Upgrade

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

Overview

ACP 4.3 uses a Cluster Version Operator (CVO)-based workflow for cluster upgrades.

In the Web Console, the upgrade request now follows a two-step flow: review RPCH items first, and then submit the upgrade request in a separate confirmation step.

When moving the platform to a new ACP Distribution Version, the upgrade normally proceeds in two stages:

1. Upgrade the global tier to the target Distribution Version by following the validated global-cluster procedure, including artifact preparation and preflight checks.
2. After the global tier reaches the target Distribution Version, upgrade workload clusters from the supported workload-cluster entry point and observe cluster status until each target cluster reaches the same Distribution Version.

A workload cluster can be upgraded only to a Distribution Version that the global tier has already reached. In environments with **global disaster recovery (DR)**, this means both the standby and primary global clusters must reach the target Distribution Version before workload clusters are upgraded to that Distribution Version. This sequencing rule does not replace the Compatible Versions prerequisite: before the global tier is upgraded to ACP 4.3, workload clusters must remain within the ACP 4.3 compatible Kubernetes version range.

TOC

Key Concepts

- **ClusterVersionShadow (`cvsh`)**: The upgrade resource used to track current version, desired version, preflight results, execution stages, and history.
- **Distribution Version**: The ACP version currently reached by a cluster. A workload cluster can be upgraded only to a Distribution Version that the global tier has already reached.
- **Preflight**: Validation checks that run before the upgrade starts applying the target version. For workload clusters, review preflight results from the upgrade status output after the upgrade request is submitted.
- **Available upgrade targets**: The upgrade versions that are currently offered for a cluster. In the Web Console, the target version for the current upgrade flow is determined by the platform.
- **upgrade.sh**: The preparation script that uploads artifacts and deploys or updates the cluster version operator before the upgrade proceeds.
- **Global DR environment**: An environment with both a primary global cluster and a standby global cluster.
- **Primary global cluster**: The global cluster that the platform access domain currently resolves to.
- **Standby global cluster**: The other global cluster in the DR pair. After a failover, the roles of the two clusters are swapped.

Upgrade Entry Points

- **Global clusters**: Follow the validated `upgrade.sh`-based procedure. After the preparation phase completes, request the upgrade from the Web Console through the two-step RPCH review flow, use ACP CLI, or update `ClusterVersionShadow.spec.desiredUpdate` directly.
- **Workload clusters**: Use the Web Console through the two-step RPCH review flow or use ACP CLI after the target Distribution Version becomes available for the workload cluster.

Post-upgrade Security Hardening

After the global cluster and all workload clusters have reached ACP 4.3, complete PKCE security hardening by following Disabling the PKCE Plain Method.

After the global cluster reaches ACP 4.3, complete the required L5 plugin compatibility upgrades in Upgrade the global cluster.

Related Documentation

- Pre-upgrade
- Upgrade the global cluster
- Upgrade workload clusters
- Global Cluster Disaster Recovery

Pre-Upgrade Preparation

Supported upgrade paths:

- **From** `4.0` → `4.3`
- **From** `4.1` → `4.3`
- **From** `4.2` → `4.3`

Before starting, ensure your current platform version is within the supported upgrade range.

TOC

Important Notes

- Ensure the directory `/cpaas/minio` on the control plane nodes of global cluster has at least **120 GB** of free disk space.
- Before upgrading the global tier to ACP 4.3, all workload clusters must remain within the ACP 4.3 **Compatible Versions** documented in Kubernetes Support Matrix.
- If any workload cluster is outside that compatible range, upgrade that workload cluster first until it enters the ACP 4.3 compatible range before upgrading the global tier.
- A workload cluster can be upgraded only to a Distribution Version that the global tier has already reached.

Download the Packages for Offline Environments

From the **Alauda Customer Portal**, download the **ACP Core Package**.

If you want to upgrade cluster **Extensions** during the upgrade, follow these steps:

1. Navigate to the following path: [Marketplace - Batch Download - Upgrade - Post-ACP v4.0 Upgrades]
2. Download the `ac-get-app.sh` .
3. Upload the script to the control node of **Global** cluster in your environment.
4. Run the script with `bash ac-get-app.sh`.
5. After it finishes, import the generated `apps.yaml` back into the Alauda Customer Portal to align the extensions list.

In addition, navigate to the **CLI Tools** section in the **Alauda Customer Portal** and download the `violet` tool. This tool is required for uploading Extensions. For more information about `violet`, see Upload Packages.

