升级

本文档将提供有关升级 ACP 的所有信息。

> **升级路径选择**
>
> 首先升级 `global` 集群，然后再升级工作负载集群。对于每个集群，升级发行版版本适用于两种节点操作系统。如果集群运行 Alauda OS，请继续执行在 **Alauda OS** 上升级 **Kubernetes**：
>
> - `global` 集群：在 Alauda OS 上升级 Kubernetes
> - 工作负载集群：在 Alauda OS 上升级 Kubernetes
>
> 有关升级模型，请参阅概述。

概览

概览

升级前准备

升级前准备

升级 **global** 集群

升级分发版本　　在 **Alauda OS** 上升级 **Kubernete**　　在 **DR** 环境中

升级工作负载集群

升级发行版　　在 **Alauda OS** 上升级 **Kubernetes**

升级验证

升级验证

升级故障排查

升级故障排查

概览

从 ACP 4.3 开始，集群升级采用基于 Cluster Version Operator (CVO) 的工作流。

可以通过 Web Console、ACP CLI 提交升级请求，也可以直接编辑 `ClusterVersionShadow`。Web Console 提供两种集群管理界面：一种是平台内置界面，另一种是由 **Alauda Container Platform Fleet Essentials** 集群插件提供的新界面。这两种界面均在 global 集群和业务集群页面中并列介绍。

将平台升级到新的 ACP Distribution Version 时，应先升级 `global` 集群，再升级业务集群：

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，其中包括准备制品和执行升级前检查。
2. global 层达到目标 Distribution Version 后，从受支持的业务集群升级入口升级业务集群，并监控集群状态，直至每个目标集群都达到相同的 Distribution Version。

业务集群只能升级到 global 层已达到的 Distribution Version。在启用了 **global** 灾难恢复 **(DR)** 的环境中，这意味着必须先将备用和主 global 集群都升级到目标 Distribution Version，然后才能将业务集群升级到该 Distribution Version。升级 global 层之前，请在 Kubernetes 支持矩阵 中确认每个业务集群的版本都在目标 Distribution Version 的兼容版本范围内。此项前提条件与升级顺序规则相互独立。

目录

按节点操作系统执行升级

Distribution Version 与 Kubernetes 版本

升级工作流

关键概念

升级范围和顺序
CVO 管理范围
升级入口
相关文档

按节点操作系统执行升级

根据节点操作系统的不同，每个集群的升级分为一个或两个步骤：

1. 升级 **Distribution Version** 适用于两种节点操作系统。CVO 会将 ACP Core 和已安装的 Aligned 插件升级到目标 Distribution Version。在运行非 Alauda OS 的集群上，此步骤还会在现有节点上原地升级 Kubernetes。
2. 在 **Alauda OS** 上升级 **Kubernetes** 仅适用于 Alauda OS 集群。Distribution Version 升级完成后，通过 Cluster API 滚动更新替换节点，从而部署 Kubernetes 和 Alauda OS 镜像。节点不会原地升级。

Distribution Version 与 Kubernetes 版本

组件	说明	管理方式
Distribution Version	集群上安装的 ACP Core 版本。	由 CVO 通过升级 **Distribution Version** 进行管理。
Kubernetes 和 OS 基线	在 Alauda OS 上，与已安装的 ACP 版本相对应、经过验证的 Kubernetes 次要版本、组件集和 Alauda OS 镜像。	由 Cluster API 和在 **Alauda OS** 上升级 **Kubernetes** 中的特定提供程序操作步骤进行管理。

在运行非 Alauda OS 的集群上，CVO 会在升级 Distribution Version 时一并升级 Kubernetes。在 Alauda OS 集群上，ACP 次要版本系列在其补丁生命周期内可能提供多个经过验证的 Kubernetes 目标版本。有关 Kubernetes 升级可用的特定补丁版本对应的 Kubernetes 和组件值，请参阅 OS 支持矩阵。

升级工作流

请按以下顺序参阅各页面：

1. 升级前准备：确认受支持的升级路径、集群就绪状态以及完整的目标版本制品集。
2. 升级 `global` 集群：升级 Distribution Version；如果启用了 global DR，则改为执行在 DR 环境中升级。如果 `global` 集群运行 Alauda OS，请继续执行在 Alauda OS 上升级 Kubernetes。在 DR 环境中，请在两个 `global` 集群上执行此操作。
3. 在 global 层达到目标 Distribution Version 后升级业务集群：升级 Distribution Version。对于运行 Alauda OS 的业务集群，请继续执行在 Alauda OS 上升级 Kubernetes。
4. 升级验证：确认 Distribution Version、Kubernetes、Core、Aligned 插件、Pod、制品、CVO 历史记录和 Web UI。
5. 升级故障排除：仅当升级前检查或执行报告阻塞或停滞时使用。

请规划维护时间线，确保准备工作在变更窗口开始前完成：

阶段	建议时间	状态变化
下载并发布制品	维护窗口之前	不会升级任何正在运行的组件。
执行升级前检查并解决阻塞	窗口开始前 1–2 周	执行只读检查；修复操作可能会单独改变环境。
升级 global 层	第一个维护窗口	Core、Kubernetes 和已安装的 Aligned 插件升级到目标版本。
升级业务集群	global 层升级后	每个选定的业务集群在各自受控的窗口内完成升级。
验证并完成后续工作	每个窗口关闭前	确认验收结果并完成所需的升级后操作。

关键概念

- **ClusterVersionShadow (`cvsh`)**：用于跟踪当前版本、目标版本、升级前检查结果、执行阶段和历史记录的升级资源。

- **Distribution Version**：集群当前已达到的 ACP 版本。业务集群只能升级到 global 层已达到的 Distribution Version。
- 升级前检查：在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升级前检查结果。
- 可用的升级目标：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**：用于升级 global 集群的准备脚本。使用平台内置 Registry 时，它会同步暂存于 `plugins/` 中的 Core 制品和软件包。使用外部 Registry 时，会跳过同步，并单独上传目标负载。该脚本还会在适当阶段运行升级前检查并部署 CVO。
- **Global DR** 环境：同时包含主 global 集群和备用 global 集群的环境。
- 主 **global** 集群：平台访问域当前解析到的 global 集群。
- 备用 **global** 集群：DR 对中的另一个 global 集群。故障转移后，两个集群的角色会互换。

升级范围和顺序

完整的 ACP 升级分阶段进行。每个集群在一个维护窗口内完成升级，但平台会逐个集群推进，先升级 global 集群，再升级业务集群。

单个集群的升级窗口包含四类工作：

工作项	是否必需	升级方式	如果运行 **Alauda OS**
ACP Core（平台本身）	必需	由通过同步升级制品准备的制品驱动 CVO	控制流程相同；Kubernetes 步骤单独处理，见下文
Aligned 插件（与匹配的 ACP 版本一同发布的集群插件和 operator）	每个已安装的插件都必需	对于通过 **ACP Upgrade to v4.5** 提供的 Aligned 应用，请将其软件包手动复制到 `plugins/`，并通过同步升级制品发布。使用 `violet` 发布其他已安装的 Aligned 软件包。随后，CVO 会在集群升级过程中升级每	相同

工作项	是否必需	升级方式	如果运行 **Alauda OS**
		个已安装的 Aligned 插件。	
Kubernetes	必须在同一窗口内完成	在非 Alauda OS 集群上，CVO 会在同一集群升级过程中，基于现有节点就地升级 Kubernetes；不会替换节点。	Distribution Version 升级后，Kubernetes 会通过 Cluster API 滚动更新，使用新的基于 Alauda OS 的镜像替换节点，而不是就地升级。对于 `global` 集群，请参阅在 Alauda OS 上升级 Kubernetes；对于业务集群，请参阅 Huawei DCS、VMware vSphere、Huawei Cloud Stack 或裸金属
Agnostic 插件（独立发布的集群插件和 operator）	可选，按插件分别处理	集群达到目标 Distribution Version 后，从 **Marketplace > Cluster Plugins** 或 operator 自身的工作流升级每个集群插件或 operator	相同

以下规则决定各部分的推进时机：

- 平台要求集群的 Kubernetes 版本与其 Core 和 Aligned 版本一起升级；新 Core 版本与旧 Kubernetes 次要版本混用未经验证。
- 请求升级前，请确保该集群上已安装的每个 Aligned 插件都有目标版本软件包可用。对于来自 **ACP Upgrade to v4.5** 的软件包，请使用同步升级制品中的 `plugins/` 发布路径；对于其他 Aligned 软件包，请使用 `violet`。已安装的 Aligned 插件缺少软件包会使 CVO 升级停滞。对于集群上未安装的 Aligned 插件，其软件包不会导致这些插件被安装。
- 集群插件是全局资源。通过 global 层制品发布或 `violet` 工作流提供的软件包可供平台中的每个集群使用；无需为每个业务集群再次发布同一个集群插件。
- operator 的作用域为各个集群。建议将 operator 推送到 global 层，但仅此操作并不足够；在升级业务集群之前，同一个 operator 必须在每个业务集群上可用。如果在 global 窗口期间忘记推送某个 operator，可以在业务集群升级前再次推送，作为备用措施。

- 仅当业务集群超出目标 ACP 版本的 Kubernetes 支持矩阵时，才必须升级业务集群。仍处于受支持范围内的业务集群，在 global 层推进后可以保留现有 Kubernetes 版本；平台会继续管理这些集群。

CVO 管理范围

Cluster Version Operator（CVO）决定由其端到端驱动的升级类型：

生命周期类型	是否由 **CVO** 驱动？	说明
Core	是，CVO 是唯一受支持的路径	Core 升级需要通过 global 集群同步工作流准备制品；CVO 使用这些制品。
Aligned 插件	默认是	集群插件或 operator 工作流也可以脱离常规的 CVO/Distribution Version 升级路径，单独升级某个 Aligned 插件，例如，在不将集群移至新的 Distribution Version 的情况下应用关键安全修复。
Agnostic 插件	否	CVO 不管理这些插件。集群达到目标 Distribution Version 后，从 **Marketplace > Cluster Plugins** 升级每个集群插件，并通过各自的工作流升级每个 operator。

实际运行生产维护窗口的客户通常会一并推进 Core、Aligned 以及正在使用的 Agnostic 插件。下面的升级页面记录了该路径：在升级前准备期间准备所有所需制品，通过 CVO 驱动 Core 和 Aligned，最后从 Marketplace 完成正在使用的 Agnostic 插件升级。

INFO

非 **Alauda OS** 和 **Alauda OS** 集群上的 **Kube-OVN**

在非 **Alauda OS** 集群上，Kube-OVN 是 ACP Core 的一部分，由 CVO 与 Core 的其余部分一起自动升级——operator 无需修改任何按集群设置的 Kube-OVN 版本注释，也无需手动操作 Kube-OVN `AppRelease`。

在 **Alauda OS** 集群（Huawei DCS、VMware vSphere、Huawei Cloud Stack 和裸金属）上，Kube-OVN 由基础设施提供商通过集群的 `Cluster` 资源上的 `cpaas.io/kube-ovn-version` 注释单独驱动——请参阅在控制平面之前升级 Kube-OVN。该路径不适用于非 Alauda OS 集群。

升级入口

- **global** 集群：按照经过验证的基于 `upgrade.sh` 的操作步骤执行。准备阶段完成后，可从 Web Console 请求升级、使用 ACP CLI，或直接更新 `ClusterVersionShadow.spec.desiredUpdate`。
- 业务集群：目标 Distribution Version 对业务集群可用后，使用 Web Console 或 ACP CLI。

相关文档

- 升级前准备
- 升级 `global` 集群的 Distribution Version
- 在 Alauda OS 上升级 `global` 集群的 Kubernetes
- 在 DR 环境中升级 global 集群
- 升级业务集群的 Distribution Version
- 在 Alauda OS 上升级业务集群的 Kubernetes
- 更改 Alauda OS 节点的计算机规格
- 升级验证
- 升级故障排除
- global 集群灾难恢复

升级前准备

Alauda Data Services Relational Database E1 v4.3.0 中移除 MySQL-PXC

如果存在 **MySQL-PXC** 实例，则在升级到 ACP v4.3.0 或更高版本后（同时升级 MySQL operator 时），这些实例将变为不受管理状态。有关迁移说明，请参阅 MySQL Operator 升级指南↗。

支持的升级路径：

在次要版本策略层面，ACP 4.2、4.3 和 4.4 均处于 ACP 4.5 发布系列的 n-3 直接升级窗口内。此策略并不意味着每个源修订版本都可以升级到每个目标修订版本。

仅升级到平台针对当前源修订版本提供的 4.5.x 目标版本。注册目标 `ProductManifest` 后，请确认目标版本显示在 `availableUpdates` 中（可在 Web Console 或 `ac adm upgrade` 中查看），并确认 `VersionUpgradePath` 预检查通过。如果未提供任何 4.5.x 目标版本，请使用包含当前源修订版本的更高 4.5.x 目标版本，或先按照平台提供的中间目标版本进行升级。请勿使用显式升级请求或禁用 `VersionUpgradePath` 来绕过不受支持的源版本到目标版本组合。

ACP 4.1.x 及更早版本不在 ACP 4.5 的直接升级窗口内。请按照平台提供的中间升级链进行升级，并在每个跳转点验证 `availableUpdates` 和 `VersionUpgradePath`。

INFO

对于使用 Alauda OS 的集群，ACP Distribution Version 必须遵循同一条由 CVO 提供的源版本到目标版本路径。当目标 ACP 版本比集群当前版本高出一个以上的 Kubernetes 次要版本时，在 Alauda OS 上升级 Kubernetes 需要预先准备中间版本的构件。开始升级窗口前，请参阅准备跨版本升级。

目录

重要说明

升级前验证模块稳定性

Kubernetes 1.35 或更高版本的节点就绪性

下载离线环境的软件包

步骤 1：下载 Core 软件包

步骤 2：下载 ACP Upgrade to v4.5

步骤 3：盘点范围内每个集群上的其他 Extensions

步骤 4：下载其他 Extension 软件包

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。
- 在升级 global 层之前，确认所有业务集群均处于 Kubernetes 支持矩阵 中记录的目标 Distribution Version 的兼容版本范围内。
- 如果任何业务集群超出目标兼容范围，请先升级该业务集群，直到其进入该范围，然后再升级 global 层。
- 业务集群只能升级到 global 层已达到的 Distribution Version。

升级前验证模块稳定性

集群升级预检要求目标集群上已安装的每个 **Core** 和 **Aligned** 插件模块均处于稳定状态。如果任一此类模块正在安装、升级或处于失败状态，`ModuleInfoStable` 和 `ClusterModuleStable` 预检会阻止升级，直到问题得到解决。常见原因是独立插件升级在请求集群升级时仍在运行。

在请求升级之前，列出目标集群上已安装的模块，并确认每个模块均处于 `Running` 阶段：

```
kubectl get moduleinfo -l cpaas.io/cluster-name=<cluster> \
  -o custom-columns='MODULE:.metadata.labels.cpaas\.io/module-name,CURREN
T:.status.version,TARGET:.spec.version,PHASE:.status.phase'
```

同时确认集群模块本身不处于部署或升级失败状态：

```
kubectl get clustermodule <cluster> -o jsonpath='{.status.base.deployStat
us}{"\n"}'
```

每个模块的 `PHASE` 必须为 `Running`，并且 `deployStatus` 不得为 `Upgrading`、`DeployFailed` 或 `UpgradeFailed`。如果模块仍在升级，请等待其达到 `Running`；如果模块处于 `Failed` 或 `Blocked` 状态，请先解决问题，再请求集群升级。

Kubernetes 1.35 或更高版本的节点就绪性

如果目标版本为 ACP 4.4 或更高版本、会将集群升级到 Kubernetes 1.35 或更高版本，并且报告了要求管理员确认节点就绪性的确认门槛，请在确认该门槛之前，验证将运行目标集群的每个生产节点。

通过 SSH 或环境中可用的其他管理员节点访问方式登录每个节点，并直接在节点操作系统上运行以下检查。所使用的账户和凭据取决于集群节点的置备方式。

验证 Linux 内核版本：

```
uname -r
```

内核必须为 `5.8` 或更高版本，并且还必须符合目标版本的 ACP 支持的操作系统和内核矩阵。

验证节点使用 cgroup v2：

```
stat -fc %T /sys/fs/cgroup/
```

预期输出为：

```
cgroup2fs
```

对于 Alauda OS 集群，Kubernetes 通过替换节点进行升级。在升级到 Kubernetes 1.35 或更高版本时，还需将所需的 kubelet 补丁应用到控制平面和工作节点引导配置。

下载离线环境的软件包

生产维护窗口通常会在同一窗口中升级 **ACP Core**、**Aligned** 插件（与同一 ACP 版本一起发布的集群插件和 operator）以及正在使用的 **Agnostic** 插件（独立发布的集群插件和 operator）。Core 和 Extension 软件包需要分别下载。在维护窗口前准备好这三组软件包，以便尽早完成制品同步。

步骤 1：下载 Core 软件包

从 灵雀云 **Customer Portal** 下载与目标架构匹配的 **ACP 4.5 Core Package**。Core Package 仅包含 Core 制品；不包含任何 Aligned 或 Agnostic Extension 软件包。

步骤 2：下载 ACP Upgrade to v4.5

使用 **Violet v4.2.13** 或更高版本直接下载场景软件包。运行 `violet version` 验证版本，并使用 `violet ac login` 登录 灵雀云 Customer Portal。有关安装和登录说明，请参阅下载软件包。

将 `<target-platform-version>` 设置为 Core Package 的确切 ACP Distribution Version，例如 `v4.5.0`。将 `<arch>` 设置为 `amd64`、`arm64` 或 `hybrid`，以匹配 Core Package。先列出场景，然后下载 **ACP Upgrade to v4.5**：

```
violet ac scenarios --arch=<arch> --platformVersion=<target-platform-vers
ion>
violet ac download-app --arch=<arch> --platformVersion=<target-platform-v
ersion> --scenario="ACP Upgrade to v4.5"
```

