Install

Installation Path Selection

This section documents installation onto a **traditional operating system** such as Ubuntu or RHEL. If your environment uses Alauda OS, see Installing the global Cluster on Immutable Infrastructure instead.

Overview

Overview

Plan

Plan

Global Cluster Disaster Recovery

Global Cluster Disaster Recovery

Prepare for Installation

Prerequisites Download Node Prepr

Prepare an External Platform Image Registry

Installing

Installing

Installer Parameters

Installer Parameters

Validation

Validation

Installation Troubleshooting

Installation Troubleshooting

Next Steps

Next Steps

Overview

Use the Install section to plan and complete an ACP 4.4 installation.

The installation deploys ACP Core and the required Aligned Extensions on the `global` cluster. Core provides the platform management plane, cluster management, users and roles, and Extension management frameworks. The required Aligned Extensions provide the Web Console and other platform-facing capabilities. For the platform architecture and definitions of `global` cluster and workload cluster, see Architecture and Glossary.

The Core Package and the required Aligned Extension packages are separate downloads. Before starting the installer, copy the eight required Extension packages listed in Download into the `plugins/` directory of the extracted Core Package. Other Extensions are installed after the platform installation through the Extend workflow.

INFO

ACP Core installation does not support direct installation of the `global` cluster into an existing Kubernetes environment. Prepare new nodes that meet the hardware, network, storage, and OS requirements before you start installation.

TOC

Installation Scope

The installation provides the platform management plane and the required Aligned Extensions. After the installation is verified, you can create or connect workload clusters and install additional Extensions according to your platform scenario.

> **Installation Path**
>
> The procedure described in this section installs the `global` cluster onto a **traditional operating system** such as Ubuntu or RHEL. If the `global` cluster uses Alauda OS, see Installing the global Cluster on Immutable Infrastructure ↗ instead.

The following work is outside this installation flow:

- Creating workload clusters.
- Connecting existing Kubernetes clusters.
- Uploading and installing Extensions other than the eight required Aligned Extensions.
- Installing optional Operators or Cluster Plugins.
- Configuring optional platform capabilities such as storage, monitoring, backup, registry, or identity integrations.

Installation Workflow

The installation follows this high-level path:

1. Plan. Choose the deployment architecture and confirm decisions that cannot be changed safely after installation.
2. If required, read Global Cluster Disaster Recovery before preparing resources. DR affects addresses, certificates, load balancing, registry selection, and the consistency of both installations.
3. Prepare for installation. Meet the hardware, storage, network, and node requirements.

4. Download. Download the matching Core Package and the required Aligned Extension packages.
5. If selected, prepare the external platform image registry and upload the complete target-version payload.
6. Installing. Extract the Core Package, stage the required packages, start the installer, and complete the Web UI workflow.
7. Use Installer Parameters while reviewing the values entered in the installer.
8. Validation. Accept the installation only after the Web UI, Core, required Aligned Extensions, image source, artifacts, and Pods are healthy.
9. If the installer stalls or validation fails, use Installation Troubleshooting.
10. Next Steps. Install the Product Docs plugin and continue with scenario-specific configuration.

The Customer Portal is the official source for ACP installation packages. For download requirements and package architecture options, see Download.

> **Separate Downloads**
>
> The Core Package does not contain the required Aligned Extension packages. Download them separately and manually copy only the eight packages listed in Download into the `plugins/` directory of the extracted Core Package before starting the installer.

Related Documentation

- Architecture
- Glossary
- Kubernetes Support Matrix
- Disk Configuration Requirements
- Extend

Plan

Before you start the installer, choose the deployment architecture and confirm the installation settings that affect later operations.

INFO

The ACP installation deploys the `global` cluster on prepared nodes. It does not support direct installation of the `global` cluster into an existing Kubernetes environment.

TOC

Choose a Deployment Architecture

Choose the architecture that matches your business scenario, management model, workload placement, resource planning, and isolation requirements.

Architecture	Use when	Planning notes
Multi-Cluster	You need one `global` cluster to manage multiple workload clusters.	Avoid running non-platform workloads on the `global` cluster. Plan resources for the expected number of managed workload clusters.
Single Cluster	You intentionally use one cluster for platform components and application workloads.	Single Cluster supports production use when it fits the business scenario. Because the `global` cluster also runs application workloads, plan resources and platform/workload isolation before installation.
Single Node	You need a test or proof-of-concept environment.	Do not use Single Node for production.

For resource requirements and sizing guidance, see Prerequisites.