WARNING

If you are upgrading **from ACP 4.0 to ACP 4.3** and **Alauda Build of TopoLVM** is installed on any target clusters, upload the TopoLVM package to those clusters before you proceed with the upgrade. This step is not required when upgrading from ACP 4.1 or ACP 4.2. You can specify multiple target clusters in `--clusters`, separated by commas.

```
violet push <path/to/directory/only_put_topolvm_plugin_here> \
  --target-catalog-source "platform" \
  --platform-address "https://example.com" \
  --platform-username "<platform_user>" \
  --platform-password "<platform_password>" \
  --clusters "cluster-a,cluster-b"
```

WARNING

Starting with v4.2, we introduced a new plugin named **Alauda Container Platform Log Essentials**. If you previously installed the log storage plugin, you also have to upload that plugin before starting the upgrade.

Upgrade the global cluster

ACP consists of a **global** cluster and one or more workload clusters. To move the platform to a new ACP Distribution Version, upgrade the global tier to the target Distribution Version first, and then upgrade workload clusters to that same Distribution Version.

ACP 4.3 uses a CVO-based workflow for cluster upgrades. A typical `global` cluster upgrade includes artifact preparation, preflight checks, upgrade request, and status observation.

Before upgrading the `global` cluster to ACP 4.3, verify that every workload cluster is on a compatible Kubernetes version. For ACP 4.3, the compatible versions are 1.34, 1.33, 1.32, and 1.31. This prerequisite is separate from the broader third-party cluster management range.

This Compatible Versions prerequisite applies whether or not the environment uses global DR. Global DR changes the procedure used to upgrade the global tier, but it does not change the requirement that workload clusters must remain within the compatible Kubernetes version range before the global tier is upgraded to the target Distribution Version.

Global cluster upgrades follow the validated `upgrade.sh`-based procedure documented on this page. You can request the global-cluster upgrade from the Web Console, by updating `ClusterVersionShadow.spec.desiredUpdate`, or by using ACP CLI with `--cluster=global`. For the complete AC CLI workflow and output interpretation, see Upgrading Clusters. For full command and flag syntax, see AC CLI Administrator Command Reference.

If the environment uses **global DR**, follow the Global DR Procedure. Otherwise, follow the standard workflow below.

TOC

Standard Workflow

1 Prepare upgrade artifacts

Run `bash upgrade.sh` from the extracted core package directory.

`upgrade.sh` prepares the resources required by the CVO-based workflow, including:

Type	Content	Purpose
Product images	`product-image`	Used to resolve the target version and image in `ProductManifest` and CVO.
CVO image	`cluster-version-operator`	Used to deploy or update the cluster version operator.

Type	Content	Purpose
Plugin artifacts	`plugins/*.tgz`	Used by the upgrade plan when plugin artifacts are required.

Registry behavior depends on how the environment is configured:

Scenario	Behavior
`--registry` is specified	Use the provided registry directly.
`--registry` is not specified	Read the registry address from `ProductBase.spec.registry.address`.
Built-in platform registry	Rebuild the access address by using the global VIP.
External registry	Automatically set `SKIP_SYNC_IMAGE=true` and skip image synchronization.
Image upload required but credentials omitted	Read `username` and `password` from the `cpaas-system/registry-admin` Secret.

Common parameters:

Parameter	Purpose
`--registry`	Specify the target registry address.
`--username` / `--password`	Specify registry credentials.
`--only-sync-image`	Synchronize images and plugin artifacts only.
`--skip-sync-image`	Skip image and plugin synchronization.
`--skip-check-artifacts`	Skip artifact validation.

```
bash upgrade.sh
```

WARNING

- Do not continue to the next step until image and plugin synchronization is complete.
- Use `--only-sync-image` only when you want artifact synchronization without further preparation.
- Use `--skip-sync-image` only when the required images and plugin artifacts have already been uploaded.

2 Run preflight checks

Run preflight before requesting the upgrade:

```
bash upgrade.sh --preflight
```

Preflight returns two parts:

Output	Purpose
`Summary`	Shows the overall result, current version, desired version, and desired image.
`Checks`	Shows the result of each individual validation item.

