升级

本文档将提供关于升级 ACP 的所有信息。

概览

ACP 4.3 使用基于 Cluster Version Operator (CVO) 的工作流进行集群升级。

在 Web Console 中，升级请求现在遵循两步流程：先审查 RPCH 项，随后在单独的确认步骤中提交升级请求。

当将平台迁移到新的 ACP Distribution Version 时，升级通常分两个阶段进行：

1. 按照经过验证的 global 集群操作步骤，将 global 层升级到目标 Distribution Version，包括 artifact 准备和 preflight 检查。
2. 在 global 层达到目标 Distribution Version 之后，从受支持的业务集群入口点升级业务集群，并观察集群状态，直到每个目标集群达到相同的 Distribution Version。

业务集群只能升级到 global 层已经达到的 Distribution Version。对于启用了 **global disaster recovery (DR)** 的环境，这意味着在业务集群升级到该 Distribution Version 之前，备用 global 集群和主 global 集群都必须先达到目标 Distribution Version。此排序规则不会替代 Compatible Versions 前提：在 global 层升级到 ACP 4.3 之前，业务集群必须保持在 ACP 4.3 兼容的 Kubernetes 版本范围内。

目录

关键概念

- **ClusterVersionShadow (`cvsh`)**：用于跟踪当前版本、目标版本、preflight 结果、执行阶段和历史记录的升级资源。
- **Distribution Version**：集群当前达到的 ACP 版本。业务集群只能升级到 global 层已经达到的 Distribution Version。
- **Preflight**：在升级开始应用目标版本之前运行的验证检查。对于业务集群，请在提交升级请求后，从升级状态输出中查看 preflight 结果。
- **Available upgrade targets**：当前为集群提供的升级版本。在 Web Console 中，当前升级流程的目标版本由平台确定。
- **upgrade.sh**：用于上传 artifact 并在升级继续之前部署或更新 cluster version operator 的准备脚本。
- **Global DR environment**：同时包含主 global 集群和备用 global 集群的环境。
- **Primary global cluster**：平台访问域当前解析到的 global 集群。
- **Standby global cluster**：DR 对中的另一个 global 集群。发生故障切换后，这两个集群的角色会互换。

升级入口点

- **Global clusters**：遵循经过验证的 `upgrade.sh` 基础操作步骤。在准备阶段完成后，可通过 Web Console 中的两步 RPCH 审核流程发起升级、使用 ACP CLI，或直接更新 `ClusterVersionShadow.spec.desiredUpdate`。
- **Workload clusters**：在目标 Distribution Version 对业务集群可用后，可通过 Web Console 中的两步 RPCH 审核流程或使用 ACP CLI。

升级后安全加固

在 global 集群和所有业务集群都达到 ACP 4.3 后，请按照 Disabling the PKCE Plain Method 完成 PKCE 安全加固。

在 global 集群达到 ACP 4.3 后，请在 Upgrade the global cluster 中完成所需的 L5 plugin 兼容性升级。

相关文档

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

升级前准备

支持的升级路径：

- 从 `4.0` → `4.3`
- 从 `4.1` → `4.3`
- 从 `4.2` → `4.3`

开始之前，请确保您当前的平台版本在支持的升级范围内。

目录

重要说明

- 确保 global 集群控制平面节点上的目录 `/cpaas/minio` 至少有 **120 GB** 的可用磁盘空间。
- 在将 global tier 升级到 ACP 4.3 之前，所有业务集群必须保持在 Kubernetes Support Matrix 中记录的 ACP 4.3 兼容版本 范围内。
- 如果有任何业务集群不在该兼容范围内，请先升级该业务集群，直到其进入 ACP 4.3 兼容范围后，再升级 global tier。
- 业务集群只能升级到 global tier 已经达到的 Distribution Version。