确认 **ACP Upgrade to v4.5** 出现在场景列表中，然后再下载。

也可以不使用 Violet，而是在 Customer Portal 中转到 **Marketplace > Batch Download > Install**。在页面中，从 **Your ACP Version** 选择目标 Core Package 版本，选择匹配的 **Architecture**，从 **Scenarios** 中选择 **ACP Upgrade to v4.5**，然后下载软件包。

此软件包集提供以下 Aligned 应用的目标版本软件包。这些软件包与 Core Package 分开。

下载的文件名称为 `<package-name>.<arch>.<version>.tgz`，其中不包含显示名称。下面的每一项都提供了软件包名称，便于确认已收到全部 11 个软件包：

Application	Package name
Alauda Container Platform Web Console	`console`
Alauda Container Platform Web Console Central	`console-central`
Alauda Container Platform Marketplace Web Console	`console-marketplace`
Alauda Container Platform Application Catalog	`application-catalog`
Alauda Container Platform Observability Essentials	`aiops-base`
Alauda Container Platform Essentials	`container-platform`
Alauda Container Platform Cluster Essentials	`acp-business`
Alauda Container Platform GatewayAPI Plugin	`gatewayapi-plugin`
Alauda Container Platform Basic Monitoring	`prometheus`
Alauda Language Pack for Chinese	`i18n-zh`
Alauda Container Platform Networking E1	`calico`

发布升级制品前，手动将全部 11 个软件包复制到解压后的 Core Package 的 `plugins/` 目录中。同步升级制品介绍了内置 Registry 和外部 Registry 的发布分支。

步骤 3：盘点范围内每个集群上的其他 Extensions

集群插件和 operator 由不同的机制管理。本步骤需要同时处理两者：

- 集群插件是全局资源。推送到全局层级的集群插件会在平台中的每个集群上变为可安装和可升级；每个集群插件只需推送一次。
- **Operator**的作用域限定为每个集群。建议先将 operator 推送到全局层级，但在升级某个业务集群之前，必须确保该 operator 软件包已在该集群上可用。

使用步骤 2 中相同的 `violet` 安装来盘点已安装的 Extensions，并发布不属于 **ACP Upgrade to v4.5** 的软件包。有关更多信息，请参阅下载软件包和上架软件包。

在任何可访问 灵雀云 平台端点的计算机上，运行 `violet list`，列出当前环境中已安装的插件（包括 Aligned 和 Agnostic），并将结果导出到 `./apps.yaml`：

```
violet list \
  --platform-address "https://<your-platform-domain>" \
  --platform-token "<platform_token>" \
  --output-file "./apps.yaml"
```

优先使用 `--platform-token`，而不是 `--platform-password`，以避免在 shell 历史记录和进程列表中暴露密码（`ps aux`）。

此基于令牌的盘点流程要求使用 Violet v4.2.13 或更高版本。较早版本可能会失败并显示 `failed to get user info from platform server`；请先升级 Violet，然后再继续。

步骤 4：下载其他 Extension 软件包

将导出的 `apps.yaml` 文件导入 灵雀云 **Customer Portal**，使软件包列表与集群当前运行的内容保持一致。下载集群所使用的其他 Aligned 和 Agnostic Extensions 的匹配目标版本软件包。

如果结果中包含 **ACP Upgrade to v4.5** 中的 11 个应用之一，请勿使用 `violet` 发布该应用的重复软件包。应通过 Core Package 的 `plugins/` 目录使用 **ACP Upgrade to v4.5** 中的软件包。仅对其余 Extension 软件包使用 `violet push`。

下载升级所需的每个软件包，确保在升级窗口开始前所有软件包都已在本地可用。下一页升级 global 集群介绍了所选 Registry 路径如何发布 Core 和手动暂存的 11 个软件包，以及 `violet` 如何发布其他 Extensions。制品发布不会升级任何已在运行的插件，因此请在维护窗口前完成发布。

> 必需的 **Aligned** 插件软件包

正在升级的集群上安装的每个 Aligned 插件都必须在升级请求前准备好其目标版本软件包。对于未安装的 Aligned 应用，CVO 不要求提供目标软件包；同步软件包也不会安装当前未安装的应用。

如果缺少必需的软件包，CVO 会阻止进度，并持续重试现有请求。对于已安装的 Aligned 集群插件，`ClusterVersionShadow` 条件会报告缺少目标 `ModulePluginConfig` 或其未处于 Ready 状态。对于已安装的 Aligned operator，请检查其目标 catalog、`Subscription`、`InstallPlan` 和 `ModuleInfo`。通过同步升级制品中的 `plugins/` 发布路径添加 **ACP Upgrade to v4.5** 中的软件包；使用 `violet` 发布其他软件包。

WARNING

从 v4.2 开始，我们引入了名为 **Alauda Container Platform Log Essentials** 的新插件。如果之前安装了日志存储插件，还必须先上传该插件，然后再开始升级。

升级 global 集群

升级分发版本 在 Alauda OS 上升级 Kubernete 在 DR 环境中

升级 `global` 集群的分发版本

> **升级路径**
>
> 此页面适用于两种节点操作系统。在非 Alauda OS `global` 集群上，此操作步骤还会原位升级 Kubernetes，并在操作完成后完成 `global` 集群升级。如果您的 `global` 集群使用 Alauda OS，请先完成此页面中的操作，然后继续执行 在 Alauda OS 上升级 Kubernetes ，通过替换节点来推出 Kubernetes。

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 分发版本，请先将 global 层升级到目标分发版本，然后将业务集群升级到相同的分发版本。

集群升级使用基于 CVO 的工作流。典型的 `global` 集群升级包括工件准备、升级前检查、升级请求和状态观察。

升级 `global` 集群前，请确认每个业务集群都处于 Kubernetes 支持矩阵 中目标版本的兼容版本范围内。此先决条件与第三方集群接入范围相互独立。

无论环境是否使用 **global DR**，都必须满足此兼容版本先决条件。global DR 会改变升级 global 层所使用的操作步骤，但不会改变以下要求：在将 global 层升级到目标分发版本之前，业务集群必须保持在兼容的 Kubernetes 版本范围内。

Global 集群升级遵循本页中记录的、经过验证的基于 `upgrade.sh` 的操作步骤。您可以从 Web Console 请求 global 集群升级，也可以通过更新 `ClusterVersionShadow.spec.desiredUpdate` 或使用带有 `--cluster=global` 的 ACP CLI 请求升级。有关完整的 AC CLI 工作流和输出解读，请参阅升级集群。有关完整的命令和标志语法，请参阅 AC CLI 管理员命令参考。

如果环境使用 **global DR**，请按照在 DR 环境中升级 Global 集群操作。否则，请按照下面的标准工作流操作。

目录

- 标准工作流
  - 同步升级工件
  - 运行升级前检查
  - 处理升级前检查阻塞项
  - 部署集群版本 operator
  - 请求升级
  - 观察执行情况
  - 从 Marketplace 升级 Agnostic 插件
- 验证升级
- 相关文档

标准工作流

global 集群升级会按照时间线分阶段执行。大部分工作会在维护窗口之前完成，从而使维护窗口本身保持简短且可预测：

阶段	时间	发生的操作	对集群的影响
1. 同步工件	窗口开始前的任意时间	对于平台内置 Registry，`upgrade.sh --only-sync-image` 会上传 Core 工件以及您手动放置在 `plugins/` 中的十一个软件包。对于外部 Registry，请在运行 `upgrade.sh` 前手动上传目标 Core payload。在这两种情况下，`violet` 都会发布窗口所需的其他 Aligned 和 Agnostic 软件包。	无 — 会发布镜像和目录条目，但不会更改正在运行的插件；不会停机。

阶段	时间	发生的操作	对集群的影响
2. 升级前检查	窗口开始前 1–2 周	`upgrade.sh --preflight` 会验证升级准备情况。请在窗口开始前解决所有阻塞项。	无 — 只读验证。
3. 升级	维护窗口期间	`upgrade.sh --skip-sync-image` 会部署或更新集群版本 operator（CVO）；然后请求并观察升级。	集群将完成升级。

阶段 1 和阶段 2 不会更改集群状态 — 请尽早运行这些阶段，以便维护窗口中只包含阶段 3。对于小型或实验室环境，也可以运行单个 `bash upgrade.sh`，同时执行同步和 CVO 部署，但生产环境的维护窗口通常会将这两个阶段分开。

1 同步升级工件

时间：维护窗口前的任意时间。此步骤会将工件上传到 Registry，不会更改集群状态。

首先，记录已配置的 Registry 地址，并确定平台使用的是内置 Registry 还是外部 Registry：

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

对于平台内置 Registry，第二个值为 `false`；对于外部 Registry，第二个值为 `true`。将目标 Aligned 软件包复制到 `plugins/` 后，再选择发布命令。

Core Package 不包含 **ACP Upgrade to v4.5** 中的 Aligned 软件包。将升级前准备中列出的十一个单独下载的软件包复制到解压后的 Core Package 的 `plugins/` 目录中。对每个软件包分别运行以下命令：

```
cp <path-to-upgrade-extension-package> ./plugins/
```

同步前，请确认该目录包含全部十一个软件包：

```
ls -1 ./plugins/
```

根据 `ProductBase.spec.registry.external` 的值，按照相应分支发布暂存的 payload。

对于平台内置 Registry（`false`），请在解压后的 Core Package 目录中以仅同步模式运行 `upgrade.sh`：

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

`--only-sync-image` 会上传 Core 镜像以及您手动放入 `plugins/` 的软件包，但不会部署集群版本 operator，也不会更改正在运行的应用。CVO 将在维护窗口内的后续步骤中部署 — 请参阅部署集群版本 operator。

对于外部 Registry（`true`），不要将 `--only-sync-image` 作为已发布的证明。`upgrade.sh` 会自动跳过镜像同步。请从目标 Core Package 的 `installer/` 目录开始，按照准备目标版本 payload，同时运行 `res/upload.sh all` 和 `res/upload.sh necessary`。两种模式可以按任意顺序运行，但都必须成功。

所选发布路径会使基于 CVO 的工作流所需的工件可用，包括：

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

对于 **ACP Upgrade to v4.5** 中的十一个 Aligned 原生应用，始终将软件包暂存到 `plugins/`。内置 Registry 路径会通过 `upgrade.sh --only-sync-image` 发布这些软件包；外部 Registry 路径则通过 `res/upload.sh all` 发布。CVO 仅会升级已安装的原生应用；为未安装的原生应用发布软件包并不会安装该应用。

对于从 `violet list` 清单下载的其他软件包，请使用 `violet push`：

软件包组	发布路径	要求
其他 Aligned 集群插件	`violet push` 到 global 层级	对于每个不属于 **ACP Upgrade to v4.5** 的已安装插件都是必需的。
其他 Aligned operator	`violet push` 到 global 层级以及安装了该 operator 的每个业务集群	在升级运行该 operator 的每个集群之前都必须完成。
Agnostic 集群插件和 operator	`violet push` 到计划升级它们的集群	对 CVO 可选；但在开始单独的 Marketplace 或 operator 升级前必须完成。

```
# Publish another cluster plugin to the global tier
violet push <path-to-other-cluster-plugin-package> \
  --platform-address "https://<your-platform-domain>" \
  --platform-token "<platform_token>"

# Publish another operator to global and each workload cluster that r
uns it
violet push <path-to-other-operator-package> \
  --platform-address "https://<your-platform-domain>" \
  --platform-token "<platform_token>" \
  --clusters "global,<workload-cluster-1>,<workload-cluster-2>"
```

建议在 global 窗口期间，将每个 operator 软件包推送到每个 ACP 集群。如果 global 窗口已结束，而您发现某个业务集群缺少 operator 软件包，仍可在升级该业务集群之前将软件包推送到该集群 — 请参阅升级业务集群。

WARNING

请求升级前，请确保目标软件包对集群上已安装的每个 Aligned 插件都可用。对于 **ACP Upgrade to v4.5** 中的原生应用，请使用 `upgrade.sh`；对于其他 Aligned 软件包，请使用 `violet`。同步软件包不会安装当前未安装的应用。对于已安装的 Aligned 集群插件，CVO 要求对应的目标 `ModulePluginConfig` 处于 Ready 状态；否则，CVO 无法构建升级计划，会通过 `ClusterVersionShadow` 报告错误，并重试现有请求。已安装的 Aligned operator 需要其目录中对应的目标 `InstallPlan`。

Registry 的行为取决于环境配置方式：

场景	行为
已指定 `--registry`	直接使用提供的 Registry。
未指定 `--registry`	从 `ProductBase.spec.registry.address` 读取 Registry 地址。
平台内置 Registry	使用 global VIP 重新构建访问地址。
外部 Registry	自动设置 `SKIP_SYNC_IMAGE=true`；目标 payload 必须已通过规范的目标版本 payload 操作步骤上传。
需要上传镜像，但未提供凭据	从 `cpaas-system/registry-admin` Secret 中读取 `username` 和 `password`。

当目标 Registry 不是平台默认 Registry 时，请添加 Registry 参数：

参数	用途
`--registry`	指定目标 Registry 地址。
`--username` / `--password`	指定 Registry 凭据。

工件验证会将 `plugins/` 中存在的每个软件包与目标 Core Package 预期的版本进行比较，若版本不匹配则失败。它不会验证软件包集合是否完整：从未暂存的软件包不会被检查，运行会继续。请先根据下载 ACP Upgrade to v4.5 确认十一个软件包，再依赖验证成功的结果。

> **WARNING**
>
> 镜像和插件同步完成之前，请勿开启维护窗口。

运行升级前检查之前，请确认 Registry 配置仍然正确：

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

将所选发布命令成功完成，以及 Registry 管理界面或 API 中的相关信息，作为阶段 1 的证据。请确认预期的目标仓库和标签均已存在，并将已发布的 Extension 集合与导出到 `apps.yaml` 的已安装清单进行比较。不要将 `ProductBase.status.artifacts` 作为目标版本的零输出门槛：它表示当前目录，其中可以合理包含可选或未安装软件包的 `Absent` 条目。

2 运行升级前检查

时间：维护窗口前 1–2 周，以便在窗口开启前有时间解决所有阻塞项。

以升级前检查模式运行 `upgrade.sh`：

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

升级前检查为只读操作 — 它会验证升级准备情况，不会更改集群状态。

> **升级前检查要求 CVO 自定义资源已存在**
>
> 对于从未部署过集群版本 operator 的集群，`--preflight` 尚无法运行，并会报错：
>
> ```
> resolve desired release: get ProductManifest for version "<targ
> et-version>": no matches for kind "ProductManifest" in version
> "config.alauda.io/v1"
> ```
>
> 该消息提到了一个 API 组，但原因是操作顺序不正确，而不是版本不匹配：`ProductManifest` 自定义资源定义和目标版本 `ProductManifest` 均由部署集群版本 operator 创建。请先执行该步骤，然后再运行升级前检查。对于之前已使用 CVO 工作流完成升级的集群，这些资源已存在，同步工件后即可随时运行升级前检查。

升级前检查会返回两部分内容：

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`	显示各项验证的结果。

默认检查集包括：

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

3 处理升级前检查阻塞项

时间：维护窗口前；升级前检查一旦报告阻塞项，就应立即处理。

在经过定制的环境中，首次运行升级前检查就全部通过并不常见。请将解决阻塞项视为准备阶段的一部分，而非例外情况 — 这正是要提前一到两周运行升级前检查的原因。

如果任何检查未通过，请勿开启维护窗口。请根据升级故障排除解决每个阻塞项，其中涵盖：

- `ResourcePatchUpgradeable` — 在经过定制的环境中最常见的阻塞项，以及所需的目标版本豁免。

- 其他常见阻塞项 — `KubernetesVersionSupported`、`VersionUpgradePath`、`ClusterRunning`、`DockerRuntimeUnsupported` 和模块稳定性。
- 管理员确认门槛 — `AdminAckRequired` 失败时的处理方式。

不要强行绕过失败的检查进行升级。`VersionUpgradePath`、`KubernetesVersionSupported` 和 `AdminAckRequired` 是受支持升级路径本身的门槛。如果无法解决阻塞项，请在继续之前联系技术支持。

4 部署集群版本 operator

时间：维护窗口开始时。

以跳过同步模式运行 `upgrade.sh`。由于已在同步升级工件中上传工件，因此跳过同步：

```
bash upgrade.sh --skip-sync-image
```

此操作会部署或更新集群版本 operator（CVO），并完成剩余准备工作。下一步请求升级后，CVO 将驱动 Core 和 Aligned 插件升级。

命令完成后，请检查目标 `ProductManifest`，再请求升级：

```
kubectl get productmanifest v<target-version> \
  -o jsonpath='{range .status.conditions[*]}{.type}{"\t"}{.status}
{"\t"}{.reason}{"\t"}{.message}{"\n"}{end}'

kubectl get productmanifest v<target-version> -o json | jq -r '
  .status.artifacts[] as $artifact
  | $artifact.channels[]
  | select(.artifactStatus != "Ready")
  | [$artifact.name, .channel, .tag, .artifactStatus]
  | @tsv'
```

每个工件的输出用于诊断，不要求为空。目标清单可能包含可选、Agnostic 或未安装的条目，这些条目未处于 Ready 状态也可能是正常情况。请将输出与已安装的 Core 和 Aligned 清单进行比较。对于已安装的 Aligned `ModulePlugin`，如果对应的目标 `ModulePluginConfig` 所属的 channel 未处于 Ready 状态，请检查该资源：

```
kubectl get modulepluginconfig <modulepluginconfig-name> -o yaml
```

请使用目标 `ProductManifest` 中的组件名称和 channel 标签，或 `ClusterVersionShadow` 错误中报告的名称，来识别 `ModulePluginConfig`。此名称中的组件版本不一定是 ACP Distribution Version。不要仅凭 `artifactStatus: Absent` 就认定 Registry manifest 缺失；请使用 `ModulePluginConfig` 条件及其 `.spec.image`，以区分 manifest 缺失与身份验证、CA、网络或软件包内容问题。

5 请求升级

部署集群版本 operator 后，可通过以下任一入口请求升级。这些入口的功能相同；请选择符合您运维模式的入口。

平台在 Web Console 中提供两个集群管理界面。**Web Console** 是平台内置的界面，也是默认入口。**Web Console (Fleet Essentials)** 是较新的集群管理界面，通过独立的集群插件提供，必须先安装才能使用。