The default check set includes:

- `ResourcePatchUpgradeable`
- `ClusterVersionUpgradeable`
- `VersionUpgradePath`
- `KubernetesVersionSupported`
- `DockerRuntimeUnsupported`
- `ClusterRunning`
- `ClusterModuleStable`
- `ControlPlaneStaticPodsPresent`

- `CustomEtcdBackupCronJobsAbsent`
- `CRIUpgradePodsAbsent`
- `ModuleInfoStable`
- `PlatformLicense`

3 Handle preflight blocks when needed

If `ResourcePatchUpgradeable` fails with `reason=UnexemptResourcePatches`, inspect the blocking `ResourcePatch` and add the required exemption annotation:

```
kubectl -n cpaas-system get cvsh global \
  -o jsonpath='{range .status.preflight.checks[?(@.name=="ResourcePat
chUpgradeable")]}{.state}{"\t"}{.reason}{"\t"}{.message}{"\n"}{end}'

kubectl get resourcepatches <rp-name> -o yaml
```

The default annotation key is `config.cpaas.io/exempt-for-ver`.

```
kubectl annotate resourcepatches <rp-name> \
  config.cpaas.io/exempt-for-ver=4.3.0 \
  --overwrite
```

If temporary troubleshooting requires specific checks to be disabled, configure `cpaas-system/cvo-config`:

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: cvo-config
  namespace: cpaas-system
data:
  preflight: |
    disabled:
      - ResourcePatchUpgradeable
      - VersionUpgradePath
```

4 Request the upgrade

After the preparation phase completes, choose one of the following entry points:

- Use the Web Console after the target version becomes available for the cluster.
- Patch `ClusterVersionShadow.spec.desiredUpdate` directly when you need to operate the underlying CVO resource.
- Use ACP CLI to request the upgrade for `global` explicitly.

If you use the Web Console, the request follows a two-step flow:

- In **Step 1**, review the RPCH list.
- Click **Acknowledge** to continue to **Step 2**.
- In **Step 2**, review **Current Version** and **Target Version**. The page does not display a plugin list or a warning panel at this stage.
- The target version is determined by the prepared upgrade artifacts and cannot be selected manually in the Web Console.
- Click **Start Upgrade**.
- Confirm the action in the dialog.
- After confirmation, the page shows that the upgrade request has been submitted and the action enters an in-progress state.

`kubectl` example:

```
kubectl patch cvsh global -n cpaas-system --type merge -p '{
  "spec": {
    "desiredUpdate": {
      "version": "4.3.0"
    }
  }
}'
```

You can also edit the resource directly:

```
kubectl edit cvsh global -n cpaas-system
```

Minimum configuration:

```
spec:
  desiredUpdate:
    version: 4.3.0
```

ACP CLI example:

```
# Request upgrade to the highest version currently published in availableUpdates
ac adm upgrade --cluster=global --to-latest

# Request upgrade to a specific target version
ac adm upgrade --cluster=global --to=4.3.0

# Show summary, preflight, and stage progress for the global cluster upgrade
ac adm upgrade status --cluster=global
```

5 Observe execution

Use the following command to inspect the overall status:

```
kubectl get cvsh -n cpaas-system
```

Important status fields:

Field	Purpose
`status.conditions`	Overall status entry point.
`status.preflight.observedAt`	Time of the latest preflight run.
`status.preflight.checks`	Detailed result of each preflight item.
`status.current`	Current applied version and image.
`status.desired`	Target version and image being reconciled.

Field	Purpose
`status.history`	Upgrade history, newest first.
`status.stages`	Upgrade stages and per-stage execution state.

Focus on these conditions first:

Condition	Interpretation
`PreflightReady`	`True` means preflight passed.
`Ready`	`True` means the cluster has reached the desired version.
`Reconciling`	`True` means the upgrade is still running.
`Stalled`	`True` means the upgrade is blocked and requires intervention.

Useful diagnostics:

```
kubectl -n cpaas-system get cvsh global \
  -o jsonpath='{range .status.conditions[*]}{.type}{"\t"}{.status}
{"\t"}{.reason}{"\t"}{.message}{"\n"}{end}'

kubectl -n cpaas-system get cvsh global \
  -o jsonpath='{.status.preflight.observedAt}{"\n"}{range .status.pre
flight.checks[*]}{.name}{"\t"}{.policy}{"\t"}{.state}{"\t"}{.reason}
{"\t"}{.message}{"\n"}{end}'