离线环境下载软件包

从 灵雀云 **Customer Portal** 下载 **ACP Core Package**。

如果您想在升级过程中升级集群 **Extensions**，请按以下步骤操作：

1. 进入路径：[Marketplace - Batch Download - Upgrade - Post-ACP v4.0 Upgrades]
2. 下载 `ac-get-app.sh` 脚本。
3. 将脚本上传到您环境中 **Global** 集群的控制节点。
4. 使用 `bash ac-get-app.sh` 运行该脚本。
5. 完成后，将生成的 `apps.yaml` 导入回 灵雀云 Customer Portal，以同步 Extensions 列表。

此外，进入 灵雀云 **Customer Portal** 的 **CLI Tools** 部分，下载 `violet` 工具。该工具用于上传 Extensions。有关 `violet` 的更多信息，请参见 Upload Packages。

WARNING

如果您正在从 **ACP 4.0** 升级到 **ACP 4.3**，且任何目标集群上安装了 灵雀云 **Build of TopoLVM**，请在升级前将 TopoLVM 软件包上传到这些集群。升级自 ACP 4.1 或 ACP 4.2 时无需此步骤。您可以在 `--clusters` 中指定多个目标集群，使用逗号分隔。

```
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

从 v4.2 开始，我们引入了一个名为 **Alauda Container Platform Log Essentials** 的新插件。如果您之前安装了日志存储插件，升级前也必须上传该插件。

升级 global 集群

ACP 由一个 **global** 集群和一个或多个业务集群组成。要将平台迁移到新的 ACP Distribution Version，请先将 global 层升级到目标 Distribution Version，然后再将业务集群升级到相同的 Distribution Version。

ACP 4.3 使用基于 CVO 的集群升级流程。典型的 `global` 集群升级包括工件准备、预检、升级请求和状态观察。

在将 `global` 集群升级到 ACP 4.3 之前，请确认每个业务集群都运行在兼容的 Kubernetes 版本上。对于 ACP 4.3，兼容版本为 1.34、1.33、1.32 和 1.31。该前置条件独立于更广泛的第三方集群管理范围。

无论环境是否使用 global DR，此兼容版本前置条件都适用。global DR 会改变升级 global 层所使用的操作步骤，但不会改变在将 global 层升级到目标 Distribution Version 之前，业务集群必须保持在兼容的 Kubernetes 版本范围内这一要求。

global 集群升级遵循本页记录的已验证 `upgrade.sh` 操作步骤。你可以通过 Web Console、更新 `ClusterVersionShadow.spec.desiredUpdate`，或使用带 `--cluster=global` 的 ACP CLI 来发起 global 集群升级。有关完整的 AC CLI 工作流和输出解读，请参见 升级集群。有关完整命令和标志语法，请参见 AC CLI 管理员命令参考。

如果环境使用 **global DR**，请遵循 Global DR 操作步骤。否则，请遵循下面的标准工作流。

目录

标准工作流

1 准备升级工件

在解压后的核心包目录中运行 `bash upgrade.sh`。

`upgrade.sh` 会准备基于 CVO 的工作流所需的资源，包括：

类型	内容	用途
产品镜像	`product-image`	用于解析 `ProductManifest` 和 CVO 中的目标版本和镜像。
CVO 镜像	`cluster-version-operator`	用于部署或更新 cluster version operator。
插件工件	`plugins/*.tgz`	当升级计划需要插件工件时使用。

Registry 行为取决于环境配置方式：

场景	行为
指定了 `--registry`	直接使用提供的 registry。
未指定 `--registry`	从 `ProductBase.spec.registry.address` 读取 registry 地址。
内置平台 registry	使用 global VIP 重建访问地址。
外部 registry	自动设置 `SKIP_SYNC_IMAGE=true` 并跳过镜像同步。
需要上传镜像但未提供凭据	从 `cpaas-system/registry-admin` Secret 中读取 `username` 和 `password`。

常用参数：

参数	用途
`--registry`	指定目标 registry 地址。
`--username` / `--password`	指定 registry 凭据。
`--only-sync-image`	仅同步镜像和插件工件。
`--skip-sync-image`	跳过镜像和插件同步。
`--skip-check-artifacts`	跳过工件验证。