Confirm Install-Time Decisions

Confirm the following decisions before running the installer:

Decision	What to confirm	Continue with
Installation packages	Choose an x86, ARM, or hybrid Core Package and download the required Aligned Extension packages for the same ACP version and architecture.	Download
Cluster network protocol	Choose IPv4, IPv6, or dual stack before starting the installer.	Installing
Cluster Endpoint and Platform Access Address	Prepare the access addresses that cluster components, administrators, and users will use.	Prerequisites and Installer Parameters
Load balancing	Decide whether to use an external load balancer or Self-built VIP.	Prerequisites and LoadBalancer Forwarding Rules

Decision	What to confirm	Continue with
Certificates	Decide whether to use installer-generated self-signed certificates or trusted certificates that you provide.	Prerequisites and Installer Parameters
Platform image source	Decide whether to use the platform built-in Registry or a customer-managed external platform image registry. For new production deployments, use an external registry when it meets the documented availability and lifecycle requirements.	Prepare an External Platform Image Registry and Installer Parameters
Nodes	Confirm node names, node roles, and whether platform node isolation is required.	Node Preprocessing and Installer Parameters
Global Cluster Disaster Recovery	Decide whether your environment requires primary and standby `global` clusters.	Global Cluster Disaster Recovery

If you plan to use Global Cluster Disaster Recovery, read Global Cluster Disaster Recovery before installing ACP Core. Disaster recovery planning affects domain names, VIPs, certificates, load balancer rules, image registry choices, package architecture, and installation parameter consistency.

Read Before You Install

Use the following pages as the source of truth before running the installer:

- Prerequisites: Prepare resource, network, and storage requirements.
- Disk Configuration Requirements: Confirm disk capacity and performance requirements.
- Node Preprocessing: Check and prepare each node before installation.
- Download: Download the Core Package and `Common ACP Extensions` separately, and verify that their version and architecture match.
- Prepare an External Platform Image Registry: If selected, upload the complete target-version Core and Extension payload before starting the installer.

- Global Cluster Disaster Recovery: Plan primary and standby `global` clusters if disaster recovery is required.
- Installing: Upload the Core Package, stage required packages, and start the installer.
- Installer Parameters: Configure the installation choices in the Web UI.

Global Cluster Disaster Recovery

Disaster Recovery Path

This page documents disaster recovery for `global` clusters running on a **traditional operating system**. If the `global` cluster uses Alauda OS, follow Global Cluster Disaster Recovery on Immutable Infrastructure ↗ instead.

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 **Alauda Container Platform etcd Synchronizer**;
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;

Network Requirements

The unified domain is a prerequisite for disaster recovery. During normal operation, the domain resolves only to the **Primary Cluster** VIP. The Standby Cluster uses the domain to access platform services on the Primary Cluster.

Before installing either cluster, complete both of the following network preparations:

- Configure each cluster's load balancer to forward the required TCP ports on its VIP to the control plane nodes behind that VIP. Configure port `80` only when users access the platform over HTTP.
- Allow the required TCP ports in **both directions** between the Primary Cluster and Standby Cluster networks. Apply the rules to firewalls, security groups, router ACLs, and any other inter-site network controls.

TCP port	Service	Why it is required
`443`	ACP platform HTTPS	The Standby Cluster uses the Platform Access Address to reach the active platform. **Alauda Container Platform etcd Synchronizer** uses this address, and logging and monitoring components send alert callbacks to it.

TCP port	Service	Why it is required
`80`	ACP platform HTTP	Required only when users access the platform over HTTP. The connectivity requirement is otherwise the same as for port `443`.
`6443`	Kubernetes API server	During installation or reinstallation of **Alauda Container Platform etcd Synchronizer**, the Standby Cluster connects to the active cluster API server to obtain the etcd CA material required for synchronization.
`11443`	Built-in image registry	The Standby Cluster pulls platform and plugin images from the registry configured through the Platform Access Address.
`2379`	etcd	**Alauda Container Platform etcd Synchronizer** on the Standby Cluster reads etcd data from the Primary Cluster and writes it to the local standby etcd.

The network policy must be bidirectional because the cluster roles are reversed after failover. Before failover, the effective service traffic is normally from Standby to Primary. After failover, the former Primary becomes the new Standby and must reach the new Primary on the same ports. This requirement does not make etcd replication bidirectional: **Alauda Container Platform etcd Synchronizer** runs only on the current Standby Cluster and writes synchronized data to its local etcd.

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 contro
l plane node IPs
do
  ssh "<user>@$i" "sudo mkdir -p /etc/kubernetes/"
  scp /etc/kubernetes/encryption-provider.conf "<user>@$i:/tmp/encrypti
on-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/en
cryption-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. Before installing the plugin, get the bearer token for accessing the active global cluster API server from the active global cluster. Copy the command output and use it when creating the `etcd-sync-active-cluster-token` Secret in the standby global cluster under the `cpaas-system` namespace. The Secret stores the token under the data key `token`.

   ```
   # Run this command on the active cluster.
   kubectl -n cpaas-system get secret k8sadmin -o jsonpath='{.data.token}'
   | base64 -d
   ```

   Copy the output value, then run this command on the standby cluster:

   ```
   ACTIVE_CLUSTER_TOKEN='<paste-the-token-from-the-active-cluster>'
   kubectl -n cpaas-system create secret generic etcd-sync-active-cluster-
   token \
     --from-literal=token="${ACTIVE_CLUSTER_TOKEN}" \
     --dry-run=client -o yaml | kubectl apply -f -
   ```

2. 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**.

