessing.

TOC

Supported OS and Kernel Versions

The following table lists the supported operating systems, their validated versions, and the corresponding tested kernel versions.

INFO

Only the kernel version shipped with the official operating system is supported. The kernel version must match the tested version (for example, A.B.C); the suffix after the dash (“-”) may differ.

If the OS, kernel version, or CPU architecture does not meet the requirements, please contact technical support.

x86

Red Hat Enterprise Linux (RHEL)

- RHEL 7.8: `3.10.0-1127.el7.x86_64`
- RHEL 8.0: `4.18.0-80.el8.x86_64`
- RHEL 8.6: `4.18.0-372.9.1.el8.x86_64`

WARNING

RHEL 7.8 does not support **Calico Vxlan IPv6**.

CentOS

- CentOS 7.6 to 7.9: `3.10.0-1127` to `3.11`

WARNING

CentOS does not support **Calico Vxlan IPv6**.

Ubuntu

- Ubuntu 20.04 LTS: `5.4.0-124-generic`
- Ubuntu 22.04 LTS: `5.15.0-56-generic`

WARNING

Ubuntu HWE (Hardware Enablement) versions are not supported.

Kylin Linux Advanced Server

- Kylin V10 SP3: `4.19.90-52.22.v2207.ky10.x86_64`

WARNING

- Kylin V10, V10-SP1, and V10-SP2 have known kernel issues that may cause **NodePort network access failures**; it is recommended to upgrade to **Kylin V10-SP3**.

ARM

Kylin Linux Advanced Server

- Kylin V10 SP3: `4.19.90-52.22.v2207.ky10.aarch64`

WARNING

- Kylin V10, V10-SP1, and V10-SP2 have known kernel issues that may cause **NodePort network access failures**; it is recommended to upgrade to **Kylin V10-SP3**.
- ARM architecture only supports `Kunpeng 920`. For other models, please contact technical support.

Execute the Quick Configuration Script

The ACP installation package provides a script for quickly configuring nodes.

Unzip the installation package to obtain the `init.sh` script file in the `res` directory. Copy the script file to the nodes and ensure that you have `root` privileges.

Execute the script:

```
bash init.sh
```

WARNING

`init.sh` cannot guarantee that all of the following checks are properly handled. You still need to continue with the steps below.

Node Checks

The following lists all the checks that must be completed on the nodes. Depending on the node's role, the required checks will vary. For example, some checks apply only to control plane nodes.

Checks are divided into two categories:

- ✅ Indicates a check that must pass.
- ⚠️ Indicates a check that must be met in specific scenarios. Please determine whether the corresponding conditions are met according to the instructions. If they are, you must resolve them.

The following is the list of checks:

- **OS and Kernel**
  - ✅ The machine's grub boot configuration must have the `transparent_hugepage=never` parameter.
  - ✅ CentOS 7.x system machine's grub boot configuration must have the `cgroup.memory=nokmem` parameter.
  - ✅ Check whether the kernel modules `ip_vs`, `ip_vs_rr`, `ip_vs_wrr`, and `ip_vs_sh` are enabled.
  - ⚠️ When the kernel version is lower than 4.19.0 (or RHEL is lower than 4.18.0), check whether the kernel modules `nf_conntrack_ipv4` and (for IPv6) `nf_conntrack_ipv6` are enabled.

- ⚠️ If the `global` cluster plans to use `Kube-OVN` CNI, the kernel modules `geneve` and `openvswitch` must be enabled.
- ✅ Disable apparmor/selinux and firewall.
- ✅ Disable `swap` .

- **Users and Permissions**

  - ✅ The node's SSH user has `root` privileges and can use `sudo` without the password.
  - ✅ The `UseDNS` and `UsePAM` parameters in `/etc/ssh/sshd_config` must be set to `no`.
  - ✅ Executing `systemctl show --property=DefaultTasksMax` returns `infinity` or a very large value; otherwise, adjust `/etc/systemd/system.conf`.