```
bash upgrade.sh
```

WARNING

- 在镜像和插件同步完成之前，不要继续执行下一步。
- 仅在你只需要工件同步而不需要进一步准备时，才使用 `--only-sync-image`。
- 仅在所需镜像和插件工件已经上传完成时，才使用 `--skip-sync-image`。

2 运行预检

在请求升级之前先运行预检：

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

预检返回两部分：

输出	用途
`Summary`	显示总体结果、当前版本、目标版本和目标镜像。
`Checks`	显示每个单独校验项的结果。

默认检查集包括：

- `ResourcePatchUpgradeable`
- `ClusterVersionUpgradeable`
- `VersionUpgradePath`
- `KubernetesVersionSupported`
- `DockerRuntimeUnsupported`
- `ClusterRunning`
- `ClusterModuleStable`
- `ControlPlaneStaticPodsPresent`
- `CustomEtcdBackupCronJobsAbsent`
- `CRIUpgradePodsAbsent`
- `ModuleInfoStable`
- `PlatformLicense`

3 在需要时处理预检阻塞

如果 `ResourcePatchUpgradeable` 因 `reason=UnexemptResourcePatches` 而失败，请检查阻塞的 `ResourcePatch`，并添加所需的豁免注解：

```
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
```

默认注解键为 `config.cpaas.io/exempt-for-ver`。

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

如果临时排障需要禁用特定检查，请配置 `cpaas-system/cvo-config`：

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

4 请求升级

准备阶段完成后，选择以下入口之一：

- 在目标版本对该集群可用后，使用 Web Console。
- 当你需要操作底层 CVO 资源时，直接 patch `ClusterVersionShadow.spec.desiredUpdate`。
- 使用 ACP CLI 显式请求 `global` 的升级。

如果你使用 Web Console，请求流程分为两步：

- 在 步骤 **1** 中，检查 RPCH 列表。

- 单击 确认 以继续到 步骤 **2**。
- 在 步骤 **2** 中，检查 当前版本 和 目标版本。此阶段页面不会显示插件列表或警告图表。
- 目标版本由已准备好的升级工件决定，无法在 Web Console 中手动选择。
- 单击 开始升级。
- 在对话框中确认操作。
- 确认后，页面会显示升级请求已提交，且操作进入进行中状态。

`kubectl` 示例：

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

你也可以直接编辑资源：

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

最小配置：

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

ACP CLI 示例：

```
# 请求升级到当前已发布在 availableUpdates 中的最高版本
ac adm upgrade --cluster=global --to-latest

# 请求升级到指定目标版本
ac adm upgrade --cluster=global --to=4.3.0

# 显示 global 集群升级的摘要、预检和阶段进度
ac adm upgrade status --cluster=global
```

5 观察执行情况

使用以下命令检查总体状态：

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

重要状态字段：

字段	用途
`status.conditions`	总体状态入口。
`status.preflight.observedAt`	最近一次预检运行时间。
`status.preflight.checks`	每个预检项的详细结果。
`status.current`	当前已应用的版本和镜像。
`status.desired`	正在协调的目标版本和镜像。
`status.history`	升级历史，最新的在前。
`status.stages`	升级阶段及每个阶段的执行状态。

首先重点关注以下条件：

条件	解释
`PreflightReady`	`True` 表示预检已通过。

条件	解释
`Ready`	`True` 表示集群已达到目标版本。
`Reconciling`	`True` 表示升级仍在运行。
`Stalled`	`True` 表示升级被阻塞且需要干预。

有用的诊断命令：

```
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 （条件）升级 灵雀云 Service Mesh Essentials

如果已安装 **Service Mesh v1**，请在升级业务集群之前先参考 Alauda Service Mesh Essentials 集群插件↗ 文档。

升级后

- 升级 Alauda AI↗
- 升级 Alauda DevOps↗
- 在所有业务集群也都升级到 ACP 4.3 之后，请按照 禁用 PKCE Plain 方法 完成 PKCE 加固。
- ACP 4.3 修复了一个 API 身份验证问题，在该问题中，某些 API 之前可以在未通过身份验证的情况下被访问。global 集群升级到 ACP 4.3 后，请将以下 L5 插件升级到与 ACP v4.3 兼