3. Access the **standby global cluster** Web Console using its VIP, and switch to **Administrator** view.
4. Navigate to **Marketplace > Cluster Plugins**, select the `global` cluster.
5. Find **Alauda Container Platform etcd Synchronizer**, click **Install**, and configure these parameters:

   - Set **Active Global Cluster VIP** to the VIP of the active global cluster.
   - When not forwarding port `2379` through the load balancer, configure **Active Global Cluster ETCD Endpoints** correctly.

- Set **Standby Cluster ETCD Endpoints** to the standby cluster etcd address. Use the default value unless the local etcd service is exposed through a different endpoint.
- Set **Active Global Cluster Token Secret** to `etcd-sync-active-cluster-token`.
- Use the default value of **Data Check Interval**.
- Leave **Print detail logs** disabled unless troubleshooting.

During installation, the system runs the `etcd-sync-bootstrap` Job before the `etcd-sync` Deployment starts. The plugin installation continues only after the Job prepares `remote-etcd-ca`, `remote-etcd-issuer`, and `remote-etcd-client`.

Verify the bootstrap Job and runtime resources:

```
kubectl get job -n cpaas-system etcd-sync-bootstrap
kubectl logs -n cpaas-system job/etcd-sync-bootstrap
kubectl get secret -n cpaas-system remote-etcd-ca
kubectl get issuer -n cpaas-system remote-etcd-issuer
kubectl get certificate -n cpaas-system remote-etcd-client
kubectl get secret -n cpaas-system remote-etcd-client
```

Verify the sync Pods are running on the standby cluster and identify the current leader:

```
kubectl get po -n cpaas-system -l app=etcd-sync
kubectl get lease -n cpaas-system etcd-sync-mirror
leader_pod=$(kubectl get lease -n cpaas-system etcd-sync-mirror -o jsonpa
th='{.spec.holderIdentity}')
kubectl logs -n cpaas-system "$leader_pod" | grep -E "Acquired leader lea
se|Start Sync update"
```

If resources with `ownerReference` dependencies need to be resynchronized, recreate the current leader Pod after `Start Sync update` appears:

```
leader_pod=$(kubectl get lease -n cpaas-system etcd-sync-mirror -o jsonpa
th='{.spec.holderIdentity}')
kubectl delete po -n cpaas-system "$leader_pod"
```

Check sync status:

```
mirror_svc=$(kubectl get svc -n cpaas-system etcd-sync-monitor -o jsonpat
h='{.spec.clusterIP}')
ipv6_regex="^[0-9a-fA-F:]+$"
if [[ $mirror_svc =~ $ipv6_regex ]]; then
  mirror_host="[$mirror_svc]"
else
  mirror_host="$mirror_svc"
fi
curl -g "http://${mirror_host}/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 the current etcd-sync leader 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,vms
  elect,vminsert)'
  ```

Disaster Recovery Process

> **Verify primary/standby consistency before uninstalling Alauda Container Platform etcd Synchronizer**
>
> This procedure uninstalls **Alauda Container Platform etcd Synchronizer**. Before you uninstall it, confirm that the standby cluster data is consistent with the primary cluster. Uninstalling the plugin

while the standby is missing data that the primary holds can cause owner references to resolve incorrectly, and workload-cluster node `Machine` objects — including immutable-OS clusters, where this destroys the backing virtual machine — may be deleted. If the consistency check reports missing or surplus keys, do not uninstall the plugin; resolve the inconsistency or contact technical support first.

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

```
# Copy installer/res/packaged-scripts/for-upgrade/ensure-asm-template.s
h 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 globa
l 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). If the check reports missing or surplus keys, the standby is not consistent with the primary: do not continue to the next step. Resolve the inconsistency or contact technical support first. On **both** clusters, also confirm that no `Machine` nodes are in a non-running state, and resolve any that are before continuing:

```
kubectl get machines.platform.tkestack.io
```

3. Uninstall **Alauda Container Platform etcd Synchronizer**;
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 <platfo
rm access domain>
# If not resolved correctly, restart coredns Pods and retry until succe
ss
```

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,v
mselect,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:

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

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

Uploading Packages

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

Prepare for Installation

Prerequisites Download Node Prepr

Prepare an External Platform Image Registry

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, ensure that the required network resources are prepared.

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.

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

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	Recommended	On Demand	Recommended for Cluster Endpoint and Platform Access Address. Please provide it in advance and ensure that the domain name resolution is correct. Using a domain name is recommended because if the VIP needs to be changed after cluster installation, you can easily update the DNS record without impacting cluster operations.

Resource	Mandatory	Quantity	Description
			A domain name is **required** in the following cases: • The `global` cluster needs to support IPv6 access; • A disaster recovery plan for the `global` cluster is planned.
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.