Web Console

这是内置的集群管理界面。每个 ACP 平台都提供此界面，除平台本身外无需安装任何组件。

请求升级需要对目标集群的 `ClusterVersionShadow` 和 `ResourcePatch` 资源具有更新权限。

1. 在左侧导航栏中，点击 **Clusters** > **Clusters**，然后点击 `global` 集群。
2. 打开 **Functional Components** 选项卡。
   该选项卡会列出每个组件及其 **Status** 和 **Version**。有可用新版本的组件会标记为 **Upgradeable**。无法升级到新版本的组件则会显示阻止原因，页面还会显示与新版本不兼容的组件。
3. 点击 **Upgrade**。
   没有可升级版本（**No upgradable version**）、升级已在运行（**Upgrading**）或集群运行状况不正常（**Cluster Abnormality**）时，该按钮处于禁用状态。将鼠标悬停在按钮上即可查看原因。
4. 在 **Personalized Configuration Confirmation** 步骤中，查看列出的配置，然后点击 **Acknowledge**。

仅当集群包含个性化配置时才会显示此步骤，包括自定义的 `ModuleInfo`、自定义的 `ClusterModule` 或 `ResourcePatch`。该步骤会列出每个配置项，并以只读 YAML 的形式显示所选配置项的覆盖值，警告内容为：*您的环境中存在以下个性化配置，请与技术支持人员确认是否需要删除或调整，否则升级可能会失败！*
当集群不包含个性化配置时，不会显示此步骤，页面会直接打开下一步所述的确认表单。

5. 在 **Confirm Component Upgrade** 步骤中，查看 **Current Version** 和 **Target Version**。
   目标版本是 `ClusterVersionShadow.status.availableUpdates` 的第一项。该版本由已为集群准备的升级工件确定，无法在此界面中手动选择。
6. 点击 **Upgrade**。
7. 在 **Are you sure you want to upgrade the cluster version?** 对话框中，设置 **Maintenance mode**，然后点击 **Upgrade** 提交请求。

> 默认情况下，此对话框会启用维护模式
>
> 确认对话框打开时，**Maintenance mode** 已启用。维护模式启用期间，非平台管理员用户将被限制访问平台功能，并会看到 **Platform maintenance announcements** 中的文本。启用开关时必须填写公告，且公告长度限制为 200 个字符。如果变更窗口不要求限制访问，请在提交前关闭该开关。该开关会写入平台范围的维护模式设置，因此变更窗口结束后请再次检查该设置。

提交请求后，页面会返回 **Functional Components** 选项卡，并显示 *The upgrade has been initiated. It may take 5~10 minute for the status to update.*。点击组件 **Status** 列中的放大镜图标，可以跟踪该组件的升级进度。

Web Console (Fleet Essentials)

此入口属于新的集群管理界面，由 **Alauda Container Platform Fleet Essentials** 集群插件提供。默认未安装该插件。

前提条件：从 **Marketplace** > **Cluster Plugins** 安装 **Alauda Container Platform Fleet Essentials v1.0.4** 或更高版本。该插件会在左侧导航栏中添加 **Clusters (New)** 入口。如果没有该入口，说明此平台未安装该插件 — 请改用 **Web Console** 选项卡。

1. 在左侧导航栏中，点击 **Clusters (New)**，然后点击 `global` 集群。
2. 保持在 **Overview** 选项卡中，找到 **Version & Update** 卡片。
   该卡片显示 **Distribution Version** 和 **Kubernetes Version**，以及状态横幅：**Up to date**、**Update Available**、**Upgrading**、**Update Blocked** 或 **Upgrade Not Enabled**。**Upgrade Not Enabled** 表示集群版本 operator 未部署在集群上 — 请先完成部署集群版本 operator。
   存在目标版本时，该卡片还会报告目标工件是否就绪：**Target artifacts ready** 或 **Target artifacts not ready**，并显示 Core 数量和 Aligned 数量。点击 **Show per-component**，可列出尚未处于 Ready 状态的 Core 和 Aligned 组件。Core 工件由 `upgrade.sh --only-sync-image` 同步；Aligned 插件不包含在 Core Package 中，需要使用 `violet push` 发布各自的软件包。就绪状态不会阻止请求，但只有目标版本的工件就绪后，该目标版本才会显示。
3. 点击 **Upgrade** 打开 **Update Cluster** 对话框。
   仅当有可用更新时，该按钮才处于启用状态。运行期间，卡片会显示处于禁用状态的 **Upgrading**；当预检检查阻止升级时，卡片会显示处于禁用状态的 **Update Blocked**，并在卡片的 **Preflight Checks** 部分列出阻止原因。重新请求升级不会清除阻止状态 — 请解决阻止原因，升级会继续进行。
4. 在 **Custom Configuration** 步骤中，查看将被豁免的资源补丁，然后点击 **Next**。
   该表格会列出每个待处理的 `ResourcePatch`，以及其 **Name**、**Target Resource**、**Current Exempt Version** 和 **Exempt To (after upgrade)**。开始升级会为每个资源添加 exempt-for-version 注解，使升级不会覆盖这些资源。如果没有需要确认的内容，该步骤会显示 **No custom configuration to confirm**。
5. 在 **Confirm & Upgrade** 步骤中，查看 **Current Version** 和 **Target Version**。
   在 `global` 集群中，当 `ClusterVersionShadow.status.availableUpdates` 包含多个目标时，**Target Version** 是下拉列表；请选择已准备其工件的版本。只有一个可用目标时，该版本以只读形式显示。此步骤还会再次说明升级范围 — Core、Aligned、Kubernetes 和 Agnostic — 以及集群升级无法回滚这一注意事项。
6. 点击 **Confirm & Start**。
   此界面没有第二个确认对话框，也没有维护模式开关。卡片会刷新为 **Upgrading** 状态，卡片中的 **Preflight Checks**、**Upgrade Progress** 和 **Update History** 部分会跟踪本次运行。

ACP CLI

在编写升级脚本时，或当前 ACP 会话已指向 global 集群时，请使用此入口。

```
# 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=<target-version>

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

有关完整的 AC CLI 操作步骤和输出解读，请参阅升级集群。有关完整的命令和标志语法，请参阅AC CLI 管理员命令参考。

kubectl

当需要直接操作底层 CVO 资源时，请使用此入口。使用目标版本修改 `ClusterVersionShadow.spec.desiredUpdate`。

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

或直接编辑资源：

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

最低配置：

```
spec:
  desiredUpdate:
    version: <target-version>
```

6 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`	最近一次预检运行的时间。
`status.preflight.checks`	每个预检项目的详细结果。
`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`	升级历史记录，最新记录在前。
`status.stages`	升级阶段及每个阶段的执行状态。

请先关注以下条件：

条件	解释
`PreflightReady`	`True` 表示预检已通过。
`Ready`	`True` 表示集群已达到所需版本。
`Reconciling`	`True` 表示升级仍在运行。

条件	解释
`Stalled`	`True` 表示升级已被阻止，需要进行干预。

如果 `Stalled=True` ，请参阅升级故障排除。要观察单个插件或 operator 模块，请读取其 `ModuleInfo` ：

```
# Current version, target, next available version, and phase per inst
alled module on this cluster
kubectl get moduleinfo -l cpaas.io/cluster-name=global \
  -o custom-columns='MODULE:.metadata.labels.cpaas\.io/module-name,CU
RRENT:.status.version,TARGET:.spec.version,NEW:.status.availableVersi
ons[0].version,PHASE:.status.phase'

# Conditions and any block reasons for one module
kubectl get moduleinfo <name> \
  -o jsonpath='{range .status.conditions[*]}{.type}{"\t"}{.status}
{"\t"}{.reason}{"\t"}{.message}{"\n"}{end}'
```

当模块的 `status.phase` 为 `Running` 且 `status.version` 等于目标版本时，表示该模块已达到目标版本。

7 从 Marketplace 升级 Agnostic 插件

CVO 驱动 Core 和 Aligned 插件。**Agnostic** 插件不在 **CVO** 的范围内，集群达到目标 Distribution Version 后，必须单独升级这些插件。每个 Agnostic 插件是否需要升级取决于其自身的 Kubernetes 兼容性 — 请参阅插件的 release notes，了解其与目标 Kubernetes 版本的兼容性。

对于 global 集群上正在使用的每个 Agnostic 插件：

1. 在 Web Console 中切换到 **Administrator** 视图。
2. 对于 Agnostic 集群插件，导航到 **Marketplace > Cluster Plugins**；对于 Agnostic operator，导航到 operator 操作流程。
3. 选择目标插件或 operator 并触发升级。Marketplace 升级流程会读取之前使用 `violet` 推送的软件包。

> 如果在升级前跳过了推送 Agnostic 插件，而 Marketplace 未提供目标版本，请先完成 `violet push` 步骤，然后重试 Marketplace 升级。

验证升级

集群达到所需版本并处理完正在使用的 Agnostic 插件后，请完成升级验证。如果 `global` 集群使用 Alauda OS，请先完成 在 Alauda OS 上升级 Kubernetes。

相关文档

- 概述
- 升级前
- 在 `global` 集群的 Alauda OS 上升级 Kubernetes
- 在 DR 环境中升级 global 集群
- 升级业务集群的 Distribution Version
- 升级验证
- 升级故障排除
- 升级集群
- global 集群灾难恢复

在 Alauda OS 上升级 `global` 集群的 Kubernetes

本文介绍如何升级运行 Alauda OS 的 `global` 集群。升级会使用 Cluster API provider 管理的新 Alauda OS 镜像替换节点；不会执行节点原地升级。

对于 Huawei DCS `global` 集群，请选择为目标 ACP 版本发布的 Alauda OS 镜像。如果集群的 VM 模板基于早于 ACP `v4.3.2` 版本发布的 Alauda OS 镜像创建，则该模板与 DCS Provider `v1.0.21` 或更高版本不匹配。请同时升级 provider 和 `global` 集群的 Distribution Version，因为平台会在该升级过程中升级已安装的 Aligned 插件；请参阅升级前准备。然后使用为目标 ACP 版本发布的 Alauda OS 镜像创建 VM 模板，并按照本文档替换节点。请参阅在 Huawei DCS 上升级 Kubernetes。

目录

- 何时使用此路径
- 两阶段升级概览
- 通用前提条件
- 操作步骤
  - 步骤 1 — 升级 Kube-OVN
  - 步骤 2 — 更新控制平面资源
  - 步骤 3 — 应用控制平面更改并等待
  - 步骤 4 — 更新工作节点资源
  - 步骤 5 — 应用工作节点更改
  - 步骤 6 — 监控滚动替换

验证

恢复注意事项

后续步骤

何时使用此路径

在以下情况下选择此升级路径：

- `global` 集群安装在 Alauda OS 上。请参阅安装 global 集群。
- 您的基础设施属于有文档说明的 provider 之一：Huawei DCS、VMware vSphere、Huawei Cloud Stack 或 Bare Metal。

对于非 Alauda OS `global` 集群，Kubernetes 会作为升级 Distribution Version 的一部分进行原地升级；本页面不适用。

在 DR 对中，请使用各自的 `global` kubeconfig，在两个 `global` 集群上分别执行此 Kubernetes 升级。有关该对集群的 Distribution Version 升级，请参阅在 DR 环境中升级。

两阶段升级概览

与业务集群一样，Alauda OS `global` 集群也遵循两阶段升级流程。

1. 阶段 **1 — ACP Core** 和 **Distribution Version**：准备构件和插件包，然后使用 CVO 将 global 层移动到目标 Distribution Version。请参阅升级前准备和升级 Distribution Version。
2. 阶段 **2 — Kubernetes** 和 **OS** 镜像：使用包含目标 Kubernetes 版本的新 Alauda OS 镜像替换节点。本文档重点介绍 `global` 集群的阶段 2。

> **Phase 1 Compatibility**
>
> 开始阶段 2 之前，请确认每个业务集群都处于目标 Distribution Version 的兼容版本矩阵范围内。超出范围的业务集群必须先升级。

通用前提条件

- `global` 集群已完成阶段 1（Distribution Version 升级）。
- `global` 集群所运行的 IaaS 的基础设施 provider 和 Cluster Bootstrap Provider 已上传并安装到 `global` 集群。如果尚未安装，请先按照 Bare Metal、Huawei DCS、Huawei Cloud Stack 或 VMware vSphere 的 provider 安装页面进行安装。在 DR 对中，请在两个 `global` 集群上都安装这些 provider。
- 基础设施 provider 版本、Alauda OS 镜像版本和 ACP 版本相匹配。阶段 1 会随 `global` 集群的 Distribution Version 一起升级 provider。
- 已备份并验证 `global` 集群的 etcd。
- OS Support Matrix 中的目标行可用，并且其中的 Kubernetes 版本、CoreDNS 标签、etcd 标签和 Kube-OVN chart 已准备就绪，同时已发布该版本对应的 Alauda OS 镜像。
- 保留之前的 machine template 和 bootstrap template，直到阶段 2 验证完成。
- 制定包含控制平面滚动替换的维护窗口计划。
- 对于跨越多个 Kubernetes 次要版本的跨版本升级，请预先准备中间版本的 Core 镜像和 OS 镜像。请参阅跨版本升级准备。

操作步骤

用于管理 `global` 集群的 Cluster API 控制器运行在 `global` 集群本身上。在本操作步骤中，请对 `kubectl` 命令使用 `global` kubeconfig。

> **Required when the target is Kubernetes 1.35**
>
> 在 Kubernetes 1.35 中，kubelet 凭证验证可能会应用于节点上已经存在的镜像。在升级到 Kubernetes 1.35 或更高版本时，请将 `imagePullCredentialsVerificationPolicy: NeverVerify` 添加到每个替换节点使用的 kubelet patch 中。在同一 `KubeadmControlPlane` 编辑中，同时更新控制平面文件和 `spec.version`。对于工作节点，请创建包含更新后 patch 的新 `KubeadmConfigTemplate`，并在与版本相同的编辑中切换 `MachineDeployment.spec.template.spec.bootstrap.configRef.name`。对于 Kubernetes 1.34 或更早版本，请勿添加此参数。请参阅 Kubernetes 1.35 所需的 kubelet patch。

步骤 1 — 升级 Kube-OVN

从目标 OS Support Matrix 行中读取 **kube-ovn (chart)** 值，然后按照共享的特定 provider 版本的 Kube-OVN 操作步骤进行操作，并将 `<cluster-name>` 设置为 `global`。

该操作步骤会先检查已安装的 provider 修订版本，然后选择受支持的新流程或旧流程，处理 `v4.4` 处的 Kube-OVN chart 名称边界，并验证 chart 名称、目标修订版本、已安装修订版本、阶段和条件。对于 provider 检查以及 `cni-kube-ovn` `AppRelease` 检查，请使用 `global` kubeconfig。只有在所有共享验证条件均通过后，才能开始替换控制平面。

步骤 2 — 更新控制平面资源

更新 `global` 集群的控制平面资源，使其引用新的 Alauda OS 镜像和 Kubernetes 版本。需要更新的字段因 provider 而异。在步骤 4 中更新工作节点资源。

Huawei DCS

开始之前，请参阅前提条件。

对于 DCS，请创建新的基础设施模板，而不要编辑正在运行的机器已引用的模板。

更新控制平面资源：

- 为目标镜像创建新的 `DCSMachineTemplate`，并将 `spec.template.spec.vmTemplateName` 设置为与目标 Kubernetes 版本匹配的 Alauda OS 模板。
- 将保留的节点本地数据（包括 `/var/cpaas`）保留在 `DCSIpHostnamePool.spec.pool[].persistentDisk` 中。不要将保留的磁盘移回 `DCSMachineTemplate`。
- 将 `KubeadmControlPlane.spec.version` 设置为目标 Kubernetes 版本。
- 根据同一 OS Support Matrix 行，在 `KubeadmControlPlane.spec.kubeadmConfigSpec.clusterConfiguration` 中设置 CoreDNS 和 etcd 镜像标签。
- 当目标为 Kubernetes 1.35 或更高版本时，在同一清单编辑中更新 `KubeadmControlPlane.spec.kubeadmConfigSpec.files` 中的控制平面 kubelet patch。

- 将 `KubeadmControlPlane.spec.machineTemplate.infrastructureRef.name` 指向新的 `DCSMachineTemplate`。
- 当集群使用池管理的持久磁盘时，保留 `KubeadmControlPlane.spec.rolloutStrategy.rollingUpdate.maxSurge: 0`。

池管理的持久磁盘在 IP 池上声明，而不是在 machine template 上声明：