kubectl -n cpaas-system get cvsh global \
  -o jsonpath='{range .status.history[*]}{.version}{"\t"}{.state}
{"\t"}{.startedTime}{"\t"}{.completionTime}{"\n"}{end}'
```

6 (Conditional) Upgrade Alauda Service Mesh Essentials

If **Service Mesh v1** is installed, refer to the Alauda Service Mesh Essentials Cluster Plugin ↗ documentation before upgrading the workload clusters.

Post-upgrade

- Upgrade Alauda AI ↗
- Upgrade Alauda DevOps ↗
- After all workload clusters have also reached ACP 4.3, complete PKCE hardening by following Disabling the PKCE Plain Method.
- ACP 4.3 fixes an API authentication issue where specific APIs could previously be accessed without authentication. After the global cluster reaches ACP 4.3, upgrade the following L5 plugins to ACP v4.3-compatible versions. Otherwise, their UI pages may fail to open:
  - `Alauda DevOps v3`
  - `Alauda AI Essentials`
  - `Alauda Hyperflux`
  - `Alauda Container Platform Data Services Essentials`
- After upgrading the plugins, verify that each listed plugin UI page opens successfully.

Global DR Procedure

Use this procedure when the environment includes both a primary global cluster and a standby global cluster. The DR-specific steps below are in addition to the standard CVO workflow.

1 Verify the DR environment before upgrading

Follow your regular global DR inspection procedures to ensure that data in the **standby global cluster** is consistent with the **primary global cluster**. For background on the DR topology and synchronization workflow, see Global Cluster Disaster Recovery.

If inconsistencies are detected, contact technical support before proceeding.

On **both** global clusters, run the following command to ensure no `Machine` nodes are in a non-running state:

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

If any such nodes exist, resolve them before continuing.

2 Uninstall the etcd synchronization plugin from the standby global cluster

1. Access the Web Console of the **standby global cluster** through its IP or VIP.
2. Switch to **Administrator** view.
3. Navigate to **Marketplace > Cluster Plugins** and select the `global` cluster.
4. Find **Alauda Container Platform etcd Synchronizer** and uninstall it.
5. Wait for the uninstallation to complete before proceeding.

3 Prepare upgrade artifacts on both global clusters

Complete **Prepare upgrade artifacts** in the standard workflow on **both** the standby global cluster and the primary global cluster.

Use the same preparation mode on both clusters.

4 Upgrade the standby global cluster

If you will use the Web Console on the standby global cluster, verify that the standby cluster `ProductBase` includes the standby VIP in `spec.alternativeURLs`:

```
apiVersion: product.alauda.io/v1alpha2
kind: ProductBase
metadata:
  name: base
spec:
  alternativeURLs:
    - https://<standby-cluster-vip>
```

After preparation completes, run the remaining steps from the standard workflow on the **standby global cluster**:

1. **Run preflight checks**
2. **Request the upgrade**
3. **Observe execution** until the standby global cluster reaches the desired version

5 Upgrade the primary global cluster

After the standby global cluster has reached the desired version, run the remaining steps from the standard workflow on the **primary global cluster**:

1. **Run preflight checks**
2. **Request the upgrade**
3. **Observe execution** until the primary global cluster reaches the desired version

6 Reinstall the etcd synchronization plugin and verify sync status

Before reinstalling the plugin, verify that port `2379` is forwarded correctly from both global-cluster VIPs to their control plane nodes when that forwarding mode is used. Port forwarding through a load balancer is not required if the standby global cluster can access the active global cluster directly.

To reinstall the plugin:

1. Access the **standby global cluster** Web Console through its VIP and switch to **Administrator** view.
2. Navigate to **Marketplace > Cluster Plugins** and select the `global` cluster.
3. Find **Alauda Container Platform etcd Synchronizer**, click **Install**, and configure the required parameters.

When you configure the plugin:

- When port `2379` is not forwarded through a load balancer, set **Active Global Cluster ETCD Endpoints** correctly.
- Use the default value of **Data Check Interval**.
- Leave **Print detail logs** disabled unless you are troubleshooting.

Verify the sync Pod is running on the standby global cluster:

```
kubectl get po -n cpaas-system -l app=etcd-sync
etcd_sync_pod=$(kubectl get po -n cpaas-system -l app=etcd-sync -o js
onpath='{.items[0].metadata.name}')
kubectl logs -n cpaas-system "$etcd_sync_pod" | grep -i "Start Sync u
pdate"
```

Once `Start Sync update` appears, recreate one of the Pods to trigger synchronization of resources with ownerReference dependencies:

```
etcd_sync_pod=$(kubectl get po -n cpaas-system -l app=etcd-sync -o js
onpath='{.items[0].metadata.name}')
kubectl delete po -n cpaas-system "$etcd_sync_pod"
```

Check sync status:

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

Output interpretation:

- `LOCAL ETCD missed keys`: Keys exist in the primary global cluster but are missing from the standby. This often resolves after restarting one `etcd-sync` Pod.
- `LOCAL ETCD surplus keys`: Keys exist in the standby global cluster but not in the primary. Review these with your operations team before deleting them.

Related Documentation

Upgrade Workload Clusters

Workload clusters can be upgraded after the global tier has already reached the target ACP Distribution Version. The global tier does not need to be upgraded again if it is already at that Distribution Version.

ACP 4.3 uses the same CVO-based workflow for workload clusters as for the `global` cluster: confirm prerequisites, run preflight checks, request the upgrade, and observe execution. This workflow has two additional rules:

- If the platform uses **global DR**, both the standby and primary global clusters must reach the target Distribution Version before any workload cluster is upgraded to that Distribution Version.
- The target version normally becomes available for a workload cluster only after the global tier has already reached that same Distribution Version.

TOC

Workflow

1 Confirm workload-cluster prerequisites

Before requesting the upgrade, verify that:

- The workload cluster version is lower than the current `global` cluster version.
- The global tier has already reached the target ACP Distribution Version.
- In global DR environments, both the standby and primary global clusters have already reached that target Distribution Version.

2 Run preflight checks

Run preflight before requesting the upgrade:

```
bash upgrade.sh --cluster=<cluster> --preflight
```

Preflight returns two parts:

Output	Purpose
`Summary`	Shows the overall result, current version, desired version, and desired image.
`Checks`	Shows the result of each individual validation item.

If preflight does not pass, do not submit the upgrade request until all blocking items are resolved.

For preflight block handling patterns, see Handle preflight blocks when needed.

3 Request the upgrade

After preflight passes, choose one of the following entry points:

- Start the upgrade from the workload cluster in the Web Console.

- Use ACP CLI when your current ACP session can access the target workload cluster. To avoid ambiguity, specify the target cluster explicitly with `--cluster`.

If you are new to ACP CLI, see Getting Started with ACP CLI. For session and cluster selection, see Managing CLI Profiles.

If you use the Web Console, the request follows a two-step flow:

- In **Step 1**, review the RPCH list.
- Click **Acknowledge** to continue to **Step 2**.
- In **Step 2**, review **Current Version** and **Target Version**. The page does not display a plugin list or a warning panel at this stage.
- The target version is the current `global` cluster version and cannot be selected manually in the Web Console.
- Click **Start Upgrade**.
- After the request is submitted, the page shows that the upgrade request has been submitted and the action enters an in-progress state.

ACP CLI example:

```
# Request upgrade to the highest version currently published in availableUpdates
ac adm upgrade --cluster=<cluster> --to-latest

# Request upgrade to a specific target version
ac adm upgrade --cluster=<cluster> --to=4.3.0
```

ACP CLI submits the requested target for the specified workload cluster. Whether the upgrade can proceed is still determined by the cluster's available upgrade targets and the upgrade controller's preflight checks. ACP CLI is not used to upgrade the `global` cluster.

For the complete AC CLI upgrade workflow (metadata preparation, available updates, request modes, and status interpretation), see Upgrading Clusters. For full command and flag syntax, see AC CLI Administrator Command Reference.

4 Observe progress

After the upgrade request is submitted, use `cvsh.status` to track current version, desired version, preflight results, stages, and history:

```
kubectl get cvsh <cluster> -n cpaas-system
kubectl get cvsh <cluster> -n cpaas-system -o yaml
```

If your current ACP context points to the target workload cluster, you can also check the CLI-reported status:

```
# Show summary, preflight, and stage progress for the target cluster
upgrade
ac adm upgrade status
```

For detailed semantics of `ac adm upgrade status` output (preflight and stage interpretation), see Upgrading Clusters.

When troubleshooting, inspect conditions, preflight details, and history first:

```
kubectl -n cpaas-system get cvsh <cluster> \
  -o jsonpath='{range .status.conditions[*]}{.type}{"\t"}{.status}
{"\t"}{.reason}{"\t"}{.message}{"\n"}{end}'

kubectl -n cpaas-system get cvsh <cluster> \
  -o jsonpath='{.status.preflight.observedAt}{"\n"}{range .status.pre
flight.checks[*]}{.name}{"\t"}{.policy}{"\t"}{.state}{"\t"}{.reason}
{"\t"}{.message}{"\n"}{end}'

kubectl -n cpaas-system get cvsh <cluster> \
  -o jsonpath='{range .status.history[*]}{.version}{"\t"}{.state}
{"\t"}{.startedTime}{"\t"}{.completionTime}{"\n"}{end}'
```

After All Workload Clusters Reach ACP 4.3

After all workload clusters reach ACP 4.3, complete PKCE security hardening by following Disabling the PKCE Plain Method.

Related Documentation

- Overview
- Upgrade the global cluster
- Upgrading Clusters