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 Bandwidth	Intra-cluster bandwidth must be ≥1 Gbps (recommended 10 Gbps). Cross-cluster bandwidth must be ≥100 Mbps (recommended 1 Gbps). Insufficient bandwidth may significantly degrade data query performance.
Network Latency	Intra-cluster latency must be ≤10 ms. Cross-cluster latency must be ≤100 ms (recommended ≤30 ms).
Network Policy	Please refer to LoadBalancer Forwarding Rules to ensure that the necessary ports are open.

Type	Requirement Description
IP Address Range	The `global` cluster nodes should avoid using the 172.17-18 network segment. If it has been used, please adjust the nerdctl 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

Source IP	Protocol	Destination IP	Destination Port	Description
				rule with your custom port number. • Later, in the Web UI installation parameters, fill in your custom port number.
`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 platform image registry instead of the Registry

Source IP	Protocol	Destination IP	Destination Port	Description
				provided by the `global` cluster, you do not need to configure this port. See Prepare an External Platform Image Registry for the external registry network requirements.

TIP

- It is recommended to configure health checks on the LoadBalancer to monitor the port status.
- Global Cluster Disaster Recovery requires load balancer listeners on both cluster VIPs and bidirectional network access between the Primary and Standby Cluster networks. See Network Requirements.
- The platform only supports HTTPS by default. If HTTP support is required, you need to open the HTTP port for all control plane nodes.

External Platform Image Registry

For new production deployments, prepare a customer-managed external platform image registry when your organization requires independent registry availability, capacity, backup, and lifecycle management.

Before running `setup.sh`, complete Prepare an External Platform Image Registry. That page defines the required address, network access, storage, and push and pull permissions; separates those requirements from production recommendations; and explains how to upload the complete target-version payload.

Download

Before installation, download the **Core Package** and the required Aligned Extension packages separately from the **Alauda Customer Portal**.

TOC

About the Customer Portal

The **Alauda Customer Portal** is the official service and delivery portal for ACP product resources. Use it to obtain verified software packages and official technical guidance for installation, upgrade, and maintenance activities.

The Customer Portal provides access to the following resources:

- Product downloads, including installation, upgrade, and extension packages.

- Product documentation and technical guidance.
- Support resources for submitting, tracking, and managing support requests.
- Marketplace packages for extending platform capabilities.
- License-related resources for deployment and maintenance planning.

Use an authorized account to download the package required for your environment. If you do not have a registered account, contact technical support.

For customer delivery and production environments, use the package versions and documentation published on the Customer Portal as the official deployment and maintenance baseline.

Download the Installation Packages

1. Download the ACP 4.4 **Core Package** that matches your target architecture.
2. Download **Common ACP Extensions** for the same ACP version and architecture. The Violet CLI is the recommended method because it can locate and download the scenario packages directly.

Download with Violet CLI (Recommended)

Install **Violet v4.2.13 or later**, run `violet version` to verify the version, and use `violet ac login` to log in to the Alauda Customer Portal. For installation and login instructions, see Download Packages.

Set `<target-platform-version>` to the exact ACP Distribution Version of the Core Package, such as `v4.4.0`. Set `<arch>` to `amd64`, `arm64`, or `hybrid` to match the Core Package. List the scenarios first, and then download **Common ACP Extensions**:

```
violet ac scenarios --arch=<arch> --platformVersion=<target-platform-vers
ion>
violet ac download-app --arch=<arch> --platformVersion=<target-platform-v
ersion> --scenario="Common ACP Extensions"
```

Confirm that **Common ACP Extensions** appears in the scenario list before downloading it.

Download from the Customer Portal

As an alternative to Violet, in the Customer Portal, go to **Marketplace > Batch Download > Install**. On the page, select the Core Package version from **Your ACP Version**, select the matching **Architecture**, select **Common ACP Extensions** from **Scenarios**, and download the packages.

> **Separate Packages**
>
> The Core Package contains Core artifacts only. It does not contain any of the Extension packages downloaded through **Common ACP Extensions**. After extracting the Core Package, you must manually copy the required Extension packages into its `plugins/` directory before starting the installer.

Required Aligned Extensions

Copy the packages for the following applications into the `plugins/` directory of the extracted Core Package:

- **Alauda Container Platform Web Console**
- **Alauda Container Platform Web Console Central**
- **Alauda Container Platform Marketplace Web Console**
- **Alauda Container Platform Helm Application Catalog**
- **Alauda Container Platform Observability Essentials**
- **Alauda Container Platform Essentials**
- **Alauda Container Platform Cluster Essentials**
- **Alauda Container Platform GatewayAPI Plugin**

Do not copy other packages from **Common ACP Extensions** into the extracted Core Package's `plugins/` directory. Install those Extensions after the platform installation by following Extend.

Choose a Package Architecture

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, and use Extension packages that match the Core Package architecture.

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.

INFO

This page applies to nodes running a **traditional operating system** such as Kylin Linux Advanced Server, RHEL, or Ubuntu, which ACP provisions over SSH. Several of the checks below — for example the SSH user and the `/etc/ssh/sshd_config` settings — exist to keep the SSH-based node join working. If your cluster uses Alauda OS, the node preprocessing steps on this page do not apply. Follow Installing the global Cluster on Immutable Infrastructure ↗ in the Immutable Infrastructure documentation instead.