```
apiVersion: infrastructure.cluster.x-k8s.io/v1beta1
kind: DCSIpHostnamePool
metadata:
  name: <global-pool-name>
  namespace: cpaas-system
spec:
  pool:
    - ip: <node-ip>
      hostname: <node-hostname>
      persistentDisk:
        - slot: 0
          quantityGB: 40
          datastoreName: <datastore-name>
          path: /var/cpaas
          format: xfs
          mountOptions:
            - defaults
```

仅当 DCS 环境要求显式的精简配置值时，才使用 `isThin`。如果省略，provider 不会发送 `isThin`，DCS 将使用平台默认值。新持久卷将创建为独立的持久普通卷。

使用 IP 池状态确认在滚动替换期间，保留的磁盘已从旧 VM 分离并连接到替换 VM。

VMware vSphere

开始之前，请参阅根据准入规则检查清单。

对于 vSphere，请创建新的基础设施模板，而不要编辑正在运行的机器已引用的模板。

更新控制平面资源：

- 导出当前的 `VSphereMachineTemplate`，为副本指定新名称，移除服务器生成的元数据和 `status`，并将 `spec.template.spec.template` 设置为在 vSphere 中根据为目标 ACP 版本发布的 Alauda OS 镜像创建的 VM 模板。
- 保留现有的 vCenter、文件夹、数据存储、网络、克隆模式、CPU、内存和磁盘配置，除非版本操作步骤明确要求更改。保持 `spec.template.spec.providerID` 未设置。
- 将保留的节点本地数据（包括 `/var/cpaas`）保留在 `VSphereMachineConfigPool.spec.configs[].persistentDisks[]` 中；不要将其移入 machine template。
- 根据同一 OS Support Matrix 行设置 `KubeadmControlPlane.spec.version`、CoreDNS 和 etcd 标签。
- 当目标为 Kubernetes 1.35 或更高版本时，在同一清单编辑中更新 `KubeadmControlPlane.spec.kubeadmConfigSpec.files` 中的控制平面 kubelet patch。
- 将 `KubeadmControlPlane.spec.machineTemplate.infrastructureRef.name` 指向新的 `VSphereMachineTemplate`。
- 当控制平面池使用固定槽位或池管理的持久磁盘时，保留 `KubeadmControlPlane.spec.rolloutStrategy.rollingUpdate.maxSurge: 0`。

Huawei Cloud Stack

开始之前，请参阅前提条件。

对于 HCS，请创建新的基础设施模板，而不要编辑正在运行的机器已引用的模板。

更新控制平面资源：

- 为目标镜像创建新的 `HCSMachineTemplate`，并将 `spec.template.spec.imageName` 设置为与目标 Kubernetes 版本匹配的 Alauda OS 镜像。
- 将保留的节点本地数据（包括 `/var/cpaas`）保留在 `HCSMachineConfigPool.spec.configs[].persistentDisks[]` 中。不要将保留的磁盘移回 `HCSMachineTemplate.spec.template.spec.dataVolumes[]`。
- 在新模板中保持运行时身份字段未设置，包括 `spec.template.spec.providerID` 和 `spec.template.spec.serverId`。

- 将 `KubeadmControlPlane.spec.version` 设置为目标 Kubernetes 版本。
- 根据同一 OS Support Matrix 行，在 `KubeadmControlPlane.spec.kubeadmConfigSpec.clusterConfiguration` 中设置 CoreDNS 和 etcd 镜像标签。
- 当目标为 Kubernetes 1.35 或更高版本时，在同一清单编辑中更新 `KubeadmControlPlane.spec.kubeadmConfigSpec.files` 中的控制平面 kubelet patch。
- 将 `KubeadmControlPlane.spec.machineTemplate.infrastructureRef.name` 指向新的 `HCSMachineTemplate`。
- 当控制平面池使用池管理的持久磁盘时，保留 `KubeadmControlPlane.spec.rolloutStrategy.rollingUpdate.maxSurge: 0`。

池管理的持久磁盘在 machine configuration pool 上声明，而不是在 machine template 上声明：

```yaml
apiVersion: infrastructure.cluster.x-k8s.io/v1beta1
kind: HCSMachineConfigPool
metadata:
  name: <global-pool-name>
  namespace: cpaas-system
  labels:
    cluster.x-k8s.io/cluster-name: global
spec:
  configs:
    - hostname: <node-hostname>
      networks:
        - subnetName: <subnet-name>
          ipAddress: <node-ip>
      persistentDisks:
        - slot: 0
          size: 100
          type: SSD
          mountPath: /var/cpaas
          format: xfs
          mountOptions:
            - defaults
            - noatime
```

在节点替换期间，不要将 HCS `dataVolumes[]` 视为保留状态。此滚动升级工作流支持高可用控制平面，该控制平面可以替换 `HCSMachineTemplate` 和 `KubeadmControlPlane` 引用，同时 `HCSMachineConfigPool` 保留足够的固定身份和持久磁盘槽位以完成发布。单控制平面 HCS 集群（包括只有一个控制平面节点的 `global` 集群）无法升级。

Bare Metal

开始之前，请参阅检查 COS_STATE 可用空间和更新镜像目录。

仅当 Bare Metal provider 已安装在 `global` 集群上，并且已安装版本中存在目标 OS Support Matrix 行时，才能使用此 YAML 工作流。在编辑 Cluster API 资源之前，请确认 provider 的 `AppRelease` 请求修订版本和已安装修订版本相匹配，并且其 chart 阶段为 `Success`。

将目标版本的 Bare Metal OS 镜像导入 `global` 集群的 registry，并按照已导入 OS 镜像并填充镜像目录中的说明，将目标 Kubernetes 版本添加到其 `elemental-image-catalog`。provider 插件和 Distribution Version 升级都不会为您添加该版本。在 DR 对中，请在两个 `global` 集群上执行此操作。然后验证目录：

```
kubectl --kubeconfig <global-kubeconfig> -n cpaas-system \
  get configmap elemental-image-catalog -o yaml
```

目标版本必须作为键出现，并且具有可访问的基础镜像引用。在缺少该键时创建的替换 `Machine` 会使其 `BaremetalMachine` 保持在 `Failed` 中，并带有 `Reason=ImageCatalogMiss`；之后再添加该键也无法恢复它。Bare Metal provider 从 `Machine.spec.version` 解析替换镜像；除非控制平面或工作节点必须移动到其他 `MachineInventoryPool`，否则仅升级 Kubernetes 不需要新的 `BaremetalMachineTemplate`。

使用同一 OS Support Matrix 行中的值更新现有的 `KubeadmControlPlane` 清单。保留此处未显示的所有字段：

```
spec:
  version: <target-kubernetes-version>
  rolloutStrategy:
    rollingUpdate:
      maxSurge: 0
  kubeadmConfigSpec:
    clusterConfiguration:
      dns:
        imageTag: <target-coredns-tag>
      etcd:
        local:
          imageTag: <target-etcd-tag>
```

保留现有的 `spec.kubeadmConfigSpec.files` 条目。对于 Kubernetes 1.35 或更高版本，按照上述说明更新该列表中现有的 `kubeletconfiguration0+strategic.json` 条目；不要将该列表替换为空值。

在步骤 3 中应用完整的保留控制平面清单。provider 会删除每台旧机器、清理其 inventory、从同一池中分配可用 inventory，并为目标镜像写入 `reprovision` 计划。

步骤 3 — 应用控制平面更改并等待

将更新后的控制平面清单应用到 `global` 集群。

```
kubectl --kubeconfig <global-kubeconfig> apply -f <updated-control-plane-manifest>
```

Cluster API provider 开始使用新镜像替换控制平面节点。当设置了 `maxSurge: 0` 时，每个旧节点都会先被驱逐并删除，然后其替换节点才能重新使用相同的固定身份、IP 地址或保留的磁盘。

等待所有控制平面节点完成替换。

```
kubectl --kubeconfig <global-kubeconfig> get machines -A -o wide
kubectl --kubeconfig <global-kubeconfig> get kubeadmcontrolplane -A
```

当每个控制平面 `Machine` 都报告新的 Kubernetes 版本和 `Phase: Running`，并且 `KubeadmControlPlane` 针对新版本报告 `Ready: True` 时，控制平面替换即完成。

步骤 4 — 更新工作节点资源

更新 `global` 集群的工作节点资源，使其引用新的 Alauda OS 镜像和 Kubernetes 版本。需要更新的字段因 provider 而异。

Huawei DCS

更新工作节点资源：

- 使用目标 `vmTemplateName` 创建新的工作节点 `DCSMachineTemplate`。
- 将每个 `MachineDeployment.spec.template.spec.version` 设置为目标 Kubernetes 版本。
- 当目标为 Kubernetes 1.35 或更高版本时，创建包含所需 kubelet patch 的新工作节点 `KubeadmConfigTemplate`，并将 `MachineDeployment.spec.template.spec.bootstrap.configRef.name` 指向它。
- 将每个 `MachineDeployment.spec.template.spec.infrastructureRef.name` 指向新的工作节点 `DCSMachineTemplate`。
- 当工作节点池使用池管理的持久磁盘时，保留每个 `MachineDeployment.spec.strategy.rollingUpdate.maxSurge: 0`。

VMware vSphere

更新工作节点资源：

- 使用目标 VM 模板和保留的 provider 配置创建新的工作节点 `VSphereMachineTemplate`。
- 将每个 `MachineDeployment.spec.template.spec.version` 设置为目标 Kubernetes 版本。
- 当目标为 Kubernetes 1.35 或更高版本时，创建包含所需 kubelet patch 的新工作节点 `KubeadmConfigTemplate`，并将 `MachineDeployment.spec.template.spec.bootstrap.configRef.name` 指向它。

- 将每个 `MachineDeployment.spec.template.spec.infrastructureRef.name` 指向新的工作节点 `VSphereMachineTemplate`。
- 当工作节点池使用固定槽位或池管理的持久磁盘时，保留每个 `MachineDeployment.spec.strategy.rollingUpdate.maxSurge: 0`。

不要编辑正在运行的机器引用的模板。保留之前的控制平面和工作节点模板，直到新的发布经过验证。

Huawei Cloud Stack

更新工作节点资源：

- 使用目标 `imageName` 创建新的工作节点 `HCSMachineTemplate`。
- 将每个 `MachineDeployment.spec.template.spec.version` 设置为目标 Kubernetes 版本。
- 当目标为 Kubernetes 1.35 或更高版本时，创建包含所需 kubelet patch 的新工作节点 `KubeadmConfigTemplate`，并将 `MachineDeployment.spec.template.spec.bootstrap.configRef.name` 指向它。
- 将每个 `MachineDeployment.spec.template.spec.infrastructureRef.name` 指向新的工作节点 `HCSMachineTemplate`。
- 当工作节点池使用池管理的持久磁盘时，保留每个 `MachineDeployment.spec.strategy.rollingUpdate.maxSurge: 0`。

Bare Metal

对于每个工作节点 `MachineDeployment`，设置目标 Kubernetes 版本并保留 `maxSurge: 0`。当目标为 Kubernetes 1.35 或更高版本时，先创建包含所需 kubelet patch 的新 `KubeadmConfigTemplate`，然后在同一清单更改中引用它：

```
spec:
  strategy:
    rollingUpdate:
      maxSurge: 0
  template:
    spec:
      version: <target-kubernetes-version>
      bootstrap:
        configRef:
          name: <target-kubeadm-config-template>
```

如果更早的目标版本不需要更改 bootstrap 设置，则省略 `bootstrap.configRef` 更改。在步骤 5 中应用完整的保留工作节点清单。

步骤 5 — 应用工作节点更改

将更新后的工作节点清单应用到 `global` 集群。

```
kubectl --kubeconfig <global-kubeconfig> apply -f <updated-worker-manifest>
```

Cluster API provider 开始使用新镜像替换工作节点。当设置了 `maxSurge: 0` 时，每个旧节点都会先被驱逐并删除，然后其替换节点才能重新使用相同的固定身份、IP 地址或保留的磁盘。

步骤 6 — 监控滚动替换

监控滚动替换，直到所有工作节点都完成替换。

```
kubectl --kubeconfig <global-kubeconfig> get machines -A -o wide
kubectl --kubeconfig <global-kubeconfig> get kubeadmcontrolplane -A
```

当每个 `Machine` 都报告新的 Kubernetes 版本和 `Phase: Running`，并且 `KubeadmControlPlane` 针对新版本报告 `Ready: True` 时，升级即完成。

验证

滚动替换完成后，验证已升级的 `global` 集群是否运行正常。

```
kubectl --kubeconfig <global-kubeconfig> get nodes -o wide
kubectl --kubeconfig <global-kubeconfig> get clusterversionshadow -o yaml
kubectl --kubeconfig <global-kubeconfig> get pods -n cpaas-system
```

所有节点都必须报告新的 Kubernetes 版本，`ClusterVersionShadow` 必须反映目标 Distribution Version，且核心平台 Pod 必须处于 `Running` 状态。

恢复注意事项

不要将 Kubernetes 次要版本降级视为普通回滚。请根据发布阶段选择恢复路径：

1. 尚未创建目标版本控制平面的 `Machine`：恢复之前的 Kube-OVN 状态和之前的清单值。在引入新的控制平面数据格式之前，取消目标版本发布。
2. 仅更改了机器模板或 **OS** 镜像，且 **Kubernetes** 次要版本未更改：将控制资源指回之前的模板，并让 Cluster API 执行另一次替换发布。保持 Kubernetes 次要版本不变。
3. 目标 **Kubernetes** 次要版本的控制平面 `Machine` 已加入集群：不要将 Kubernetes、CoreDNS 或 etcd 修补回之前的次要版本。停止进一步发布，并在目标次要版本上向前修复；或者通过已验证的升级前 etcd 备份或 Global DR 操作恢复 `global` 集群。

如果已创建目标次要版本的控制平面 `Machine`，但它从未加入集群，请先恢复正常的 etcd quorum，并在更改清单之前确定是否可以安全移除失败的替换节点。

Huawei DCS

对于使用池管理持久磁盘的 DCS 集群，请在回滚前确认磁盘状态：

首先，在删除或重新创建机器之前检查 `DCSIpHostnamePool.status.persistentDiskStatus`。不要删除 `DCSIpHostnamePool.spec.pool[].persistentDisk` 中列出的保留 DCS 卷。

对于阶段 1，请使用在阶段 1 恢复期间恢复 Kube-OVN 中的 DCS 规则；具体操作取决于已安装的 provider 版本。在同一次要版本的替换恢复过程中返回之前的机器模板时，保留

`maxSurge: 0`。如果目标次要版本的控制平面机器已加入集群，请保留目标基线，并使用向前恢复或备份恢复。

VMware vSphere

对于阶段 1，请使用在阶段 1 恢复期间恢复 Kube-OVN 中的 vSphere 规则。对于同一次要版本的模板恢复，请保留 `maxSurge: 0`，将控制平面和工作节点资源指向保留的之前的 `VSphereMachineTemplate` 资源，并验证每个固定插槽都重新连接了其池管理持久磁盘。如果目标次要版本的控制平面机器已加入集群，请保留目标 Kubernetes、CoreDNS、etcd 和 Kube-OVN 基线，并使用向前恢复或备份恢复。

Huawei Cloud Stack

HCS `dataVolumes[]` 上的节点本地数据不是恢复源，因为节点替换可能会删除旧 VM 及其附加卷。在 `HCSMachineConfigPool.spec.configs[].persistentDisks[]` 中声明的数据会在替换期间重新连接。对于阶段 1，请使用在阶段 1 恢复期间恢复 Kube-OVN 中的 HCS 规则；具体操作取决于已安装的 provider 版本。对于同一次要版本的模板恢复，请保留 `maxSurge: 0`，并将控制资源指向保留的之前的 `HCSMachineTemplate` 资源。如果目标次要版本的控制平面机器已加入集群，请保留目标基线，并使用向前恢复或备份恢复。

Bare Metal

对于阶段 1，请使用在阶段 1 恢复期间恢复 Kube-OVN 中的 Bare Metal 规则，然后仅当尚未创建目标次要版本控制平面的 `Machine` 时，才恢复之前的清单。

每个完成 `reprovision` 计划的主机都已替换其 OS 镜像并清除了 Kubernetes 管理的状态。如果目标次要版本的控制平面机器已加入集群，请不要将 Kubernetes、CoreDNS 或 etcd 修补回之前的次要版本。向前修复，或通过已验证的 etcd 备份或 Global DR 操作恢复 `global` 集群。

后续步骤

- 将业务集群升级到相同的 Distribution Version：请参阅升级业务集群的 Distribution Version，然后参阅在 Alauda OS 上升级业务集群的 Kubernetes。
- 查看新镜像附带的机器配置变更：请参阅机器配置。

在 DR 环境中升级全局集群

当环境同时包含主全局集群和备用全局集群时，请使用此操作步骤。DR 特定步骤是在升级全局集群中的标准 CVO 工作流基础上执行的附加步骤。

目录

升级操作步骤

- 升级前验证 DR 环境
- 从备用全局集群卸载 etcd 同步插件
- 在两个全局集群上同步升级构件
- 升级备用全局集群
- 升级主全局集群
- 重新安装 etcd 同步插件并验证同步状态

升级操作步骤

1 升级前验证 DR 环境

按照常规的全局 DR 检查操作步骤，确保备用全局集群中的数据与主全局集群一致。有关 DR 拓扑和同步工作流的背景信息，请参阅全局集群灾难恢复。

如果检测到不一致，请勿在下一步卸载 etcd 同步插件，并在继续之前联系技术支持。在备用全局集群缺少主全局集群所持有的数据时卸载插件，可能导致所有者引用解析错误，

并可能删除工作负载集群节点 `Machine` 对象，包括 Alauda OS 集群；在这种情况下，删除操作会销毁其后端虚拟机。

在两个全局集群上运行以下命令，确保没有 `Machine` 节点处于非运行状态：

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

如果存在此类节点，请先解决这些问题，然后再继续。

2 从备用全局集群卸载 etcd 同步插件

1. 通过 IP 或 VIP 访问备用全局集群的 Web Console。
2. 切换到管理员视图。
3. 导航到**Marketplace >** 集群插件，并选择 `global` 集群。
4. 找到 灵雀云容器平台 全局集群灾难恢复，然后将其卸载。
5. 等待卸载完成后再继续。

3 在两个全局集群上同步升级构件

在备用全局集群和主全局集群上完成同步升级构件。在两个集群上使用相同的 registry 类型和完全一致的目标版本负载。对于外部 registry，请为两个全局集群使用的每个 registry 端点完成两种上传模式。

4 升级备用全局集群

如果要在备用全局集群上使用 Web Console，请验证备用集群 `ProductBase` 在 `spec.alternativeURLs` 中包含备用 VIP：

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

同步完成后，在备用全局集群上执行其余标准工作流步骤：

1. 运行升级前检查。
2. 部署集群版本 operator。
3. 请求升级。
4. 观察执行过程，直到备用全局集群达到目标版本。

5 升级主全局集群

备用全局集群达到目标版本后，在主全局集群上执行其余标准工作流步骤：

1. 运行升级前检查。
2. 部署集群版本 operator。
3. 请求升级。
4. 观察执行过程，直到主全局集群达到目标版本。

6 重新安装 **etcd** 同步插件并验证同步状态

重新安装插件之前，请验证在使用该转发模式时，端口 `2379` 是否已从两个全局集群 VIP 正确转发到其控制平面节点。如果备用全局集群可以直接访问活动全局集群，则无需通过负载均衡器进行端口转发。

从活动集群获取活动全局集群 API 服务器的 bearer 令牌，然后使用该令牌在备用集群上创建或更新 `etcd-sync-active-cluster-token` Secret。仅在当前 shell 中保留该令牌，并在创建 Secret 后将其清除。