- **Node Network**

  - ✅ `hostname` must comply with the following rules:
    - No more than 36 characters.
    - Starts and ends with a letter or number.
    - Contains only lowercase letters, numbers, `-`, and `.`, but cannot contain `.-`, `..`, or `-.`.
  - ✅ `localhost` in `/etc/hosts` must resolve to `127.0.0.1`.
  - ✅ The `/etc/resolv.conf` file must exist and contain `nameserver` configurations, but must not contain addresses starting with 172 (disable systemd-resolved).
  - ⚠️ The `/etc/resolv.conf` file should not configure search domains (if you must configure them, see Configure Search Domain).
  - ✅ The machine's IP address cannot be a loopback, multicast, link-local, all-0, or broadcast address.
  - ✅ Executing `ip route` must return a default route or a route pointing to `0.0.0.0`.
  - ✅ The nodes must not occupy the following ports:
    - **Control plane nodes:** `2379`, `2380`, `6443`, `10249` ~ `10256`
    - **Node where the installer is located:** `8080`, `12080`, `12443`, `16443`, `2379`, `2380`, `6443`, `10249` ~ `10256`
    - **Worker nodes:** `10249` ~ `10256`

- ✅ If the `global` cluster uses **Kube-OVN** or **Calico**, ensure that the following ports are not occupied:
  - **Kube-OVN:** `6641`, `6642`
  - **Calico:** `179`
- ⚠️ Ensure that the IP addresses in the network segment `172.16.x.x` ~ `172.32.x.x` required by Docker are not occupied. If the IPs in this network segment are occupied and cannot be changed, please contact technical support.

- **Software and Directory Requirements:**
  - ✅ Must have the following installed: `ip`, `ss`, `tar`, `swapoff`, `modprobe`, `sysctl`, `md5sum`, and `scp` or `sftp`.
  - ⚠️ If you plan to use local storage **TopoLVM** or **Rook**, you need to install `lvm2`.
  - ✅ The `/etc/systemd/system/kubelet.service` file is not allowed to exist.
  - ✅ `/tmp` mount parameters must not contain `noexec`.
  - ✅ Remove packages that conflict with `global` cluster components (see Remove Conflicting Packages).
  - ✅ The following files must be deleted if they exist:
    - `/var/lib/docker`
    - `/var/lib/containerd`
    - `/var/log/pods`
    - `/var/lib/kubelet/pki`

- **Cross-Node Checks**
  - ✅ There must be no network firewall restrictions between nodes in the `global` cluster.
  - ✅ The `hostname` of each node in the cluster must be unique.
  - ✅ The time zones of all nodes must be unified, and the time synchronization error must be ≤ 10 seconds.

Appendix

Remove Conflicting Packages

Before installation, applications may already be running in the docker/containerd environment on the nodes, or software conflicting with the `global` cluster may have been installed. Therefore, it is necessary to check and uninstall conflicting packages.

DANGER

- To avoid application interruption or data loss, be sure to confirm whether there are conflicting software packages. When a conflict is found, please develop an application switching plan and back up your data before uninstalling.
- After uninstalling conflicting packages, you still need to check whether there are other potentially conflicting binary files in directories such as `/usr/local/bin/` (such as software related to docker, containerd, runc, podman, container network, container runtime, or Kubernetes).

The following commands can be used for reference.

CentOS / RedHat

Check:

```
for x in \
    docker docker-client docker-common docker-latest \
    podman-docker podman \
    runc \
    containernetworking-plugins \
    apptainer \
    kubernetes kubernetes-master kubernetes-node kubernetes-client \
    ; do
    rpm -qa | grep -F "$x"
done
```

Uninstall:

```
for x in \
    docker docker-client docker-common docker-latest \
    podman-docker podman \
    runc \
    containernetworking-plugins \
    apptainer \
    kubernetes kubernetes-master kubernetes-node kubernetes-client \
    ; do
    yum remove "$x"
done
```

Ubuntu

Check:

```
for x in \
    docker.io \
    podman-docker \
    containerd \
    rootlesskit \
    rkt \
    containernetworking-plugins \
    kubernetes \
    ; do
    dpkg-query -l | grep -F "$x"
done

for x in \
    kubernetes-worker \
    kubectl kube-proxy kube-scheduler kube-controller-manager kube-apiserver \
    k8s microk8s \
    kubeadm kubelet \
    ; do
    snap list | grep -F "$x"
done
```