TOC

Supported OS and Kernel Versions

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

The platform enforces the version-matching granularity declared in the support list:

- **OS version**: Use the distribution release listed below. A different major or minor release is not officially supported unless it is also listed.
- **Kernel version**: Use an official kernel supplied by the operating system vendor. When the list gives an exact `x.y.z-build` value, the build suffix can vary but `x.y.z` must match. When the list gives a kernel series, use a kernel from that series that is validated for the installed ACP patch. Do not assume that an arbitrary newer kernel is supported.

INFO

- For ACP clusters running Kubernetes 1.35 or later, nodes must use cgroup v2 and Linux kernel 5.8 or later. Use only the OS and kernel combinations listed below that are validated for the installed ACP patch.
- `x86-64-v2` is a CPU instruction set baseline, but ACP does not impose it as a universal requirement on x86 nodes running a user-provided traditional operating system. The CPU must still meet the requirements of the selected operating system and the components to be deployed. If compatibility cannot be confirmed, contact technical support.

x86

Red Hat Enterprise Linux (RHEL)

- RHEL 9.6: `5.14.0-570.12.1`

Ubuntu

- Ubuntu 22.04 LTS: `5.15.0-56-generic`

Note: Ubuntu HWE (Hardware Enablement) versions are not supported.

Kylin Linux Advanced Server

- Kylin Linux Advanced Server V11: official kernel from the 6.6 series validated for the installed ACP patch

Note: Flannel is not supported on Kylin Linux Advanced Server V11.

ARM

Kylin Linux Advanced Server

- Kylin Linux Advanced Server V11: official kernel from the 6.6 series validated for the installed ACP patch

Note: ARM architecture only supports `Kunpeng 920`, and Flannel is not supported. For other CPU 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.
  - ✅ Check whether the kernel modules `ip_vs`, `ip_vs_rr`, `ip_vs_wrr`, and `ip_vs_sh` 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` parameter in `/etc/ssh/sshd_config` must be set to `no`.
  - ✅ Set the `UsePAM` parameter in `/etc/ssh/sshd_config` to `no` before adding the node, then restart `sshd`. PAM session policies (such as a forced password change, `pam_access`, `faillock`, or `pam_limits`) can otherwise block the SSH-based node join. After the node reaches the `Ready` state, you can restore `UsePAM yes`. On

SELinux-enforcing systems that use password authentication, `UsePAM no` can itself break SSH login; use key-based authentication in that case.

- ✅ `systemctl show --property=DefaultTasksMax` must return `infinity`; a low limit can make busy containers fail to create threads. If it is not `infinity`, set `DefaultTasksMax=infinity` in `/etc/systemd/system.conf` and run `systemctl daemon-reexec`.

- **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 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 `172.17.x.x` ~ `172.18.x.x` required by nerdctl 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/nerdctl`
    - `/opt/nerdctl/`
    - `/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/nerdctl/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, nerdctl, containerd, runc, podman, container network, container runtime, or Kubernetes).

The following commands can be used for reference.

RHEL

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-ap
iserver \
    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-ap
iserver \
    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 ku
bernetes-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 ku
bernetes-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.

Prepare an External Platform Image Registry

For new production deployments, you can use a customer-managed external image registry as the central image source for ACP Core, managed clusters, and Extensions. The registry must contain the complete target-version payload before you start the installer.

> **Prepare the Registry Before Installation**
>
> When an external registry is selected, the installer does not deploy the platform built-in Registry and does not copy missing images or packages into the external registry. Complete both upload modes on this page before running `setup.sh`.

TOC

Understand the Registry Roles

The following registry roles are independent:

Role	Purpose	Lifecycle owner
Platform built-in Registry	Stores platform Core images and Extension artifacts when the `global` cluster is installed with **Platform Deployment** as its image repository.	ACP deploys the service; the platform administrator operates its capacity and backup.
External platform image registry	Replaces the platform built-in Registry as the image source for Core, managed clusters, and Extensions.	You provide and operate the registry, and you upload every required target-version payload.

This page configures the platform image source used by ACP components and clusters. It does not configure an image service for application workloads.

The platform built-in Registry remains supported for existing environments and non-production evaluation. This page does not describe an online migration from the built-in Registry to an external registry.

Registry Requirements

Required Capabilities and Access

Before uploading the target-version payload, confirm that the registry:

- Supports the standard container image push and pull operations performed by the Core Package tools.
- Uses an address in the form `<host-or-IP>[:port]`, for example `registry.example.com:5000`. Do not include `http://` or `https://` in the installer registry address.
- Is reachable through the required firewalls, proxies, and network routes from the installation node and every cluster node that pulls platform images.

- Provides upload credentials that can create or update the required repository paths and push images. Credentials passed to `setup.sh` must have pull permission. The upload and pull credentials may belong to different accounts; when you authenticate `setup.sh`, provide its username and password together.
- Has enough available storage for the target-version Core and Extension payload that you upload.