```
# Run on the active cluster and copy the output without storing it in
a document.
kubectl -n cpaas-system get secret k8sadmin -o jsonpath='{.data.toke
n}' | base64 -d
```

在备用集群上运行：

```
read -rsp "Active global cluster token: " ACTIVE_CLUSTER_TOKEN
printf '\n'
kubectl -n cpaas-system create secret generic etcd-sync-active-cluste
r-token \
  --from-literal=token="${ACTIVE_CLUSTER_TOKEN}" \
  --dry-run=client -o yaml | kubectl apply -f -
unset ACTIVE_CLUSTER_TOKEN
```

要重新安装插件：

1. 通过 VIP 访问备用全局集群的 Web Console，并切换到管理员视图。
2. 导航到**Marketplace >** 集群插件，并选择 `global` 集群。
3. 找到 灵雀云容器平台 全局集群灾难恢复，单击安装，并配置所需参数。

配置插件时：

- 将活动全局集群 **VIP** 设置为活动全局集群的 VIP。
- 当端口 `2379` 未通过负载均衡器转发时，正确设置活动全局集群 **ETCD** 端点。
- 将备用集群 **ETCD** 端点设置为备用集群 etcd 地址。除非本地 etcd 服务通过其他端点公开，否则使用默认值。
- 将活动全局集群令牌 **Secret** 设置为 `etcd-sync-active-cluster-token`。
- 使用数据检查间隔的默认值。
- 除非正在进行故障排除，否则不要启用打印详细日志。

重新安装期间，系统会先运行 `etcd-sync-bootstrap` Job，然后启动 `etcd-sync` Deployment。验证引导 Job 和运行时资源：

```
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
```

在备用全局集群上验证同步 Pod 和当前 leader：

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

如果需要重新同步具有 `ownerReference` 依赖项的资源，请在 `Start Sync update` 出现后重新创建当前 leader Pod：

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

- `LOCAL ETCD missed keys`：主全局集群中存在这些键，但备用集群中缺少这些键。重启当前 `etcd-sync` leader Pod 后通常可以解决此问题。
- `LOCAL ETCD surplus keys`：备用全局集群中存在这些键，但主集群中不存在。请与运维团队一起检查这些键，然后再删除。

验证成功后，从插件设置或 release 值中移除所有剩余的旧版明文令牌配置。如果活动集群令牌或远程 `etcd-ca` 后续发生更改，请再次运行插件升级或重新安装工作流，以便 `etcd-sync-bootstrap` 刷新运行时凭据和证书。

两个全局集群在目标版本上均运行正常且同步已恢复后，继续执行升级工作负载集群或升级验证。如果 `global` 集群运行 Alauda OS，请先在两个 `global` 集群上运行 Alauda OS 上的

Kubernetes 升级。

升级工作负载集群

升级发行版

升级发行版

在 Alauda OS 上升级 Kubernetes

裸金属	**Huawei DCS**	华为云 **Stacl**
VMware vSphere	将现有 **DCS** 集群迁移到池管理的	将现有 **HCS**

升级业务集群的发行版

> **升级路径**
>
> 此页面适用于两种节点操作系统。对于使用非 Alauda OS 的业务集群，此操作步骤还会原地升级 Kubernetes；操作完成后，集群升级即告完成。对于使用 Alauda OS 的业务集群，请先完成此页面上的操作，然后继续执行在 Alauda OS 上升级 Kubernetes，通过替换节点来推出 Kubernetes。

global 层已达到目标 ACP 发行版后，即可升级业务集群。如果 global 层已达到该发行版，则无需再次升级。

业务集群升级采用与 `global` 集群相同的基于 CVO 的工作流：确认前提条件、运行预检、请求升级并观察执行情况。此工作流还有两条额外规则：

- 如果平台使用 **global DR**，则 standby 和 primary global 集群都必须达到目标发行版，然后才能将任何业务集群升级到该发行版。
- 通常只有在 global 层已达到相同发行版后，目标版本才会对业务集群开放。

业务集群升级的阶段安排与 global 集群升级类似，但流程更短。业务集群没有自己的 cluster version operator — CVO 在 global 集群上运行，并集中驱动业务集群升级 — 因此业务集群本身不需要执行制品同步或 CVO 部署步骤：

阶段	时间	操作
1. 预检	升级窗口前 1–2 周	确认前提条件并运行 `upgrade.sh --cluster=<cluster> --preflight`；在升级窗口开启前解决所有阻塞项。
2. 升级	维护窗口期间	请求升级并观察执行情况，直到集群达到目标版本。

目录

工作流

- 确认业务集群前提条件
- 确保 Aligned operator 软件包可用
- 运行预检
- 请求升级
- 观察进度
- 从 Marketplace 升级 Agnostic 插件

验证升级

相关文档

工作流

1 确认业务集群前提条件

请求升级前，请确认：

- 业务集群版本低于当前 `global` 集群版本。
- global 层已达到目标 ACP 发行版。
- 在 global DR 环境中，standby 和 primary global 集群都已达到目标发行版。

2 确保 **Aligned operator** 软件包可用

升级 global 集群中推荐的做法是在 global 集群升级窗口期间，将 operator 软件包推送到每个 ACP 集群。如果已遵循此建议，业务集群就已拥有所需的全部 operator 软件包，可以跳过本节。

集群插件无需针对每个业务集群单独发布。通过 `upgrade.sh` 或 `violet` 提供给 global 层的集群插件，可在平台上的所有集群中安装和升级。

如果 global 窗口期间未向此业务集群推送 operator 软件包（例如，该集群在 global 窗口期间处于离线状态，或是在之后才添加的），请在提交升级请求前，为此业务集群上安装

的每个 Aligned operator 推送目标版本的软件包：

```
violet push <path/to/operator-package> \
  --platform-address "https://<your-platform-domain>" \
  --platform-token "<platform_token>" \
  --clusters "<workload-cluster-name>"
```

只需推送此业务集群上实际安装的 Aligned operator 软件包。提供给 global 层的 Aligned 集群插件软件包对所有业务集群都可用，无需针对每个集群单独发布。

如果已安装的 Aligned operator 的目标软件包不可用，CVO 可能会等待匹配的目标 `InstallPlan`，随后在重试现有请求时报告错误。请检查 operator 的 catalog、`Subscription`、`InstallPlan` 和 `ModuleInfo`；使用上述命令发布缺失的软件包，或修正报告的 catalog 或 registry 问题。继续观察现有请求，无需再次提交。此业务集群上未安装的 Aligned operator 软件包不会阻塞升级。

3 运行预检

时间：维护窗口前 1–2 周，以便在窗口开启前留出时间解决所有阻塞项。

对目标集群以预检模式运行 `upgrade.sh`：

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

预检为只读操作，只验证升级就绪情况，不会更改集群状态。

预检会返回两部分结果：

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`	显示各项验证的结果。

如果预检未通过，在解决所有阻塞项之前，请勿提交升级请求。

有关预检阻塞项的处理模式，请参阅升级故障排除。

4 请求升级

时间：维护窗口期间，且预检通过之后。

通过以下任一入口请求升级。这些入口功能相同，请根据您的运维模式选择合适的入口。

平台在 Web Console 中提供两种集群管理界面。**Web Console** 是平台内置的界面，也是默认入口。**Web Console (Fleet Essentials)** 是较新的集群管理界面，以独立集群插件的形式交付，使用前必须先安装。

Web Console

这是内置的集群管理界面。业务集群升级使用与 global 集群升级相同的页面，但页面打开时选择的是业务集群，而不是 `global`。

请求升级需要对目标集群的 `ClusterVersionShadow` 和 `ResourcePatch` 资源具有更新权限。

1. 在左侧导航栏中，单击 **Clusters** > **Clusters**，然后单击业务集群。
2. 打开 **Functional Components** 选项卡，然后单击 **Upgrade**。
   当没有可升级的版本（**No upgradable version**）、已有升级正在运行（**Upgrading**）或集群运行状况不佳（**Cluster Abnormality**）时，此按钮将被禁用。global 层达到目标发行版之前，业务集群不会有可升级的版本。
3. 在 **Personalized Configuration Confirmation** 步骤中，检查列出的配置，然后单击 **Acknowledge**。
   仅当集群包含个性化配置时才会显示此步骤，包括自定义的 `ModuleInfo`、自定义的 `ClusterModule` 或 `ResourcePatch`。如果集群不包含这些配置，则不会显示此步骤，页面会直接打开确认表单。
4. 在 **Confirm Component Upgrade** 步骤中，检查 **Current Version** 和 **Target Version**。
   对于业务集群，目标版本是 global 层已达到的发行版，无法在此界面中手动选择。
5. 单击 **Upgrade**，然后在 **Are you sure you want to upgrade the cluster version?** 对话框中确认。

> 默认开启维护模式，且会应用于整个平台

此确认对话框与 global 集群升级共用，打开时已启用 **Maintenance mode**。此开关设置的是平台范围的维护模式，而非业务集群的设置：启用后，非平台管理员用户将无法访问整个平台上的功能，并会看到 **Platform maintenance announcements** 中的文本。除非您打算在此次业务集群升级期间限制整个平台，否则请在提交前关闭此开关。

提交请求后，页面会返回 **Functional Components** 选项卡，并显示 *The upgrade has been initiated. It may take 5~10 minute for the status to update.*

Web Console (Fleet Essentials)

此入口属于新的集群管理界面，以 **Alauda Container Platform Fleet Essentials** 集群插件的形式交付。默认不会安装。

前提条件：从 **Marketplace** > **Cluster Plugins** 安装 **Alauda Container Platform Fleet Essentials v1.0.4** 或更高版本。该插件会在左侧导航栏中添加 **Clusters (New)** 入口。如果没有此入口，说明此平台未安装该插件，请改用 **Web Console** 选项卡。

1. 在左侧导航栏中，单击 **Clusters (New)**，然后单击业务集群。
2. 停留在 **Overview** 选项卡，找到 **Version & Update** 卡片。
   global 层尚未达到目标发行版时，横幅会显示 **Target not ready**。在 global DR 环境中，primary 和 standby global 集群都必须先达到目标版本。
3. 单击 **Upgrade**，打开 **Update Cluster** 对话框。
4. 在 **Custom Configuration** 步骤中，检查待处理的 `ResourcePatch` 条目，这些条目将在升级到目标版本时被豁免，然后单击 **Next**。
5. 在 **Confirm & Upgrade** 步骤中，检查 **Current Version** 和 **Target Version**。
   对于业务集群，目标版本与当前 global 集群版本相同，且为只读 — `global` 集群中提供的下拉列表不适用于此处。
6. 单击 **Confirm & Start**。
   此界面没有第二个确认对话框，也没有维护模式开关。卡片会刷新并进入 **Upgrading** 状态，卡片中的 **Preflight Checks**、**Upgrade Progress** 和 **Update History** 区域会显示此次升级的进度。

ACP CLI

如果当前 ACP 会话可以访问目标业务集群，请使用 ACP CLI。为避免歧义，请使用 `--cluster` 明确指定目标集群。ACP CLI 不用于升级 `global` 集群。

```
# 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=<target-version>
```

ACP CLI 会为指定的业务集群提交所请求的目标版本。能否继续升级，仍取决于该集群可用的升级目标以及升级控制器的预检结果。

如果您刚开始使用 ACP CLI，请参阅ACP CLI 入门。有关会话和集群选择，请参阅管理 CLI 配置文件。有关完整的 AC CLI 升级工作流（元数据准备、可用更新、请求模式和状态解读），请参阅升级集群。有关完整的命令和标志语法，请参阅AC CLI 管理员命令参考。

5 观察进度

提交升级请求后，使用 `cvsh.status` 跟踪当前版本、目标版本、预检结果、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，也可以查看 CLI 报告的状态：

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

有关 `ac adm upgrade status` 输出（预检和阶段解读）的详细说明，请参阅升级集群。

如果升级报告 `Stalled=True`，请使用升级故障排除。缺失的 Aligned operator 软件包与集群插件软件包有不同的恢复路径。

上述命令用于跟踪集群级（Core 和 Aligned）升级。若要观察单个插件或 operator 模块（例如升级 Agnostic 插件时），请读取其 `ModuleInfo`：

```
kubectl get moduleinfo -l cpaas.io/cluster-name=<cluster> \
  -o custom-columns='MODULE:.metadata.labels.cpaas\.io/module-name,CURRENT:.status.version,TARGET:.spec.version,NEW:.status.availableVersions[0].version,PHASE:.status.phase'
```

当模块的 `status.phase` 为 `Running`，且 `status.version` 等于目标版本时，即表示模块已达到目标版本。

6 从 Marketplace 升级 Agnostic 插件

CVO 会驱动 Core 和 Aligned 插件。**Agnostic** 插件不在 **CVO** 的范围内，必须在此业务集群达到目标发行版后逐一升级。

对于此业务集群上正在使用的每个 Agnostic 插件：

1. 在 Web Console 中，切换到 **Administrator** 视图。
2. 对于 Agnostic 集群插件，导航到 **Marketplace > Cluster Plugins**；对于 Agnostic operator，导航到 operator 工作流。
3. 选择目标插件或 operator，然后触发升级。

每个 Agnostic 插件是否需要在此窗口期间升级，取决于其自身的 Kubernetes 兼容性 — 请查看插件的 release notes，确认其与业务集群已升级到的新 Kubernetes 版本的兼容性。

如果 Marketplace 未向此业务集群提供 operator 的目标版本，请先按照确保 Aligned operator 软件包可用中的说明处理该 operator，然后重试 Marketplace 升级。

验证升级

业务集群及其正在使用的 Agnostic 插件均达到目标版本后，请完成升级验证。如果业务集群使用 Alauda OS，请先完成Alauda OS 上的 Kubernetes 升级。

相关文档

- 概述
- 升级 `global` 集群的发行版
- 升级业务集群 Alauda OS 上的 Kubernetes
- 升级验证
- 升级故障排除
- 升级集群

为业务集群升级 Alauda OS 上的 Kubernetes

本节介绍如何升级由 Cluster API 管理的 Alauda OS 业务集群上的 Kubernetes。

目录

升级概述
前提条件
阶段 1：升级 ACP Core 和 Distribution Version
升级后验证
阶段 2：升级 Kubernetes 版本
在控制平面之前升级 Kube-OVN
确定已安装的提供商版本
应用特定于提供商版本的操作步骤
验证 Kube-OVN AppRelease
在阶段 1 恢复期间恢复 Kube-OVN
Kubernetes 1.35 所需的 kubelet 补丁
准备跨版本升级
步骤 1 — 确认 Core 跳转并预先同步中间镜像
步骤 2 — 根据置备机制准备中间 OS 镜像
步骤 3 — 对每个中间 Kubernetes/OS 版本行应用特定于平台的操作步骤
特定于平台的说明
相关主题

升级概述

Alauda OS 上的集群升级分为两个阶段：

1. 阶段 **1**：通过 CVO 升级 ACP Core 和集群 Distribution Version。请参阅升级 Distribution Version。
2. 阶段 **2**：升级 Kubernetes，并使用匹配的 Alauda OS 镜像替换节点。本节介绍阶段 2。

> **INFO**
>
> 阶段 1 的前提条件、支持的 ACP 升级路径、制品准备和 CVO 操作步骤见下方阶段 1 中列出的页面。本节介绍阶段 2 中 Alauda OS 上的 Kubernetes 升级和 OS 替换操作步骤。

有关 Distribution Version 与 Kubernetes 和 OS 基线之间的关系，请参阅 Distribution Version 与 Kubernetes Version。阶段 2 中提供的、针对特定补丁版本的 Kubernetes 和组件确切值，请参阅 OS Support Matrix。

前提条件

- 完成升级前准备，包括验证源 ACP 版本能否升级到目标 Distribution Version。
- 在开始阶段 2 之前，完成 ACP Core 和 Distribution Version 升级。
- 验证控制平面可访问且运行正常。
- 验证所有节点均处于 `Ready` 状态。
- 按照特定于提供商的操作步骤创建并验证所需备份。
- 确认有足够的 IP 或机器槽位容量进行滚动替换。

阶段 1：升级 ACP Core 和 Distribution Version

使用以下页面作为本阶段的唯一操作步骤来源：

- 升级概述

- 升级前准备
- 升级 `global` 集群的 Distribution Version
- 升级业务集群的 Distribution Version

按此顺序准备制品和插件包，升级 `global` 层级，然后通过 CVO 将每个业务集群升级到目标 Distribution Version。请勿在阶段 2 中使用 Kubernetes 中间版本跳转，以绕过不受支持的 ACP Core 升级路径。

> **INFO**
>
> Fleet Essentials 1.0.4 及更高版本可以使用 ACP 4.3 及更高版本的 CVO 工作流来请求升级 Distribution Version。此功能仅适用于阶段 1。Fleet Essentials 不会替换不可变节点，也不会在阶段 2 中执行 Kubernetes 和 Alauda OS 的滚动升级；请使用本节中特定于提供商的 YAML 操作步骤。

升级后验证

Distribution Version 升级完成后：

- 验证集群报告的 Distribution Version 为目标版本，且 CVO 不再处于进行中或阻塞状态。
- 确认 OS Support Matrix 中存在目标行和所有必需的中间行。
- 只有在管理端的 Cluster API 提供商均运行正常后，才能继续进行阶段 2。

阶段 2：升级 **Kubernetes** 版本

升级 Distribution Version 后，通过特定于提供商的 Cluster API 操作步骤升级 Kubernetes 和 Alauda OS 镜像。

对于 Huawei DCS、Huawei Cloud Stack、VMware vSphere 和裸金属集群，在更改 `KubeadmControlPlane` 之前，先为当前升级跳转升级 Kube-OVN chart。等待业务集群的 `cni-kube-ovn` `AppRelease` 报告目标已安装修订版本且 `phase=Success`，然后升级控制平面，最后升级工作节点。在跨版本升级中，对每个中间 Kubernetes minor 版本重复此 Kube-OVN 优先顺序。

在控制平面之前升级 **Kube-OVN**

将本节作为业务集群和 `global` 集群的权威 Kube-OVN 操作步骤。对于每个 Kubernetes 和 OS 跳转，从匹配的 OS Support Matrix 行中读取 **kube-ovn (chart)** 版本，完成以下特定于提供商版本的操作步骤，并在更改 `KubeadmControlPlane` 之前验证生成的 `AppRelease`。

针对管理集群运行提供商版本检查以及 `Cluster` 或基础设施集群命令。针对正在升级的集群运行 `cni-kube-ovn` `AppRelease` 命令。升级 `global` 集群时，两个资源都位于 `global` 集群上，因此所有命令均使用 `global` kubeconfig。

确定已安装的提供商版本

请勿根据 ACP Distribution Version 推断提供商版本。在管理集群上检查提供商 `AppRelease`。资源名称可能有所不同，因此以下命令会按 chart 名称查找提供商，并打印请求的修订版本和已安装的修订版本：