Uninstall:

```
for x in \
    docker.io \
    podman-docker \
    containerd \
    rootlesskit \
    rkt \
    containernetworking-plugins \
    kubernetes \
    ; do
    apt-get purge "$x"
done

for x in \
    kubernetes-worker \
    kubectl kube-proxy kube-scheduler kube-controller-manager kube-apiserver \
    k8s microk8s \
    kubeadm kubelet \
    ; do
    snap remove --purge "$x"
done
```

Kylin

Check:

```
for x in \
    docker docker-client docker-common \
    docker-engine docker-proxy docker-runc \
    podman-docker podman \
    containernetworking-plugins \
    apptainer \
    containerd \
    kubernetes kubernetes-master kubernetes-node kubernetes-client kubernetes-kubeadm \
    ; do
    rpm -qa | grep -F "$x"
done
```

Uninstall:

```
for x in \
    docker docker-client docker-common \
    docker-engine docker-proxy docker-runc \
    podman-docker podman \
    containernetworking-plugins \
    apptainer \
    containerd \
    kubernetes kubernetes-master kubernetes-node kubernetes-client kubernetes-kubeadm \
    ; do
    yum remove "$x"
done
```

Configure Search Domain

In Linux OS, the `/etc/resolv.conf` file is used to configure DNS client domain name resolution settings. The `search` line specifies the domain search path for DNS queries.

Configuration Requirements

- **Number of Domains:** The number of domains in the `search` line should be less than `domainCountLimit - 3` (default `domainCountLimit` is 32).
- **Length of Single Domain:** Each domain name must not exceed 253 characters.
- **Total Character Length:** The total character count of all domain names and spaces must not exceed `MaxDNSSearchListChar` (default is 2048).

Example

```
search domain1.com domain2.com domain3.com
```

- The total number of domains is 3.
- The length of a single domain, such as `domain1.com`, is 11.
- The total character length is 35, i.e., 11 + 11 + 11 + 2 (two spaces).

WARNING

- If the `search` line in the `/etc/resolv.conf` file does not meet the above limitations, it may cause DNS query failures or performance degradation.
- Before modifying the `/etc/resolv.conf` file, it is recommended to back up the file.

Installing

This section describes the specific steps for installing the `global` cluster.

Before starting the installation, please ensure that you have completed the prerequisite checks, installation package download and verification, node preprocessing, and other preparatory work.

TOC

Process

1 Upload and Extract Installation Package

Upload the Core Package installation package to any machine of the `global` cluster control plane nodes, and extract it according to the following command:

```
# Assume that the /root/cpaas-install folder already exists on the machine
tar -xvf {Path to Core Package File}/{Core Package File Name} -C /root/cpaas-install
cd /root/cpaas-install/installer || exit 1
```

INFO

- This machine will become the first control plane node after the `global` cluster installation is complete.
- After the Core Package is extracted, at least **100GB** of disk space is required. Please ensure sufficient storage resources.
- If you have already downloaded extensions, complete the ACP Core installation first, and then follow Extend to upload and install them.

2 Start the Installer

Execute the following installation script to start the installer. After the installer starts successfully, the command line terminal will output the web console access address.

After waiting for about 5 minutes, you can use a browser on your PC to access the web console provided by the installer.

```
bash setup.sh
```

WARNING

Ensure that the IP address and port 8080 of the node where the installer is located can be accessed normally, so that the web console provided by the installer can be accessed smoothly after the installer starts successfully.

Network Mode and IP Family

```
bash setup.sh --network-mode calico
```

The `--network-mode` parameter affects the CNI of the `global` cluster created by the installer. If this parameter is not specified, the CNI of the `global` cluster will default to Kube-OVN. If you want Calico as the CNI, you must explicitly specify `--network-mode calico`.

```
bash setup.sh --ip-family ipv6
```

If you plan to create a `global` cluster with Single-stack Network IPv6, you must explicitly specify `--ip-family ipv6` when starting the installer. Without this parameter, the `global` cluster created by the installer will support Single-stack Network IPv4 and Dual-stack Network by default.