Production Recommendations

For a production registry:

- Use HTTPS with a certificate that is valid for the registry address. Ensure that the installation node and every cluster node that pulls platform images trust the issuing CA.
- Use a stable DNS name that resolves from the installation node and all cluster nodes.
- Plan capacity for every required CPU architecture and for the versions retained for upgrade, rollback, and node recovery.
- Provide availability, monitoring, backup, and recovery appropriate to your environment. Registry availability is required when new Pods start, nodes are replaced, clusters are created or upgraded, and Extensions are installed or upgraded.
- Configure retention so it does not delete manifests or digests still referenced by installed platform versions, clusters, or Extensions.

> **Customer-Managed Service**
>
> ACP consumes the external registry but does not install, upgrade, back up, monitor, or repair it. Follow your registry vendor's administration documentation for those operations.

Prepare the Target-Version Payload

Use the Core Package that exactly matches the ACP Distribution Version and CPU architecture that you will install. Do not reuse a payload from another version or architecture.

1. Complete Download, extract the Core Package, and change to its `installer/` directory.

2. Copy the required Aligned Extension packages into `plugins/` as described in Installing. The `all` upload mode processes the packages currently present in this directory.
3. Set the registry address and upload credentials in the current shell. Read the password without writing it into shell history:

```
export REGISTRY_ADDRESS="registry.example.com"
export REGISTRY_USERNAME="<registry-username>"
read -rsp "Registry password: " REGISTRY_PASSWORD
export REGISTRY_PASSWORD
printf '\n'
```

   For a registry that allows anonymous push, set both `REGISTRY_USERNAME` and `REGISTRY_PASSWORD` to empty values.
4. From the extracted `installer/` directory, run both upload modes. You can run the commands in either order, but both must finish successfully:

```
bash res/upload.sh all \
  "$REGISTRY_ADDRESS" "$REGISTRY_USERNAME" "$REGISTRY_PASSWORD"

bash res/upload.sh necessary \
  "$REGISTRY_ADDRESS" "$REGISTRY_USERNAME" "$REGISTRY_PASSWORD"
```

   The modes overlap, but neither replaces the other:

   - `all` uploads the product image, the cluster version operator image, and every `*.tgz` package currently in `plugins/`. It does not upload the installer artifact.
   - `necessary` uploads the product image, the cluster version operator image, the installer artifact, and the bootstrap plugins required to create the `global` cluster. It does not upload every package in `plugins/`.
5. In the registry administration interface or API, confirm that the newly uploaded repositories and manifests are present and are not immediately eligible for cleanup.

Troubleshooting

Symptom	Likely cause	Action
`x509: certificate signed by unknown authority`	The registry CA or an intermediate certificate is not trusted.	Install the complete CA chain on the installation node and every cluster node that pulls images, then retry. Do not replace HTTPS with an insecure registry as a production workaround.
Certificate name mismatch	The registry address does not match the certificate subject alternative names.	Use the registry DNS name covered by the certificate or replace the certificate.
`unauthorized` or `authentication required`	The upload credentials are incorrect or lack push permission.	Verify the account used by `res/upload.sh` and confirm that it can push to the required repository paths.
`denied` or repository creation fails	The upload account cannot create or update the required repository paths.	Grant the account the required repository permissions and rerun both upload modes.
A required package is reported as not found	The Core Package is incomplete, the wrong version or architecture was extracted, or required Extension packages were not copied into `plugins/`.	Verify the package checksum, version, architecture, and `plugins/` contents before retrying.

Next Step

After both upload modes succeed and the registry contains the target payload, continue with Installing.

Installing

This section describes how to install ACP Core and the required Aligned Extensions, and deploy the `global` cluster.

> **Installation Path**
>
> This installation path applies to clusters running on a **traditional operating system**. If your environment uses Alauda OS, see Installing the global Cluster on Immutable Infrastructure ↗ instead.

Before starting the installation, ensure that you have completed the prerequisite checks, downloaded and verified the Core Package and required Aligned Extension packages, completed node preprocessing, and finished the other preparation work. If you selected an external platform image registry, its complete target-version payload must already be available before you run `setup.sh`.

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.
- The Core Package does not contain the required Aligned Extension packages. Prepare them separately as described in Download.

2 Prepare the Required Aligned Extension Packages

Copy each of the eight packages listed in Required Aligned Extensions into the `plugins/` directory of the extracted Core Package. Run the following command once for each package:

```
cp <path-to-extension-package> /root/cpaas-install/installer/plugins/
```

Confirm that the directory contains the eight required packages before starting the installer:

```
ls -1 /root/cpaas-install/installer/plugins/
```

Do not copy other packages from **Common ACP Extensions** into this directory. The installer processes only the packages that you manually place in `plugins/`; it does not download packages from the Customer Portal.

3 Start the Installer

Choose the command that matches the platform image source. After the installer starts successfully, the terminal outputs 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.