```
kubectl -n cpaas-system get apprelease \
  -o custom-columns='APPRELEASE:.metadata.name,CHART:.spec.source.charts
[*].name,REQUESTED:.spec.source.charts[*].targetRevision,INSTALLED:.statu
s.charts[*].installedRevision,PHASE:.status.charts[*].phase'
```

根据实际情况，查找 chart 名称以 `chart-cluster-api-provider-dcs`、`chart-cluster-api-provider-hcs`、`chart-cluster-api-provider-vsphere` 或 `chart-cluster-api-provider-baremetal` 结尾的行。使用已安装的修订版本选择选项卡。如果请求的修订版本和已安装的修订版本不同，或者提供商 chart 不处于 `Success` 状态，请在开始升级集群之前完成或修复提供商升级。

Provider	完整源代码协调	本指南保留的旧版行为
Huawei DCS	`v1.0.22` 及更高版本	`v1.0.21` 及更早版本
Huawei Cloud Stack	`v1.0.4` 及更高版本	`v1.0.3` 及更早版本
VMware vSphere	`v1.0.16` 及更高版本	`v1.0.15` 及更早版本
Bare Metal	`v1.0.0` 及更高版本	无；`v1.0.0` 是此操作步骤支持的首个版本。

Kube-OVN chart 源代码还会在 chart 版本 `v4.4` 处发生变化：

目标 **Kube-OVN chart** 版本	预期的 `spec.source.charts[0].name`
早于 `v4.4`	`acp/chart-cpaas-kube-ovn`
`v4.4` 及更高版本	`acp/chart-kube-ovn`

如果目标 OS Support Matrix 行使用 Kube-OVN chart `v4.4` 或更高版本，请先将基础设施提供商升级到表中的完整源代码协调版本，然后再继续。手动 Kube-OVN 源代码补丁不能替代该提供商升级，因为相同的提供商版本还会协调所需的 CoreDNS 和 kube-proxy 仓库变更。DCS Self-built VIP 是单独的创建时端点功能，需要 DCS Provider `v1.0.22+` 和 ACP `v4.4+`；将现有外部端点更改为内部端点不属于本升级操作步骤。

应用特定于提供商版本的操作步骤

Huawei DCS

验证 DCS 基础设施集群使用 Kube-OVN：

```
kubectl get dcscluster <cluster-name> -n cpaas-system \
  -o jsonpath='{.spec.networkType}{"\n"}'
# Expected output: kube-ovn
```

DCS v1.0.22+

在 CAPI `Cluster` 上更新目标 chart 版本。提供商根据目标版本选择 chart 名称，并协调完整的 `AppRelease` 源代码和规范。

```
kubectl annotate cluster <cluster-name> -n cpaas-system \
  cpaas.io/kube-ovn-version=<kube-ovn-version-from-matrix> --overwrit
e
```

请勿手动修补 `AppRelease` 源代码。继续执行验证 Kube-OVN AppRelease。

DCS v1.0.21 and earlier

> **WARNING**
>
> 此旧版流程仅在目标 Kube-OVN chart 早于 `v4.4` 时有效。对于 `v4.4` 或更高版本的 chart，请先将 DCS provider 升级到 `v1.0.22` 或更高版本。

旧版提供商会更新 `spec.values`，但保留现有源代码。在 `Cluster` 上记录目标版本，然后仅在正在升级的集群上将现有旧版 chart 修订版本修补为目标版本：

```
kubectl annotate cluster <cluster-name> -n cpaas-system \
  cpaas.io/kube-ovn-version=<kube-ovn-version-from-matrix> --overwrit
e

kubectl patch apprelease cni-kube-ovn -n cpaas-system --type='json' \
  -p='[{"op":"replace","path":"/spec/source/charts/0/targetRevisio
n","value":"<kube-ovn-version-from-matrix>"}]'
```

保留旧版 chart 名称 `acp/chart-cpaas-kube-ovn`。请勿使用此补丁将旧提供商移动到 chart `v4.4` 或更高版本。

Huawei Cloud Stack

验证 HCS 基础设施集群使用 Kube-OVN：

```
kubectl get hcscluster <cluster-name> -n cpaas-system \
  -o jsonpath='{.spec.networkType}{"\n"}'
# Expected output: kube-ovn
```

HCS v1.0.4+

在 CAPI `Cluster` 上更新目标 chart 版本。提供商根据目标版本选择 chart 名称，并协调完整的 `AppRelease` 源代码和规范。

```
kubectl annotate cluster <cluster-name> -n cpaas-system \
  cpaas.io/kube-ovn-version=<kube-ovn-version-from-matrix> --overwrit
e
```

请勿手动修补 `AppRelease` 源代码。继续执行验证 Kube-OVN AppRelease。

HCS v1.0.3 and earlier

> **WARNING**
>
> 此旧版流程仅在目标 Kube-OVN chart 早于 `v4.4` 时有效。对于 `v4.4` 或更高版本的 chart，请先将 HCS provider 升级到 `v1.0.4` 或更高版本。

旧版提供商会更新 `spec.values`，但保留现有源代码。在 `Cluster` 上记录目标版本，然后仅在正在升级的集群上将现有旧版 chart 修订版本修补为目标版本：

```
kubectl annotate cluster <cluster-name> -n cpaas-system \
  cpaas.io/kube-ovn-version=<kube-ovn-version-from-matrix> --overwrit
e

kubectl patch apprelease cni-kube-ovn -n cpaas-system --type='json' \
  -p='[{"op":"replace","path":"/spec/source/charts/0/targetRevisio
n","value":"<kube-ovn-version-from-matrix>"}]'
```

保留旧版 chart 名称 `acp/chart-cpaas-kube-ovn`。请勿使用此补丁将旧提供商移动到 chart `v4.4` 或更高版本。

VMware vSphere

验证 CAPI `Cluster` 已启用 Kube-OVN 协调：

```
kubectl get cluster <cluster-name> -n cpaas-system \
  -o jsonpath='{.metadata.annotations.cpaas\.io/network-type}{"\n"}'
# Expected output: kube-ovn
```

vSphere v1.0.16+

在 CAPI `Cluster` 上更新目标 chart 版本。提供商根据目标版本选择 chart 名称，并协调完整的 `AppRelease` 源代码和规范。

```
kubectl annotate cluster <cluster-name> -n cpaas-system \
  cpaas.io/kube-ovn-version=<kube-ovn-version-from-matrix> --overwrit
e
```

请勿手动修补 `AppRelease` 源代码。继续执行验证 Kube-OVN AppRelease。

vSphere v1.0.15 and earlier

> **WARNING**
>
> 此旧版流程仅在目标 Kube-OVN chart 早于 `v4.4` 时有效。对于 `v4.4` 或更高版本的 chart，请先将 vSphere provider 升级到 `v1.0.16` 或更高版本。旧版控制器会硬编码旧 chart 名称，并可能覆盖不受支持的手动源代码更改。

对于较早的目标 chart，仅更新 annotation。旧版提供商会协调完整的规范，但保留旧版 chart 名称。

```
kubectl annotate cluster <cluster-name> -n cpaas-system \
  cpaas.io/kube-ovn-version=<kube-ovn-version-from-matrix> --overwrit
e
```

Bare Metal

此操作步骤适用于 Bare Metal provider `v1.0.0` 及更高版本。没有可供选择的更早受支持版本流程。

验证基础设施集群使用 Kube-OVN，然后在 CAPI `Cluster` 上更新目标 chart 版本：

```
kubectl get baremetalcluster <cluster-name> -n cpaas-system \
  -o jsonpath='{.spec.networkType}{"\n"}'
# Expected output: kube-ovn

kubectl annotate cluster <cluster-name> -n cpaas-system \
  cpaas.io/kube-ovn-version=<kube-ovn-version-from-matrix> --overwrite
```

提供商根据目标版本选择 chart 名称，并协调完整的 `AppRelease` 源代码和规范。请勿手动修补 `AppRelease` 源代码。

验证 Kube-OVN AppRelease

针对正在升级的集群运行以下命令：

```
kubectl get apprelease cni-kube-ovn -n cpaas-system \
  -o jsonpath='Chart: {.spec.source.charts[0].name}{"\n"}Target: {.spec.source.charts[0].targetRevision}{"\n"}Installed: {.status.charts.*.installedRevision}{"\n"}Phase: {.status.charts.*.phase}{"\n"}'

kubectl get apprelease cni-kube-ovn -n cpaas-system \
  -o jsonpath='{range .status.conditions[*]}{.type}={.status} ({.reason}){"\n"}{end}'
```

在更改控制平面之前，验证以下所有内容：

- `spec.source.charts[0].name` 与上述 chart 名称边界匹配。
- `targetRevision` 和 `installedRevision` 均与 OS Support Matrix 行匹配。
- `phase` 为 `Success`。
- `Sync` 和 `Health` 条件均为 `True`。

正常的阶段顺序为 `Upgrading → HealthChecking → Success`。请勿仅根据 `installedRevision` 继续操作：它可能在 `HealthChecking` 期间发生变化，此时 Kube-OVN Pod 可能尚未验证为 Ready。如果 `AppRelease` 达到 `DownloadFailed`、`DeployFailed` 或 `NotReady`，请停止操作并检查其条件消息：

```
kubectl describe apprelease cni-kube-ovn -n cpaas-system
```

在阶段 1 恢复期间恢复 Kube-OVN

仅当目标 Kubernetes minor 版本上尚未创建任何控制平面 `Machine` 时，才能使用此恢复流程。如果目标 minor 版本的控制平面 machine 已加入，请保留目标 Kube-OVN 基线，并使用前向恢复或受支持的备份或 DR 恢复操作步骤。

在管理集群中的 CAPI `Cluster` 上恢复先前的 chart 版本：

```
kubectl annotate cluster <cluster-name> -n cpaas-system \
  cpaas.io/kube-ovn-version=<previous-kube-ovn-version> --overwrite
```

- **DCS** `v1.0.22+`、**HCS** `v1.0.4+`、**vSphere** `v1.0.16+` 和 **Bare Metal** `v1.0.0+`：恢复先前的 `cpaas.io/kube-ovn-version` annotation。提供商会恢复匹配的 chart 名称、修订版本和完整规范。请勿手动修补源代码。
- **DCS** `v1.0.21` 及更早版本或 **HCS** `v1.0.3` 及更早版本，且目标早于 **chart** `v4.4`：恢复 annotation 后，在正在恢复的集群上将现有 `targetRevision` 修补回先前的 chart 修订版本：

  ```
  kubectl patch apprelease cni-kube-ovn -n cpaas-system --type='json' \
    -p='[{"op":"replace","path":"/spec/source/charts/0/targetRevision","v
  alue":"<previous-kube-ovn-version>"}]'
  ```

- **vSphere** `v1.0.15` 及更早版本，且目标早于 **chart** `v4.4`：仅恢复先前的 annotation；提供商会协调完整的旧版规范。

恢复先前目标后，在更改任何控制平面清单之前，重复执行验证 Kube-OVN AppRelease。

Kubernetes 1.35 所需的 kubelet 补丁

当升级跳转的目标为 Kubernetes 1.35 或更高版本时，位于 `/etc/kubernetes/patches/kubeletconfiguration0+strategic.json` 的 kubelet 补丁文件必须包含以下设置：