容的版本。否则，其 UI 页面可能无法打开：

- `Alauda DevOps v3`
- `Alauda AI Essentials`
- `Alauda Hyperflux`
- `Alauda Container Platform Data Services Essentials`

- 升级插件后，请确认列表中的每个插件 UI 页面都能成功打开。

Global DR 操作步骤

当环境同时包含主 global 集群和备用 global 集群时，请使用此操作步骤。下面与 DR 相关的步骤是在标准 CVO 工作流之外的补充。

1 在升级前验证 DR 环境

按照常规的 global DR 检查操作步骤，确保 备用 **global** 集群 中的数据与 主 **global** 集群 一致。有关 DR 拓扑和同步工作流的背景信息，请参见 Global Cluster Disaster Recovery。

如果检测到不一致，请在继续之前联系技术支持。

在 两个 global 集群上运行以下命令，确保没有任何 `Machine` 节点处于非运行状态：

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

如果存在此类节点，请先解决再继续。

2 从备用 global 集群卸载 etcd 同步插件

1. 通过 IP 或 VIP 访问 备用 **global** 集群 的 Web Console。
2. 切换到 管理员 视图。
3. 导航到 **Marketplace > Cluster Plugins**，并选择 `global` 集群。
4. 找到 灵雀云容器平台 **etcd Synchronizer** 并将其卸载。
5. 等待卸载完成后再继续。

3 在两个 **global** 集群上准备升级工件

在 备用 **global** 集群 和 主 **global** 集群 上完成标准工作流中的 准备升级工件。

在两个集群上使用相同的准备模式。

4 升级备用 **global** 集群

如果你将使用备用 global 集群上的 Web Console，请确认备用集群 `ProductBase` 的 `spec.alternativeURLs` 中包含备用 VIP：

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

准备完成后，在 备用 **global** 集群 上执行标准工作流中的剩余步骤：

1. 运行预检
2. 请求升级
3. 观察执行情况，直到备用 global 集群达到目标版本

5 升级主 **global** 集群

在备用 global 集群达到目标版本后，在 主 **global** 集群 上执行标准工作流中的剩余步骤：

1. 运行预检
2. 请求升级
3. 观察执行情况，直到主 global 集群达到目标版本

6 重新安装 **etcd** 同步插件并验证同步状态

在重新安装插件之前，请确认在使用该转发模式时，端口 `2379` 已从两个 global 集群的 VIP 正确转发到各自的控制平面节点。如果备用 global 集群可以直接访问活动 global 集群，则不需要通过负载均衡器进行端口转发。

要重新安装该插件：

1. 通过 VIP 访问 备用 **global** 集群 的 Web Console，并切换到 管理员 视图。
2. 导航到 **Marketplace > Cluster Plugins**，并选择 `global` 集群。
3. 找到 灵雀云容器平台 **etcd Synchronizer**，单击 **Install**，并配置所需参数。

配置插件时：

- 当端口 `2379` 没有通过负载均衡器转发时，请正确设置 **Active Global Cluster ETCD Endpoints**。
- 使用 **Data Check Interval** 的默认值。
- 除非你正在排查问题，否则保持 **Print detail logs** 为禁用状态。

验证同步 Pod 是否在备用 global 集群上运行：

```
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"
```

一旦出现 `Start Sync update`，请重新创建其中一个 Pod，以触发对具有 ownerReference 依赖关系资源的同步：

```
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"
```

检查同步状态：

```
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"
```

输出解释：

- `LOCAL ETCD missed keys`：这些 Key 存在于主 global 集群中，但在备用 global 集群中缺失。通常在重启一个 `etcd-sync` Pod 后可以解决。
- `LOCAL ETCD surplus keys`：这些 Key 存在于备用 global 集群中，但不在主集群中。在删除之前，请先与运维团队确认这些 Key。