3 Parameter Configuration

After completing the installation parameter configuration according to the page guide, confirm the installation.

Parameter Description provides detailed descriptions of key parameters. Please read carefully and configure according to actual needs.

4 Verify Successful Installation

After the installation is complete, the platform access URL will be displayed on the page. Click the **Access** button to open the platform Web UI and verify whether the platform is accessible.

Next, run the following commands on the installation node to verify the installation status:

```
# Check if there are any failed Charts
kubectl get apprelease --all-namespaces
# Check if there are any Pods not in Running or Completed status
kubectl get pod --all-namespaces | awk '{if ($4 != "Running" && $4 !=
"Completed")print}' | awk -F'[/ ]+' '{if ($3 != $4)print}'
```

5 Install Product Docs Plugin

> **INFO**
>
> The **Alauda Container Platform Product Docs** plugin provides access to product documentation within the platform. All help links throughout the platform will direct users to this documentation. If this plugin is not installed, clicking help links in the platform will result in 404 access errors.

1. Navigate to **Administrator**.
2. In the left sidebar, click **Marketplace** > **Cluster Plugins** and select the `global` cluster.
3. Locate the **Alauda Container Platform Product Docs** plugin and click **Install**.

Parameter Description

Parameter	Description
Kubernetes Version	All optional versions are rigorously tested for stability and compatibility. **Recommendation:** Choose the latest version for optimal features and support.
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 Address	Enter the pre-prepared domain name. If no domain name is available, enter the pre-prepared `global VIP`.

	`Self-Built VIP` is disabled by default, only enable it if you have not provided a LoadBalancer. After enabling, the installer will automatically deploy `keepalived` to provide software load balancing support. **Note:** The following conditions must be met when using `Self-Built VIP`, • A usable VRID is available; • The host network supports the VRRP protocol; • All control plane nodes and the VIP must be on the same subnet. **Tip:** For single-node deployments in feature experience scenarios, you can directly enter the node IP. There is no need to enable `Self-Built VIP` or prepare network resources such as `global VIP`.

Platform Access Address	If you do not need to distinguish between **Cluster Address** and **Platform Access Address**, enter the same address as the **Cluster Address**. If you need to distinguish, for example, if the `global` cluster is only for internal network access and the platform needs to provide external network access, enter the pre-prepared domain name or `External IP`. The platform uses HTTPS access by default and does not enable HTTP. If you need to enable HTTP access, enable it in **Advanced Settings** (not recommended). **Note:** A domain name must be entered in the following cases, • A disaster recovery plan for the `global` cluster is planned; • The platform needs to support IPv6 access. **Tip:** If you need to configure more platform access addresses, you can add them in **Other Settings > Other Platform Access Addresses** in the next step. Or, after installation, add them in platform management according to the user manual.
Certificate	The platform provides self-signed certificates to support HTTPS access by default. If you need to use a custom certificate, you can upload an existing certificate.
Image Repository	The `Platform Deployment` image repository is used by default, which contains images of all components. If you need to use an `External` image repository, please contact technical support to obtain the image synchronization plan before configuring.
Container Network	The default subnet and Service network segment of the cluster cannot overlap.

	When using the Kube-OVN Overlay network, ensure that the container network and the host network are not in the same network segment, otherwise it may cause network exceptions.
Node Name	If you select `Host Name as Node Name`, ensure that the host names of all nodes are unique.
`global` **Cluster Platform Node Isolation**	Enable only when you plan to run application workloads in the `global` cluster. **After enabling:** • Nodes can be set to `Platform Exclusive`, i.e., only run platform components, ensuring platform and application workloads are isolated; • Workloads of the DaemonSet type are excluded.
Add Node	**Control Plane Node:** • Supports adding 1 or 3 control plane nodes (3 for high availability configuration); • If `Platform Exclusive` is enabled, `Deployable Applications` is forced to be disabled, and control plane nodes only run platform components; • If `Platform Exclusive` is disabled, you can choose whether to enable `Deployable Applications`, allowing control plane nodes to run application workloads. **Worker Node:** • If `Platform Exclusive` is enabled, `Deployable Applications` is forced to be disabled; • If `Platform Exclusive` is disabled, `Deployable Applications` is forced to be enabled.