```
{
  "imagePullCredentialsVerificationPolicy": "NeverVerify"
}
```

在 Kubernetes 1.35 中，即使镜像已存在于节点上，kubelet 也可能执行凭证验证。`NeverVerify` 告知 kubelet 不要验证这些预拉取镜像的拉取凭证。仅将此字段添加到将在 Kubernetes 1.35 或更高版本上运行的 kubelet 配置中；请勿为 Kubernetes 1.34 或更早版本添加此字段。

仅在跳转到 Kubernetes 1.35 或更高版本时应用此更改：

- 对于控制平面节点，在将 `spec.version` 更改为 1.35 的同一次编辑中，更新 `KubeadmControlPlane.spec.kubeadmConfigSpec.files` 中匹配的文件。目标版本仍为 1.34 时，请勿单独推出此策略。
- 对于工作节点，创建包含更新后文件的新 `KubeadmConfigTemplate`，然后在更新目标版本和基础设施模板的同一次编辑中更新 `MachineDeployment.spec.template.spec.bootstrap.configRef.name`。
- 如果文件使用 `contentFrom.secret`，请验证所引用的 Secret key 包含此参数。旧版本专用 Secret 不会自动适用于 1.35 跳转。

对于 Kubernetes 1.34 或更早版本，省略 `imagePullCredentialsVerificationPolicy`。仅在目标为 Kubernetes 1.35 或更高版本的升级跳转中添加它。

请参阅特定于平台的指南：

- Huawei DCS
- Huawei Cloud Stack
- VMware vSphere
- Bare Metal

准备跨版本升级

当目标 ACP Distribution Version 比集群当前版本高出多个 Kubernetes 小版本时，必须在开始阶段 2 之前完成额外准备。单个小版本的升级可跳过本节。

在 Alauda OS 上，每个 Kubernetes 小版本都内置于匹配的 OS 镜像中，并且每个 Kubernetes 小版本都对应 OS Support Matrix 中的特定 ACP 版本行。因此，Kubernetes 版本必须通过所需的中间 OS 镜像逐次前进一个小版本。请将这些 Kubernetes 和 OS 跳转与 ACP Core 升级路径分开处理：Distribution Version 必须遵循 CVO 提供的每个源版本到目标版本的跳转，而阶段 2 可能仅为推进 Kubernetes 和 OS 镜像而需要额外的中间版本行。

在开始阶段 2 之前，确定每个中间 Kubernetes 小版本及其匹配的 OS Support Matrix 行。根据升级前准备，单独确认阶段 1 已完成所有必需的 Distribution Version 跳转。

步骤 1 — 确认 Core 跳转并预先同步中间镜像

如果 CVO 将中间 Distribution Version 作为受支持 Core 路径的一部分提供，请在阶段 1 期间为该跳转完成完整的 ACP 工件准备以及 Core/CVO 升级。对于 ACP 4.0.x 到 4.4 等路径，这是必需的，且 `--only-sync-image` 无法替代此操作。在每次 Core 跳转中验证 `availableUpdates` 和 `VersionUpgradePath`。

完成所有必需的 Core 跳转后，某些中间 OS Support Matrix 行可能仍仅用于阶段 2 中 Kubernetes 和 OS 的顺序推出。对于这些仅涉及 Kubernetes/OS 的版本行，请下载相应的 Core 软件包，并仅将其中的镜像同步到 registry。请勿仅因某个版本行被用作 Kubernetes 或 OS 的过渡版本而请求额外的 Core 升级。

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

有关命令选项，请参阅同步升级工件。

此操作会将中间版本的 CoreDNS、etcd 和 Kube-OVN chart 镜像置于 registry 中，以便在将 `KubeadmControlPlane` 修补为匹配的 OS Support Matrix 行时，分阶段的 Kubernetes 推出可以拉取这些镜像。在 air-gapped 环境中，缺少中间镜像会导致新的控制平面组件无法启动。

对于必需的 Core 跳转，请按照阶段 1 的操作步骤准备并升级 Aligned 插件。对于仅涉及 Kubernetes/OS 的中间版本行，Aligned 插件无需额外的 `violet push`；该 Kubernetes 中间版本跳转使用的 Kube-OVN chart 已包含在上述 Core 镜像同步中。

步骤 2 — 根据置备机制准备中间 OS 镜像

对于每个中间 Kubernetes/OS 版本行，使该 ACP 版本发布的 Alauda OS 镜像可供节点置备机制使用，以便 Kubernetes 步骤可以依次使用每个镜像替换节点：

- **IaaS** 提供商（**Huawei DCS**、**VMware vSphere**、**Huawei Cloud Stack**）：上传 OS 镜像，并将其注册为 VM 模板（或 VM 镜像，具体取决于平台）。记录其注册名称 — machine template 会引用该名称。
- **Bare Metal**：使 OS 镜像可供集群使用的节点重新置备流程使用。

随后，控制平面和工作节点会依次使用中间 OS 镜像进行推出，并与暂存的 Kubernetes 小版本匹配，直到集群达到目标 ACP 版本行。

步骤 3 — 对每个中间 Kubernetes/OS 版本行应用特定于平台的操作步骤

依次对每个中间 Kubernetes/OS 版本行重复执行特定于平台的阶段 2 操作步骤，并使用 OS Support Matrix 中该版本行的 Kube-OVN、`KubeadmControlPlane` 和 `MachineDeployment` 值。对于每次跳转，先升级 Kube-OVN 并等待 `phase=Success`，然后升级控制平面，最后升级工作节点。在跳转到 Kubernetes 1.35 时，还要将必需的 kubelet 修补应用到控制平面和工作节点的引导配置。完成整个跳转后，再开始下一次跳转。所有中间跳转成功后，再针对目标版本行完整执行一次相同的操作步骤。

特定于平台的说明

选择您的平台，查看详细的升级说明：

裸金属

关键注意事项
升级顺序
provider 升级期间的存储兼容性
前提条件
检查 COS_STATE 可用空间
更新镜像目录
在升级控制平面前升级 Kube-OVN
升级控制平面
升级工作节点
跨版本升级
从失败的阶段 2 升级中恢复
验证
故障排查
其他资源

Huawei DCS

升级顺序
先决条件
使用 YAML
Fleet Essentials 边界
Additional Resources

华为云 Stacl

概览
升级顺序
前提条件
控制平面升级
工作节点升级
从失败的第 2 阶
其他资源

VMware vSphere

升级顺序
前提条件
OS Support Matrix 中的必需值
操作步骤
在不更改 Kubernetes 版本的情况下更新 O
不代表失败的滚动升级暂停
从失败的 Phase 2 升级中恢复
验证

将现有 DCS

概述
开始之前
检查当前磁盘布
确定哪些磁盘可
更新 `DCSMachi`

将现有 HCS 集群迁移到由池管理的持久磁盘

概述

开始之前

检查当前磁盘布局

决定要保留哪些磁盘

创建新的机器模板

准备滚动发布策略

更新机器配置池

触发滚动替换

验证迁移

故障处理

限制和恢复说明

相关主题

相关主题

- 在 Alauda OS 上创建集群
- 管理 Alauda OS 集群上的节点
- 基础设施资源

在裸金属上升级 Kubernetes

本指南介绍如何完成裸金属集群升级工作流的第 2 阶段。在升级 Kubernetes 之前，请先完成升级 Distribution Version中所述的 Distribution Version 升级。

INFO

本页面在完整 **ACP** 升级流程中的位置

本页面仅介绍升级中的 Kubernetes 步骤。完整的 ACP 升级流程（包括升级构件同步、通过 CVO 升级 ACP Core、升级 Aligned 插件，以及从 Marketplace 升级 Agnostic 插件）记录在以下页面中。开始本页面中的 Kubernetes 步骤之前，请先完成这些步骤：

- 升级概览（范围和顺序）
- 升级前准备
- 升级 `global` 集群的 Distribution Version（Core、Aligned、Agnostic）
- 升级业务集群的 Distribution Version（Core、Aligned、Agnostic）

当集群在裸金属主机上运行 Alauda OS 时，请使用本页面，因为裸金属上的 Kubernetes 步骤会使用新的 `elemental upgrade` 镜像替换每个节点，而不是在原位置升级二进制文件。

目录

关键注意事项

升级顺序

provider 升级期间的存储兼容性

前提条件

检查 COS_STATE 可用空间

删除升级失败后留下的部分快照

更新镜像目录

在升级控制平面前升级 Kube-OVN

升级控制平面

升级工作节点

跨版本升级

从失败的阶段 2 升级中恢复

验证

故障排查

其他资源

关键注意事项

裸金属升级会替换每个节点，不会在现有 OS 上运行 `kubeadm upgrade`。其机制如下：

1. CAPI 根据滚动发布策略删除一个 `Machine`。
2. provider 将 `clean` 计划写入 inventory 的 plan secret。
3. 主机停止 kubelet，停用所有受管数据挂载，清除 CRI 工作负载，停止 containerd，然后返回同一池中的 `Available`。
4. CAPI 创建替代的 `Machine`。
5. provider 从同一池中选择一个 `Available` inventory，根据 `elemental-image-catalog` 解析新的 Kubernetes 版本，并写入 `reprovision` 计划。
6. 主机准备受管卷激活，运行 `cloud-init clean` → `elemental upgrade --system <new-image>` → 重启。`initramfs` 会清除 Kubernetes 持久化状态；受管挂载单元会在 kubelet 之前启动，cloud-init 会重新执行，并针对新的控制平面运行 `kubeadm init` / `kubeadm join`。

开始之前，operator 必须牢记以下两个结构性影响：

- **Kubernetes** 管理的系统状态不会保留。`/var/lib/kubelet`、`/var/lib/containerd`、`/var/lib/etcd` 和 `/etc/kubernetes`

会在每次重新置备的 initramfs 清理步骤中被清除。Storage v2 不允许将这些受保护路径作为受管数据挂载点。

- 已声明的数据卷可以保留在 **inventory** 上。 清理计划会停用这些卷，但不会清除其文件系统；重新置备会在 kubelet 启动之前激活所选 inventory 中已准备好的卷。这是主机本地保留机制，不是备份或数据迁移机制。
- 同一个 `MachineInventory` 可能不会被再次选中。 provider 不保证由 `clean` 计划释放的 inventory 会被重新分配给替代的 `Machine`；数据不会自动跟随替代节点。由于替换操作会在 `maxSurge=0` 时按先删除后添加的方式执行（旧节点和新节点不重叠），因此池容量只需满足期望的副本数，发布过程中旧节点和新节点不会共存。

升级顺序

按以下顺序升级裸金属集群：

1. 前提条件：在 `global` 集群上升级 ACP 平台。这会将裸金属 provider、`elemental-operator` 以及相关 CAPI 组件升级到能够理解新模式的版本。仅在管理侧控制器完成推出并变为 Ready 后，才触发业务集群升级。
2. 在业务集群上升级 Distribution Version（Aligned Extensions）。请参阅升级 Distribution Version。
3. 将目标 Alauda OS 镜像对导入平台注册表，并将目标 Kubernetes 版本添加到 `elemental-image-catalog`。请参阅更新镜像目录。
4. 将 Kube-OVN 升级到目标 ACP 版本所需的 chart 版本，并等待 `AppRelease` 达到 `Success`。
5. 升级控制平面的 Kubernetes 版本（一次替换一个控制平面节点）。
6. 将 worker 节点升级到目标 Kubernetes 版本（在 `maxUnavailable` 预算内替换所有 worker 节点）。

Cluster API 负责编排滚动替换。

> **WARNING**
>
> 跳过步骤 1 可能导致两种故障：旧 provider 会静默忽略新的模式字段；或者控制器镜像在推出过程中发生替换，导致计划 secret 状态机中断。在开始业务集群推出之前，务必先完成管理侧升级。

provider 升级期间的存储兼容性

在引入受管理存储之前创建的现有集群仍保持兼容，只要 `MachineInventory.spec.storage` 保持缺失或设置为 `volumes: []`。在这种非受管理模式下，存储控制器不会执行磁盘操作，也不会仅仅因为管理侧 provider 已升级而重写运行中 Machine 的计划。

这种兼容性并不意味着可以在每台旧主机上原地启用受管理存储：

- 仅在 `global` 集群上升级 provider 和 `elemental-operator`，不会向旧版主机 OS 安装 `elemental-storage-observer.service`，也不会安装早期启动阶段的存储集成。
- 在此类主机上保持 `spec.storage` 为空。若要稍后启用受管理存储，请使用已批准且支持 observer 的镜像重新安装、重置或替换主机，然后在 inventory 未分配时，验证 observer 服务和新的 `status.observedStorage` 报告。
- 仅在满足上述主机镜像前提条件后，才配置并准备受管理卷。请勿向正在运行或已分配的 inventory 添加非空的存储声明。

有关准备和验证操作步骤，请参阅管理裸金属主机上的数据磁盘。

前提条件

开始之前，请确保满足以下所有前提条件：

- Distribution Version 升级已完成。
- 已安装的裸金属 provider `AppRelease` 报告的请求修订版本与已安装修订版本相同，且状态为 `phase=Success`。
- 可通过现有的 `<control-plane-vip>:<control-plane-port>` 访问控制平面。
- 所有当前节点均运行正常，且状态为 `Ready`。
- 在更改 Kube-OVN 或重新置备任何节点之前，请使用 ACP Cluster Enhancement 创建最新的按需 etcd 备份。确认最新的 `status.records` 条目报告状态为 `result: Success`，记录其 `backupTimestamp` 和 `fileName`，并确认相应的备份归档文件存在；即使任何控制平面节点被重新置备或变得不可用，仍能检索该文件。
- 目标 Kubernetes 版本是 `elemental-image-catalog` 中的一个键。如果不是，请在开始之前添加该版本（参阅更新镜像目录）。
- 已从 OS Support Matrix 中匹配的 ACP 行读取目标 **kube-ovn (chart)** 版本。

- `BaremetalCluster.spec.networkType` 为 `kube-ovn`；对于其他任何值，provider 都会跳过 Kube-OVN 协调。
- 集群中的每台主机都可以访问平台 registry。
- 推出过程中可能会被重新置备的每台主机，其 `state` 分区至少有 4 GiB 可用空间，且没有先前升级失败留下的部分快照（参阅检查 COS_STATE 可用空间）。
- `KubeadmControlPlane.spec.rolloutStrategy.rollingUpdate.maxSurge = 0` 和 `MachineDeployment.spec.strategy.rollingUpdate.maxSurge = 0` 均已设置；裸金属不会对物理主机进行超额配置。
- 相关池有足够的容量，能够一次替换一个节点，同时不低于所需的副本数。
- 每个存储声明非空的 inventory 均报告状态为 `StoragePrepared=True`，且每个当前已分配的 inventory 均报告状态为 `StorageActive=True`。在推出之前，记录每个 `BaremetalMachine.status.machineInventoryRef` 以及受管理文件系统 UUID，以便检测是否选择了不同的 inventory，而不是假定数据随 Machine 一同迁移。

> **INFO**
>
> 可使用 `/var/cpaas/backup` 下的默认本地归档进行恢复；不需要 S3 兼容存储。但是，裸金属替换可能导致原节点本地副本不可用，而且无法保证替换后的 `Machine` 使用相同的 `MachineInventory`。如果恢复设计无法保证在节点替换或故障后访问本地归档，请在开始第 2 阶段之前，将副本保存在节点之外的 S3 兼容存储或其他安全位置。

检查 COS_STATE 可用空间

每次重新置备都会在 `COS_STATE` 上创建新的快照，并将完整的目标系统写入其中。由于升级目标是*不同的*镜像，因此该分区必须同时容纳两个彼此无关、大小约为 3.5 GiB 的系统。空间不足的主机会在执行到一半时因 `no space left on device` 而失败，而失败的尝试会留下部分快照——因此每次自动重试开始时的可用空间都会比上一次更少。在 8 GiB 分区上重试三次就足以将其完全填满。

在开始推出之前检查这一点。检查受影响池中的每台主机，而不只是预计会被替换的主机：`clean` 计划释放的 inventory 不保证会再次被选中，因此任何 `Available` inventory 都可能接收新镜像。

在每台主机上：

```
# Free space on COS_STATE. Use btrfs, not df — df does not account for sn
apshot extents.
btrfs filesystem usage -h /

# Retained snapshots
snapper --no-dbus list
```

`Free (estimated)` 必须至少为 **4 GiB**——目标系统约为 3.5 GiB，此外还需要为复制期间写入的 btrfs 元数据预留空间。如果低于此值，请在开始推出之前释放空间。

删除升级失败后留下的部分快照

Elemental 在向快照写入数据时会将其标记为 `update-in-progress=yes`，并且仅在升级完成后清除该标记。仍带有此标记、且既不是默认快照也不是正在运行的快照的快照，就是失败尝试的残留，可以安全删除。

列出包含以下三个判定列的快照：

```
snapper --no-dbus --csvout list --columns number,default,active,userdata
```

仅当以下三个条件全部满足时，才删除快照——`default` 为 `no`，`active` 为 `no`，且 userdata 列包含 `update-in-progress=yes`：

```
snapper --no-dbus delete --sync <number>
```

> **WARNING**
>
> 绝不要删除 `default` 或 `active` 列为 `yes` 的快照。前者是主机下一次将要引导的快照，后者是当前正在运行的快照；删除其中任何一个都会导致主机无法引导。

随后重新运行 `btrfs filesystem usage -h /`，确认空间确实已回收。在已达到 100% 使用率的分区上，`snapper delete` 本身可能会因缺少元数据空间而失败——这正是最初导致部分快照累积的相同情况。如果删除失败，请将该主机视为需要重新安装，而不要针对它重试推出。

如果删除所有部分快照后，主机的可用空间仍少于 4 GiB，则分区本身过小。在应用 COS_STATE sizing 之前安装的主机使用 8 GiB 的 `COS_STATE`，而分区大小在安装时即已固定——扩大分区需要重新安装。请单独规划此操作；不要针对这类主机开始推出并期待其成功。

更新镜像目录

`elemental-image-catalog` 是用于向裸金属 provider 引入 Kubernetes 版本的资源。

> **INFO**
>
> 裸金属 provider 插件不包含任何 Kubernetes 版本或 OS 镜像。其 chart 创建的 `elemental-image-catalog` 不包含任何条目；升级插件或 Distribution Version 都不会添加条目。每个新的 Kubernetes 版本都必须显式添加，且需先将对应的 OS 镜像导入平台 registry。

首先，按照导入 OS 镜像并填充镜像目录中的说明，将目标版本的 `baremetal-base-image` 和 `baremetal-base-image-iso` 从 Bare Metal OS 归档导入平台 registry。然后使用以下任一选项添加版本。

选项 **A — 添加 chart override**。 在 `global` 集群上 provider 的 `ModuleInfo` 中，将新版本追加到 `spec.valuesOverride` 内的 `provider.imageCatalog.images` 下（使用 `global.registry.address` 作为 registry），与已在此处设置的值并列。不要替换整个 `valuesOverride`：这会删除 `caCertSecretKey`，导致 `elemental-operator` 无法启动。若要查找并编辑 `ModuleInfo`，请参阅安装集群插件：

```
spec:
  valuesOverride:
    acp/chart-cluster-api-provider-baremetal:
      provider:
        imageCatalog:
          images:
            <new-kubernetes-version>:
              repository: tkestack/baremetal-base-image
              tag: <official-image-tag>
```

平台会重新渲染 ConfigMap；provider 进程内的 watch 会热加载缓存，无需重启。在 DR 配对中，请在两个 `global` 集群上都执行此操作。

选项 **B — 直接修补 ConfigMap**。 如果安装插件时未使用 chart override，或者需要将已批准的镜像引用直接固定在目录中，请使用此选项：

```
kubectl -n cpaas-system patch configmap elemental-image-catalog \
  --type='json' \
  -p='[{"op":"add","path":"/data/<new-kubernetes-version>","value":"<regi
stry-address>/tkestack/baremetal-base-image:<new-tag>"}]'
```

无论使用哪种选项，继续之前都要进行验证：

```
kubectl -n cpaas-system get configmap elemental-image-catalog -o yaml
```

键必须包含前导 `v`（例如 `v1.34.5`）。如果 CAPI 创建替代的 `Machine` 时缺少该键，则生成的 `BaremetalMachine` 会进入 `Failed`，并带有 `ImageResolved=False / Reason=ImageCatalogMiss`，且不会写入 `reprovision` 计划。之后再添加该键不会重启它：`BaremetalMachine` 会一直处于 `Failed`，直到你删除该 `Machine`，并由其所有者创建新的实例。使用 `maxSurge: 0` 时，此时待替换的节点已经消失，因此必须在更改任何版本之前添加该键。

裸金属 provider 使用 `base-image` 进行重新置备，不会自动重建 SeedImages。在后续主机入驻之前，请使用从同一归档导入的 `base-image-iso` 刷新 `MachineRegistration` / `SeedImage`。不要混用来自不同归档的镜像，也不要替换为独立构建的镜像。

在升级控制平面前升级 Kube-OVN

按照在升级控制平面前升级 Kube-OVN中的裸金属流程操作。

共享操作步骤会验证 `BaremetalCluster.spec.networkType`，更新 CAPI `Cluster` 注释，让 provider 协调完整的 chart 源和规范，并检查 chart 名称、请求的修订版本、已安装的修订版本、阶段和条件。不要手动修补 `AppRelease` 源。

升级控制平面

继续之前，请完成共享的在升级控制平面前升级 Kube-OVN 操作步骤，并确认 `cni-kube-ovn` 已安装目标修订版本，且状态为 `phase=Success`。

将 `KubeadmControlPlane.spec.version` 修补为新的 Kubernetes 版本。如果组件镜像标签（DNS、etcd）已固定，请在同一次编辑中更新这些标签：

```
kubectl -n cpaas-system edit kubeadmcontrolplane <cluster-name>-control-plane
```

- `spec.version` ← 目标 Kubernetes 版本（必须与某个 `elemental-image-catalog` 键匹配）。
- `spec.kubeadmConfigSpec.clusterConfiguration.dns.imageTag` ← 与新版本匹配的 CoreDNS 镜像标签。
- `spec.kubeadmConfigSpec.clusterConfiguration.etcd.local.imageTag` ← 与新版本匹配的 etcd 镜像标签。
- 目标版本为 Kubernetes 1.35 或更高版本时，请按照适用于 Kubernetes 1.35 的必需 kubelet 补丁中的说明，在同一次编辑中更新 `spec.kubeadmConfigSpec.files` 中的 `/etc/kubernetes/patches/kubeletconfiguration0+strategic.json`。

仅升级 Kubernetes 时，裸金属 provider 不需要新的 `BaremetalMachineTemplate`：模板只包含池引用，升级镜像会通过目录从 `Machine.spec.version` 解析。只有需要将控制平面迁移到其他池时，才需要新模板。

Cluster API 会逐个滚动升级控制平面节点（因为 `maxSurge=0`）：