相关文档

- 概述
- 升级前
- 升级业务集群
- 升级集群
- Global Cluster Disaster Recovery

升级业务集群

在 global 层已经达到目标 ACP Distribution Version 之后，业务集群即可升级。如果 global 层已经处于该 Distribution Version，则无需再次升级。

ACP 4.3 对业务集群采用与 `global` 集群相同的基于 CVO 的工作流：确认前提条件、运行 preflight 检查、发起升级请求，以及观察执行过程。此工作流还有两条额外规则：

- 如果平台使用 **global DR**，则在任何业务集群升级到该 Distribution Version 之前，standby 和 primary global 集群都必须先达到目标 Distribution Version。
- 目标版本通常只有在 global 层已经达到相同的 Distribution Version 之后，才会对业务集群开放。

目录

工作流

1 确认业务集群前提条件

在发起升级请求之前，请验证以下内容：

- 业务集群版本低于当前 `global` 集群版本。
- global 层已经达到目标 ACP Distribution Version。
- 在 global DR 环境中，standby 和 primary global 集群都已经达到该目标 Distribution Version。

2 运行 **preflight** 检查

在发起升级请求之前运行 preflight：

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

Preflight 会返回两部分内容：

输出	目的
`Summary`	显示总体结果、当前版本、期望版本和期望镜像。
`Checks`	显示每个单独验证项的结果。

如果 preflight 未通过，请不要提交升级请求，直到所有阻塞项都已解决。

有关 preflight 阻塞处理模式，请参见 按需处理 preflight 阻塞。

3 发起升级请求

在 preflight 通过后，可选择以下任一入口：

- 在 Web Console 中从业务集群发起升级。
- 当当前 ACP 会话可以访问目标业务集群时，使用 ACP CLI。为避免歧义，请使用 `--cluster` 显式指定目标集群。

如果你是首次使用 ACP CLI，请参见 ACP CLI 入门。有关会话和集群选择，请参见 管理 CLI 配置文件。

如果使用 Web Console，请求会遵循两步流程：

- 在 **Step 1** 中，查看 RPCH 列表。
- 单击 **Acknowledge** 继续到 **Step 2**。
- 在 **Step 2** 中，查看 **Current Version** 和 **Target Version**。此阶段页面不会显示插件列表或警告 panel。
- 目标版本是当前 `global` 集群版本，不能在 Web Console 中手动选择。
- 单击 **Start Upgrade**。
- 请求提交后，页面会显示升级请求已提交，且操作进入进行中状态。

ACP CLI 示例：

```
# 请求升级到当前在 availableUpdates 中发布的最高版本
ac adm upgrade --cluster=<cluster> --to-latest

# 请求升级到指定目标版本
ac adm upgrade --cluster=<cluster> --to=4.3.0
```

ACP CLI 会为指定的业务集群提交所请求的目标版本。升级是否可以继续，仍由集群的可用升级目标和升级控制器的 preflight 检查决定。ACP CLI 不用于升级 `global` 集群。

有关完整的 AC CLI 升级工作流（元数据准备、available updates、请求模式和状态解读），请参见 升级集群。有关完整的命令和标志语法，请参见 AC CLI Administrator 命令参考。

4 观察进度

在升级请求提交后，使用 `cvsh.status` 跟踪当前版本、期望版本、preflight 结果、阶段和历史记录：

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

如果当前 ACP 上下文指向目标业务集群，也可以检查 CLI 报告的状态：

```
# 显示目标集群升级的摘要、preflight 和阶段进度
ac adm upgrade status
```

有关 `ac adm upgrade status` 输出的详细语义（preflight 和阶段解读），请参见 升级集群。

排障时，请先检查 conditions、preflight 详情和历史记录：

```
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}'
```

在所有业务集群都达到 **ACP 4.3** 后

在所有业务集群都达到 ACP 4.3 后，请按照 禁用 PKCE Plain 方法 完成 PKCE 安全加固。

相关文档

- 概览
- 升级 global 集群
- 升级集群