	When using Kube-OVN, you can specify the node network card by entering the gateway name. If the node availability check fails, please adjust it according to the page prompt and add it again.

Installer Cleanup

Normally, the installer will be automatically deleted after installation. If the installer is not automatically deleted after 30 minutes of installation, please execute the following command on the node where the installer is located to force delete the installer container:

```
docker rm -f minialauda-control-plane
```

Additional Resources

- Upload and Install Extensions

Global Cluster Disaster Recovery

TOC

Overview

This solution is designed for disaster recovery scenarios involving the `global` cluster. The `global` cluster serves as the control plane of the platform and is responsible for managing other clusters. To ensure continuous platform service availability when the `global` cluster fails, this solution deploys two `global` clusters: a Primary Cluster and a Standby Cluster.

The disaster recovery mechanism is based on real-time synchronization of etcd data from the Primary Cluster to the Standby Cluster. If the Primary Cluster becomes unavailable due to a failure, services can quickly switch to the Standby Cluster.

Supported Disaster Scenarios

- Irrecoverable system-level failure of the Primary Cluster rendering it inoperable;
- Failure of physical or virtual machines hosting the Primary Cluster, making it inaccessible;
- Network failure at the Primary Cluster location resulting in service interruption;

Unsupported Disaster Scenarios

- Failures of applications deployed within the `global` cluster;
- Data loss caused by storage system failures (outside the scope of etcd synchronization);

The roles of **Primary Cluster** and **Standby Cluster** are relative: the cluster currently serving the platform is the Primary Cluster (DNS points to it), while the standby cluster is the Standby Cluster. After a failover, these roles are swapped.

Notes

- This solution only synchronizes etcd data of the `global` cluster; it does not include data from registry, chartmuseum, or other components;
- In favor of facilitating troubleshooting and management, it is recommended to name nodes in a style like `standby-global-m1`, to indicates which cluster the node belongs to (Primary or Standby).
- Disaster recovery of application data within the cluster is not supported;
- Stable network connectivity is required between the two clusters to ensure reliable etcd synchronization;
- If the clusters are based on heterogeneous architectures (e.g., x86 and ARM), use a dual-architecture installation package;

- The following namespaces are excluded from etcd synchronization. If resources are created in these namespaces, users must back them up manually:

```
cpaas-system
cert-manager
default
global-credentials
cpaas-system-global-credentials
kube-ovn
kube-public
kube-system
nsx-system
cpaas-solution
kube-node-lease
kubevirt
nativestor-system
operators
```

- If both clusters are set to use built-in image registries, container images must be uploaded separately to each;

- If the Primary Cluster deploys **Alauda DevOps Eventing v3** (knative-operator) and instances thereof, the same components must be pre-deployed in the standby cluster.

Process Overview

1. Prepare a unified domain name for platform access;
2. Point the domain to the **Primary Cluster's** VIP and install the **Primary Cluster**;
3. Temporarily switch DNS resolution to the standby VIP to install the Standby Cluster;
4. Copy the ETCD encryption key of the **Primary Cluster** to the nodes that will later be the control plane nodes of Standby Cluster;
5. Install and enable the etcd synchronization plugin;
6. Verify sync status and perform regular checks;
7. In case of failure, switch DNS to the standby cluster to complete disaster recovery.

Required Resources

- A unified domain which will be the `Platform Access Address`, and the TLS certificate plus private key for serving HTTPS on that domain;
- A dedicated virtual IP address for each cluster — one for the **Primary Cluster** and another for the Standby Cluster;
  - Preconfigure the load balancer to route TCP traffic on ports `80`, `443`, `6443`, `2379`, and `11443` to the control-plane nodes behind the corresponding VIP.