For the platform built-in Registry:

```
bash setup.sh
```

For an external registry that permits anonymous pull:

```
bash setup.sh --registry "<registry-address>"
```

For an authenticated external registry, provide the pull username and password together. The pull account used by `setup.sh` can be different from the push account used to upload the payload:

```
export REGISTRY_ADDRESS="registry.example.com"
export REGISTRY_USERNAME="<registry-pull-username>"
read -rsp "Registry password: " REGISTRY_PASSWORD
export REGISTRY_PASSWORD
printf '\n'

bash setup.sh \
  --registry "$REGISTRY_ADDRESS" \
  --username "$REGISTRY_USERNAME" \
  --password "$REGISTRY_PASSWORD"

unset REGISTRY_PASSWORD
```

The registry address must use the form `<host-or-IP>[:port]` without `http://` or `https://`. Do not provide only one of `--username` and `--password`.

> **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.

IP Family

```
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.

4 Parameter Configuration

Configure the installation in the Web UI, review every value, and then confirm the installation. Use Installer Parameters for the field-by-field reference.

Validate the Installation

After the installer reports that the installation is complete, validate Core and the required Aligned Extensions before continuing with post-install tasks. For the checklist and commands, see Validation.

Installer Parameter Reference

For the detailed field reference, see Installer Parameters.

Installation Troubleshooting

For installer stalls, failed resources, image-pull failures, missing artifacts, and cleanup, see Installation Troubleshooting.

Additional Resources

- Upload and Install Extensions
- Installer Parameters
- Validation
- Installation Troubleshooting
- Next Steps

Installer Parameters

Use this reference while configuring the `global` cluster in the installer Web UI. Confirm the network, access, image source, and node decisions before you submit the installation.

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 Endpoint	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. 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 Endpoint** and **Platform Access Address**, enter the same address as the **Cluster Endpoint**. 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	`Platform Deployment` uses the Registry deployed with the `global` cluster. `External` uses a customer-managed platform image registry. Its address and pull credentials must match the values passed to `setup.sh`, and the registry must already contain the complete target-version Core and Extension payload.
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, adjust it according to the page prompt and add it again.

After reviewing the parameters, return to Installing to submit the installation.

Validation

Validate ACP Core and the required Aligned Extensions after the installer completes and before you continue with post-install tasks.

TOC

Prerequisites

- The ACP installation completed.
- You can access the platform Web UI address shown by the installer.
- You can run `kubectl` from the installation node.

Validate Web UI Access

After the installation is complete, the installer displays the platform access URL. Click **Access** to open the platform Web UI and verify that the platform login page is reachable.

Validate Core Applications, Extensions, and Pods

Run the following commands on a control plane node of the new `global` cluster to confirm the installed state:

```
# All nodes are Ready
kubectl get nodes

# ClusterModule/global reports the base module as healthy
kubectl get clustermodule global -o jsonpath='{.status.phase}'

# No AppRelease is in a failed state
kubectl get apprelease -A

# Core and Aligned modules report their current phase and version
kubectl get moduleinfo -l cpaas.io/cluster-name=global

# No Pod is stuck outside Running or Completed
kubectl get pod --all-namespaces | awk '{if ($4 != "Running" && $4 != "Co
mpleted")print}' | awk -F'[/ ]+' '{if ($3 != $4)print}'
```

The installation is healthy when:

- All `global` cluster nodes are `Ready`.
- `ClusterModule/global` reports a healthy phase.
- Every AppRelease is in a non-failed state.
- The following applications are installed, and their corresponding `ModuleInfo` resources report `Running` at the target version:
  - **Alauda Container Platform Web Console**
  - **Alauda Container Platform Web Console Central**
  - **Alauda Container Platform Marketplace Web Console**
  - **Alauda Container Platform Helm Application Catalog**

- **Alauda Container Platform Observability Essentials**
- **Alauda Container Platform Essentials**
- **Alauda Container Platform Cluster Essentials**
- **Alauda Container Platform GatewayAPI Plugin**

- Critical Pods in `cpaas-system` are `Running` or `Completed`.

In the Web Console, open **Administrator > Marketplace > Cluster Plugins** and confirm that the required applications are available without an installation failure.

Validate the Platform Image Source

Confirm that `ProductBase` records the expected platform image source:

```
kubectl get productbase base \
  -o jsonpath='{.spec.registry.address}{"\t"}{.spec.registry.external}
{"\n"}'
```

For an external registry, the address must match the prepared registry and the second value must be `true`.

For an authenticated external registry, confirm that the pull Secret exists without printing its contents:

```
kubectl -n cpaas-system get secret global-registry-auth
```

Confirm that the `global` cluster records the same registry address:

```
kubectl -n cpaas-system get cluster global \
  -o jsonpath='{.metadata.annotations.cpaas\.io/registry-address}{"\n"}'
```

`ProductBase` also contains catalog entries for optional or uninstalled Extensions. Do not require every entry in `ProductBase.status.artifacts` to be `Ready`; an unrelated `Absent` entry does not by itself indicate an installation failure.

Confirm that the required Extension catalog entries exist and that Pods are not blocked by image pulls:

```
kubectl get moduleplugins

kubectl get pods --all-namespaces \
  | awk '$4 ~ /ImagePullBackOff|ErrImagePull/'
```

The Pod command must produce no output. Installation acceptance is based on the required applications, their `ModuleInfo` and `AppRelease` state, and their running Pods. If a required installed application is unavailable, follow Installation Troubleshooting to distinguish a missing manifest from DNS, routing, firewall, CA trust, pull credential, or registry availability problems.

Next Step

After validation succeeds, continue with Next Steps.

Installation Troubleshooting

Use this page when the installer does not progress, an installation resource fails, or post-install validation reports an image or artifact problem.

TOC

Common Stalls and Where to Look

Collect the signal in the second column before changing the environment or opening a support ticket.

Symptom	First place to look	What you are looking for
Web UI does not load	Terminal output of `setup.sh` and `nerdctl ps` on the installation node	The `minialauda-control-plane` container is Running and port `8080` is reachable from the browser.
Installer log stops advancing	`tail -f /var/cpaas/data/installer.log`	The last completed phase and the first repeated or failed operation.
Stuck on control plane provisioning	`kubectl get machines -A` and `kubectl describe` on a machine that is not `Ready`	The `Bootstrap` and `Infrastructure` conditions and the node address in `status.addresses`.
Stuck on network and Core add-ons	`kubectl -n kube-system get pods` on the new cluster	Kube-OVN, CoreDNS, and kube-proxy Pods are `Running`; image-pull failures usually identify a registry path problem.
AppRelease shows `Failed`	`kubectl describe apprelease <name> -n <namespace>`	Status conditions and recent Events that identify the chart or dependency failure.
`ClusterModule/global` does not become healthy	`kubectl describe clustermodule global`	The status condition that identifies the blocking Core or Aligned module.

Image Pull and Artifact Failures

Symptom	Checks	Recovery
Pod is `ImagePullBackOff` or `ErrImagePull`	Run `kubectl describe pod <pod> -n <namespace>` and test the configured registry from the failing node.	Correct the reported DNS, route, firewall, CA trust, pull credential, registry availability, or missing-manifest problem.
A required Core or Aligned application is unavailable after installation	Inspect the corresponding `AppRelease`, `ModuleInfo`, `ModulePlugin`, and Pod Events. If package availability is still unclear, inspect the matching entry in `kubectl get productbase base -o yaml` and the exact `repository:tag` in the registry.	If the registry confirms that the required manifest is missing, upload the matching installation-version package. For authorization, CA, DNS, routing, or connectivity errors, correct the reported registry problem instead of republishing the package. Ignore unrelated `Absent` entries for optional or uninstalled catalog packages.
A recovered or replacement node cannot pull an installed-version image	Inspect the failing Pod and registry retention records.	Restore the referenced digest, then correct registry availability, backup, or retention policy before retrying recovery.

For upload-command failures that occur before the installer starts, use External Platform Image Registry Troubleshooting.

Installer Cleanup

The installer normally removes itself after installation. If the installer container remains 30 minutes after installation completes, run this command on the installation node:

```
nerdctl rm -f minialauda-control-plane
```

Collect Evidence Before Escalation

Capture `/var/cpaas/data/installer.log`, the relevant `kubectl describe` output, failing resource conditions, and the exact Core Package version and architecture. Do not include registry passwords or Secret contents.

Next Steps

After Core and the required Aligned Extensions pass validation, complete the following post-install tasks according to your scenario.

Priority	Task	When to use	Continue with
Recommended immediately	Install the Product Docs plugin.	Install this plugin so in-console help links can open product documentation.	Install Product Docs Plugin
Conditional	Create workload clusters or connect existing clusters.	Use this path when your architecture requires workload clusters separate from the `global` cluster.	Clusters
Conditional	Upload and install additional Extensions Packages.	Use this path for packages from **Common ACP Extensions** that were not part of the eight required installation packages, or when you need other platform capabilities.	Extend
Recommended before broad user onboarding	Configure identity providers, users, roles, projects, and project membership.	Use this path before onboarding administrators, project users, or application teams.	Users and Roles and Projects

Priority	Task	When to use	Continue with
Recommended before application workloads	Prepare storage, networking, registry access, and workload prerequisites.	Use this path before application teams deploy workloads.	Storage, Networking, Registry, and Clusters
Recommended before production operation	Review backup, upgrade, monitoring, and operational documentation.	Use this path before operating the platform for production workloads.	Backup, Upgrade, and Observability

These tasks do not all apply to every installation. Follow the paths that match your deployment architecture and operational requirements.

TOC

Install Product Docs Plugin

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

1 Open Cluster Plugins

Navigate to **Administrator**. In the left sidebar, click **Marketplace** > **Cluster Plugins** and select the `global` cluster.

2 Install the Plugin

Locate the **Alauda Container Platform Product Docs** plugin and click **Install**.