Procedure

Step 1: Install the Primary Cluster

NOTES OF DR (Disaster Recovery Environment) INSTALLING

While installing the primary cluster of the DR Environment,

- First of all, documenting all of the parameters set while following the guide of the installation web UI. It is necessary to keep some options the same while installing the standby cluster.
- A **User-provisioned** Load Balancer MUST be preconfigured to route traffic sent to the virtual IP. The `Self-built VIP` option is NOT available.
- The `Platform Access Address` field MUST be a domain, while the `Cluster Endpoint` MUST be the virtual IP address.
- Both clusters MUST be configured to use `An Existing Certificate` (has be the same one), request a legit certificate if necessary. The `Self-signed Certificate` option is NOT available.
- When `Image Repository` is set to `Platform Deployment`, both `Username` and `Password` fields MUST NOT be empty; The `IP/Domain` field MUST be set to the domain used as the `Platform Access Address`.
- Both `HTTP Port` and `HTTPS Port` fields of `Platform Access Address` MUST be 80 and 443.

- When coming to the second page the of the installation guide (Step: `Advanced`), the `Other Platform Access Addresses` field MUST include the virtual IP of current Cluster.

Refer to the following documentation to complete installation:

- Prepare for Installation
- Installing

Step 2: Install the Standby Cluster

1. Temporarily point the domain name to the standby cluster's VIP;

2. Log into the first control plane node of the **Primary Cluster** and copy the etcd encryption config to all standby cluster control plane nodes:

```
# Assume the primary cluster control plane nodes are 1.1.1.1, 2.2.2.2 & 3.3.3.3
# and the standby cluster control plane nodes are 4.4.4.4, 5.5.5.5 & 6.6.6.6
for i in 4.4.4.4 5.5.5.5 6.6.6.6  # Replace with standby cluster control plane node IPs
do
  ssh "<user>@$i" "sudo mkdir -p /etc/kubernetes/"
  scp /etc/kubernetes/encryption-provider.conf "<user>@$i:/tmp/encryption-provider.conf"
  ssh "<user>@$i" "sudo install -o root -g root -m 600 /tmp/encryption-provider.conf /etc/kubernetes/encryption-provider.conf && rm -f /tmp/encryption-provider.conf"
done
```

3. Install the standby cluster in the same way as the primary cluster

NOTES FOR INSTALLING STANDBY CLUSTER

While installing the standby cluster of the DR Environment, the following options MUST be set to the same as the **primary cluster**:

- The `Platform Access Address` field.
- All fields of `Certificate`.
- All fields of `Image Repository`

- Important: ensure the credentials of image repository and the ACP admin user match those set on the **Primary Cluster**.

and MAKE SURE you followed the `NOTES OF DR (Disaster Recovery Environment) INSTALLING` in Step 1.

Refer to the following documentation to complete installation:

- Prepare for Installation
- Installing

Step 3: Enable etcd Synchronization

1. When applicable, configure the load balancer to forward port `2379` to control plane nodes of the corresponding cluster. ONLY TCP mode is supported; forwarding on L7 is not supported.

   INFO

   Port forwarding through a load balancer is not required. If direct access from the standby cluster to the active global cluster is available, specify the etcd addresses via **Active Global Cluster ETCD Endpoints**.

2. Access the **standby global cluster** Web Console using its VIP, and switch to **Administrator** view;

3. Navigate to **Marketplace > Cluster Plugins**, select the `global` cluster;

4. Find **Alauda Container Platform etcd Synchronizer**, click **Install**, configure parameters:

   - When not forwarding port `2379` through load balancer, its required to configure **Active Global Cluster ETCD Endpoints** correctly;
   - Use the default value of **Data Check Interval**;
   - Leave **Print detail logs** switch disabled unless troubleshooting.

Verify the sync Pod is running on the standby cluster:

```
kubectl get po -n cpaas-system -l app=etcd-sync
kubectl logs -n cpaas-system $(kubectl get po -n cpaas-system -l app=etcd-sync --no-headers | head -1) | grep -i "Start Sync update"
```

Once “Start Sync update” appears, recreate one of the pods to re-trigger sync of resources with ownerReference dependencies:

```
kubectl delete po -n cpaas-system $(kubectl get po -n cpaas-system -l app=etcd-sync --no-headers | head -1)
```

Check sync status:

```
mirror_svc=$(kubectl get svc -n cpaas-system etcd-sync-monitor -o jsonpath='{.spec.clusterIP}')
ipv6_regex="^[0-9a-fA-F:]+$"
if [[ $mirror_svc =~ $ipv6_regex ]]; then
  export mirror_new_svc="[$mirror_svc]"
else
  export mirror_new_svc=$mirror_svc
fi
curl $mirror_new_svc/check
```

Output explanation:

- `LOCAL ETCD missed keys`: Keys exist in the Primary but are missing from the standby. Often caused by GC due to resource order during sync. Restart one etcd-sync Pod to fix;
- `LOCAL ETCD surplus keys`: Extra keys exist only in the standby cluster. Confirm with ops team before deleting these keys from the standby.

If the following components are installed, restart their services:

- Alauda Container Platform Log Storage for Elasticsearch:

  ```
  kubectl delete po -n cpaas-system -l service_name=cpaas-elasticsearch
  ```

- Alauda Container Platform Monitoring for VictoriaMetrics:

```
kubectl delete po -n cpaas-system -l 'service_name in
(alertmanager,vmselect,vminsert)'
```

Disaster Recovery Process

1. Restart Elasticsearch on the standby cluster in case it is necessary:

```
# Copy installer/res/packaged-scripts/for-upgrade/ensure-asm-template.sh to /root:
# DO NOT skip this step

# switch to the root user if necessary
sudo -i

# check whether the Log Storage for Elasticsearch is installed on global cluster
_es_pods=$(kubectl get po -n cpaas-system | grep cpaas-elasticsearch | awk '{print
$1}')
if [[ -n "${_es_pods}" ]]; then
    # In case the script returned the 401 error, restart Elasticsearch
    # then execute the script to check the cluster again
    bash /root/ensure-asm-template.sh

    # Restart Elasticsearch
    xargs -r -t -- kubectl delete po -n cpaas-system <<< "${_es_pods}"
fi
```

2. Verify data consistency in the standby cluster (same check as in Step 3);

3. Uninstall the etcd synchronization plugin;

4. Remove port forwarding for `2379` from both VIPs;

5. Switch the platform domain DNS to the standby VIP, which now becomes the Primary Cluster;

6. Verify DNS resolution:

```
kubectl exec -it -n cpaas-system deployments/sentry -- nslookup <platform access domain>
# If not resolved correctly, restart coredns Pods and retry until success
```

7. Clear browser cache and access the platform page to confirm it reflects the former standby cluster;

8. Restart the following services (if installed):

   - Alauda Container Platform Log Storage for Elasticsearch:

     ```
     kubectl delete po -n cpaas-system -l service_name=cpaas-elasticsearch
     ```

   - Alauda Container Platform Monitoring for VictoriaMetrics:

     ```
     kubectl delete po -n cpaas-system -l 'service_name in (alertmanager,vmselect,vminsert)'
     ```

   - cluster-transformer:

     ```
     kubectl delete po -n cpaas-system -l service_name=cluster-transformer
     ```

9. If workload clusters send monitoring data to the Primary, restart warlock in the workload cluster:

   ```
   kubectl delete po -n cpaas-system -l service_name=warlock
   ```

10. On the original Primary Cluster, repeat the Enable etcd Synchronization steps to convert it into the new standby cluster.

Routine Checks

Regularly check sync status on the standby cluster:

```
curl $(kubectl get svc -n cpaas-system etcd-sync-monitor -o
jsonpath='{.spec.clusterIP}')/check
```

If any keys are missing or surplus, follow the instructions in the output to resolve them.

Uploading Packages

WARNING

When using `violet` to upload packages to a standby cluster, the parameter `--dest-repo <VIP addr of standby cluster>` must be specified.
Otherwise, the packages will be uploaded to the image repository of the **primary cluster**, preventing the standby cluster from installing or upgrading extensions.

Also be awared that either authentication info of the standby cluster's image registry or `--no-auth` parameter MUST be provided.

For details of the `violet push` subcommand, please refer to Upload Packages.