```
kubectl -n cpaas-system get kubeadmcontrolplane <cluster-name>-control-plane -w
kubectl -n cpaas-system get baremetalmachines.infrastructure.cluster.x-k8s.io
kubectl -n cpaas-system get machineinventorypools.infrastructure.cluster.x-k8s.io <cluster-name>-control-plane-pool
kubectl get nodes -o wide
# workload cluster
```

每个节点的预期操作顺序：

1. CAPI 将一个旧的 `Machine` 标记为待删除；对应的 `BaremetalMachine` 转为 `Preparing`。
2. provider 写入 `clean` 计划；托管卷会停用但不会擦除，最终清单状态为 `Available`，所有者注释已清除，存储已返回到 `Prepared`。
3. CAPI 创建新的 `Machine`（使用目标版本）；随后创建新的 `BaremetalMachine`。
4. provider 从控制平面池分配一个 `Available` 清单，解析新镜像，并写入 `reprovision` 计划。
5. 主机为所选清单中已准备好的卷暂存激活操作，运行 `elemental upgrade --system <new-image>`，重启，在 kubelet 启动前启动所需的挂载单元，并对存续的控制平面执行 `kubeadm join`。只有在所需存储已激活后，`BaremetalMachine.status.phase` 才会变为 `Running`。

重复步骤 1–5，直到所有控制平面节点都已替换。

在整个滚动升级期间，`alive` 会继续管理 VIP。随着控制平面成员发生变化，裸金属 provider 会重新渲染 `alive` chart 的值（`peer` 列表和 `ipvs.ips`），并滚动更新静态 Pod 清单。

升级工作节点

将每个 `MachineDeployment.spec.template.spec.version` 修补到新的 Kubernetes 版本。CAPI 会在 `maxUnavailable` 预算内替换工作节点——裸金属 provider 会复用与控制平面相同的工作节点池和镜像目录解析逻辑。

对于 Kubernetes 1.35 或更高版本，首先创建一个包含所需 kubelet 补丁的新工作节点 `KubeadmConfigTemplate`，然后在与版本升级相同的 `MachineDeployment` 编辑中更新 `spec.template.spec.bootstrap.configRef.name`。下面的命令包含该引用。对于不需要 bootstrap 更改的早期目标版本，请省略 `bootstrap` 对象。按照更新 Bootstrap 模板创建新模板。

```
kubectl -n cpaas-system patch machinedeployment <cluster-name>-workers \
  --type='merge' \
  -p='{"spec":{"template":{"spec":{"version":"<new-kubernetes-version
>","bootstrap":{"configRef":{"name":"<new-bootstrap-template>"}}}}}}'
```

查看：

```
kubectl -n cpaas-system get machinedeployments.cluster.x-k8s.io -w
kubectl -n cpaas-system get baremetalmachines.infrastructure.cluster.x-k8
s.io
kubectl get nodes -o wide
```

工作节点升级所需调整项少于控制平面：`MachineDeployment` 上没有 `clusterConfiguration` 块，因此无需更新 DNS / etcd 标签。工作节点上的 Kubernetes 组件版本会跟随新的基础镜像。

对于更早的版本，仅当发行版本更改了 cloud-init 或 `kubeletExtraArgs` 等设置时，才需要更改 bootstrap 模板。

跨版本升级

Kubernetes 控制平面必须经过受支持的次要版本跃迁。使用准备跨版本升级来识别并预先准备每个中间 ACP 行。

对于每个中间行以及最终的目标行：

1. 导入该行的 OS 镜像对，并将其 Kubernetes 版本添加到 `elemental-image-catalog`，如更新镜像目录中所述。
2. 将 Kube-OVN 升级到该行的 **kube-ovn (chart)** 版本，并等待 `phase=Success`。
3. 将控制平面升级到该行的 Kubernetes 版本，并等待每个控制平面节点变为 `Ready`。
4. 将每个工作节点 `MachineDeployment` 升级到相同的 Kubernetes 版本，并等待滚动升级完成后再开始下一次跃迁。

版本值始终来自 OS Support Matrix，而不是本指南中的固定示例。裸金属滚动升级策略已经将节点替换串行化，因此跨版本升级所需的额外池容量不会超过同一 minor 版本升级所需的容量。

从失败的阶段 2 升级中恢复

不要将 Kubernetes minor 版本降级视为普通回滚：

1. 如果尚未创建目标版本控制平面 `Machine`，请使用阶段 1 恢复期间恢复 Kube-OVN恢复 Kube-OVN，然后恢复之前的 `KubeadmControlPlane` 和 `MachineDeployment` manifest 值。
2. 如果目标 minor 版本的控制平面机器已加入，请停止进一步发布，并在目标 minor 版本上向前修复。如果向前修复无法恢复集群健康状态，请按照 ACP etcd 恢复操作步骤，从经过验证的升级前备份恢复集群。不要将 Kubernetes、CoreDNS 或 etcd 修补回之前的 minor 版本。

恢复 etcd 属于灾难恢复操作，而不是升级前验证步骤。不要仅为验证备份而在健康集群上运行恢复操作步骤。

每个完成 `reprovision` 计划的主机都已替换其 OS 镜像并清除 Kubernetes 管理的状态。更改版本字段不会重建之前的节点状态。

验证

发布完成后：

```
kubectl -n cpaas-system get baremetalmachines.infrastructure.cluster.x-k8
s.io \
  -l cluster.x-k8s.io/cluster-name=<cluster-name> \
  -o custom-columns='NAME:.metadata.name,PHASE:.status.phase,INVENTORY:.s
tatus.machineInventoryRef.name,PLAN_SECRET:.status.planSecretRef.name'
kubectl -n cpaas-system get machines.cluster.x-k8s.io \
  -l cluster.x-k8s.io/cluster-name=<cluster-name>
kubectl -n cpaas-system get apprelease cni-kube-ovn
kubectl get nodes -o wide
```

`cni-kube-ovn` `AppRelease` 应报告目标已安装修订版本，并显示 `phase=Success`。对于目标集群，每个当前的 `BaremetalMachine` 都应为 `Running`，每个 CAPI `Machine` 都应报告新的 `spec.version`，并且每个业务 `Node` 都应为 `Ready`，且使用新的 `kubelet` 版本。

对于每个当前的 `BaremetalMachine`，使用 `INVENTORY` 和 `PLAN_SECRET` 列验证支撑运行中节点的清单和计划：

```
kubectl -n cpaas-system get machineinventories.elemental.cattle.io <inven
tory-name> \
  -o jsonpath='{.status.plan.state}{" storage="}{.status.storage.phase}
{"\n"}'
kubectl -n cpaas-system get secret <plan-secret-name> \
  -o jsonpath='{.metadata.annotations.baremetal\.alauda\.io/plan\.type}
{"\n"}'
```

每个被引用的 `MachineInventory` 都应报告 `Applied`，并且每个被引用的计划 Secret 都应报告 `reprovision`。仅对当前 `BaremetalMachine` 对象所引用的清单执行此检查。未参与发布的备用清单可能没有 `reprovision` 计划。已完成 `clean` 计划但未被选中用于替换 Machine 的清单可能仍报告 `clean`；这两种情况都不表示发布未完成。

对于包含受管卷的清单，还应验证 `status.storage.phase=Active`、`StoragePrepared=True`、`StorageActive=True` 和 `BaremetalMachine.status.conditions[StorageReady]=True/AllVolumesReady`。将其当前文件系统 UUID 与升级前记录进行比较。不同的 UUID 通常表示选择了不同的清单；不得将其解释为原始数据已迁移。

`MachineInventoryPool.status` 应使用 `preparing = reprovisioning = 0` 满足 `available + allocated + preparing + reprovisioning + unavailable = total`。当池中包含备用清单时，`available` 可能仍大于零；这些清单不在上述计划验证范围内。

故障排查

问题	检查项
Kube-OVN 注解已更改，但 chart 名称、目标修订版本或已安装修订版本未收敛	检查 `AppRelease` 条件和 provide Kube-OVN 验证 操作；不要手动修
发布卡在第一个新节点上	`BaremetalMachine.status.cond` — 确认新密钥位于 `elemental-im` `Cluster` 携带 `cpaas.io/regist`
清理后清单未变为可分配	检查 `MachineInventory.status.` `status.observedStorage`。在清

问题	检查项
	前，必须根据主机的新观察结果确认 且已 Prepared。
替换 Machine 等待 `StorageReady`	验证所选清单是否为预期清单，所需 UUID 是否仍然匹配，所需挂载单元 主机是否生成了重启后的最新观察结
主机重启进入一个始终无法加入集群的状态	`MachineInventory.status.plan` 态）；检查主机的串行控制台以查找 错误。验证主机是否可以访问平台注
`elemental upgrade` 因 `no space left on device` 失败	`state` 分区太小 — 8 GiB 的 `COS` 在运行的系统和升级到不同的镜像。 一个部分快照，因此可用空间会持续 级留下的部分快照中的说明移除这些 以恢复。应用 COS_STATE 大小调 新安装才能扩展分区。
新控制平面节点的 `kubeadm join` 失败	确认 VIP 仍可访问；`alive` 可能已 中的 `kubectl -n kube-system g` `Ln`、`keepalived` 日志。
Worker 发布因设置了 `MachineDeployment.spec.strategy.rollingUpdate.maxSurge` `> 0` 而停滞	裸金属池无法实现超额配置。设置 `maxUnavailable`，以便在池中为
发布过程中池容量耗尽	`MachineInventoryPool.status.` `allocated` 仍包含尚未替换的节点 `MachineInventory` 以释放队列， `Preparing` 节点完成排空。

有关完整的 operator 侧状态机参考（包括每个条件原因和恢复操作），请参阅 Provider 概述。

其他资源

- Cluster API 升级文档 ↗

- Kubernetes 版本偏差策略 ↗

在 Huawei DCS 上升级 Kubernetes

本指南介绍如何完成 Huawei DCS 集群升级流程的第 2 阶段。在升级 Kubernetes 之前，请先完成升级 Distribution Version 中所述的 Distribution Version 升级。

INFO

本页面在完整 **ACP** 升级流程中的位置

本页面仅介绍 Kubernetes 升级步骤。完整的 ACP 升级流程（包括升级工件同步、通过 CVO 升级 ACP Core、升级 Aligned 插件，以及从 Marketplace 升级 Agnostic 插件）记录在以下页面中。在开始本页面所述的 Kubernetes 步骤之前，请先完成这些步骤：

- 升级概览（范围和顺序）
- 升级前准备
- 升级 `global` 集群的 Distribution Version（Core、Aligned、Agnostic）
- 升级业务集群的 Distribution Version（Core、Aligned、Agnostic）

当同一集群运行在 Alauda OS 上时，请使用本页面，因为此时 Kubernetes 步骤会从新的基于 Alauda OS 的 VM 模板替换节点，而不是在原节点上直接升级二进制文件。

INFO

Alauda OS 映像与 **DCS Provider** 配对

请选择为集群上安装的 ACP 版本发布的 Alauda OS 映像。请参阅下载 Alauda OS。DCS Provider 版本、Alauda OS 映像版本和 ACP 版本必须匹配。

如果现有集群的 VM 模板是使用早于 ACP `v4.3.2` 版本发布的 Alauda OS 映像创建的，则它们与 DCS Provider `v1.0.21` 或更高版本不匹配。升级 `global` 集群时，请同时升级 Provider 及其 Distribution Version，因为平台会在该升级过程中升级已安装的 Aligned 插件；请参阅升级前准备。

然后，使用为目标 ACP 版本发布的 Alauda OS 映像创建 VM 模板，并按照本页面中的步骤使用该模板替换节点。请参阅OS 支持矩阵中的必需值。

INFO

现有集群迁移

如果您的集群运行 ACP `v4.2.1` 或更高版本，并且仍在旧模板磁盘布局中保留数据，请先完成将现有 Huawei DCS 集群迁移到由存储池管理的持久磁盘中的迁移操作步骤，然后再依赖升级时的磁盘保留功能。

目录

- 升级顺序
- 先决条件
- 使用 YAML
  - OS Support Matrix 中的必需值
  - 在控制平面之前升级 Kube-OVN
  - 升级控制平面基础设施
    - 操作步骤
  - 升级控制平面 Kubernetes 版本
    - 操作步骤
  - 升级工作节点
    - 操作步骤
  - 从失败的第 2 阶段升级中恢复
- Fleet Essentials 边界
- Additional Resources

升级顺序

按照以下顺序升级 DCS 集群：

1. （先决条件） 首先升级管理集群上的 ACP 平台。这会将 cluster-api-provider-dcs 控制器及相关 CAPI 组件（core、KubeadmControlPlane provider、bootstrap provider）升级到能够理解新架构的版本。仅当管理端控制器完成发布并变为 Ready 后，才可触发业务集群升级。
2. 升级业务集群上的 Distribution Version（Aligned Extensions）。请参阅升级 Distribution Version。
3. 将 Kube-OVN 升级到目标 ACP 版本所需的 chart 版本，并等待 `AppRelease` 达到 `Success`。
4. 升级控制平面 Kubernetes 版本。
5. 将工作节点升级到目标 Kubernetes 版本。

Cluster API 使用内置安全机制编排滚动更新，以减少服务中断。

> **WARNING**
>
> 跳过步骤 1 可能导致两种故障：旧控制器会静默忽略写入 `DCSIpHostnamePool` / `DCSMachineTemplate` 的新架构字段；或者控制器映像在发布过程中切换，导致持久磁盘状态机进度中断。在开始业务集群滚动升级之前，请务必先完成管理端升级。

先决条件

开始之前，请确保满足以下所有先决条件：

- Distribution Version 升级已完成
- 控制平面可访问
- 所有节点均健康且处于 Ready 状态
- 已使用受支持的 ACP 备份操作步骤完成当前 etcd 备份，并验证备份有效
- IP Pool 具有足够的滚动更新容量
- VM 模板支持目标 Kubernetes 版本。有关版本映射，请参阅OS 支持矩阵

- 对于跨越多个 Kubernetes 次要版本的跨版本升级，已预先准备好中间版本的 Core 映像和 VM 模板。请参阅跨版本升级准备
- 目标 Kubernetes 版本与您的业务负载和附加组件兼容
- 如果使用存储池管理的持久磁盘，则 DCS VM 模板必须为 `4.2.1` 或更高版本，因为安全关机和磁盘分离依赖客户机工具
- 如果依赖存储池管理的持久磁盘，请保留 `KubeadmControlPlane.spec.rolloutStrategy.rollingUpdate.maxSurge = 0` 和每个 `MachineDeployment.spec.strategy.rollingUpdate.maxSurge = 0`
- 如果集群使用额外 NIC，请保留所需的 `DCSIpHostnamePool.spec.pool[].additionNic[]` 条目，并确认 DVS 和 Port Groups 可从可能接收替换 VM 的主机访问

WARNING

磁盘保留模型

升级依赖 Cluster API 的滚动更新机制。每个集群有四类磁盘；只有存储池管理的磁盘类会在删除并重新创建后保留。

磁盘类	声明位置
系统磁盘（根卷）	用于 `vmTemplateName` 的 VM 模板映像
模板本地磁盘	`DCSMachineTemplate.spec.template.spec.vmConfig.dcsMachineDiskSp`
存储池管理的持久磁盘	`DCSIpHostnamePool.spec.pool[].persistentDisk`
外部 CSI 卷（cinder 等）	业务负载 PVC / CSI 驱动

“保留”表示重新挂载同一磁盘标识，并不表示磁盘内容会回溯。在升级窗口期间写入存储池管理磁盘的任何内容，都会在升级后以及回滚后保留。

存储池管理磁盘的保留要求逐个替换，因此请在 `KubeadmControlPlane.spec.rolloutStrategy.rollingUpdate` 和 `MachineDeployment.spec.strategy.rollingUpdate` 上都保留 `maxSurge = 0`。

如果现有集群仍在旧模板磁盘布局中保留数据，请先按照将现有 Huawei DCS 集群迁移到由存储池管理的持久磁盘中的说明进行迁移。

INFO

滚动替换期间的额外 **NIC**

额外 NIC 声明在 `DCSIpHostnamePool.spec.pool[].additionNic[]` 中，而不是 `DCSMachineTemplate` 中。升级期间，每个替换 VM 都会使用其占用的 Pool 槽位中的额外 NIC。下面的升级模板步骤不应将额外 NIC 设置移入 `DCSMachineTemplate`。

INFO

每个升级步骤使用新的机器模板

本页面中的每个升级步骤都会创建一个新的 `DCSMachineTemplate` 及新的 `metadata.name`，应用该模板，然后将 `KubeadmControlPlane` 或 `MachineDeployment` 的 `infrastructureRef.name` 更新为新模板。请保留之前的模板，直到新的发布过程运行正常，以便在需要时回滚。

使用 YAML

基于 YAML 的升级不依赖 Fleet Essentials。

OS Support Matrix 中的必需值

ACP 版本、Kubernetes 版本以及匹配的 CoreDNS、etcd 和 Kube-OVN 版本之间的权威映射位于 OS Support Matrix 中。开始之前，请找到与目标 ACP 版本对应的行；该行提供了以下 YAML 步骤所需的组件值。

VM 模板名称不在矩阵中。它是根据目标 ACP 版本发布的 Alauda OS 镜像在 DCS 上创建的模板名称——请参阅下载 Alauda OS——并且必须将 `DCSMachineTemplate.spec.template.spec.vmTemplateName` 设置为该名称，同时应用于控制平面和工作节点模板。

从该行读取的单元格映射到升级清单的方式如下：

OS Support Matrix 列	用于设置
Kubernetes Version	`KubeadmControlPlane.spec.version` 和 `MachineDeployment.spec.template.s`
coredns	`KubeadmControlPlane.spec.kubeadmConfigSpec.clusterConfiguration.dns.im`
etcd	`KubeadmControlPlane.spec.kubeadmConfigSpec.clusterConfiguration.etcd.l`
kube-ovn (chart)	`Cluster.metadata.annotations["cpaas.io/kube-ovn-version"]` 和 provider 管 `kube-ovn` `AppRelease`

CoreDNS 和 etcd 的镜像标签仅适用于控制平面，因为 `clusterConfiguration` 是一个 `KubeadmControlPlane` 字段。工作节点从新的 VM 模板继承容器镜像版本；`MachineDeployment` 不携带自身的 `dns` / `etcd` 标签。Kube-OVN 注解位于 `Cluster` 资源上，而不是 `KubeadmControlPlane` 上，因为 DCS provider 会独立于 Kubernetes 控制平面 rollout 监视该资源。

请与集群的平台负责人确认，目标 ACP 版本发布的 Alauda OS 镜像已上传到 DCS 平台，并记录使用该镜像创建的 VM 模板名称。如果应用 `DCSMachineTemplate` 时 DCS 上不存在该 VM 模板，升级将失败。

在控制平面之前升级 **Kube-OVN**

请遵循在控制平面之前升级 Kube-OVN，并选择与已安装的 DCS provider 版本匹配的 Huawei DCS 操作步骤。该共享操作步骤负责 provider 版本边界、`v4.4` 处的 Kube-OVN chart 名称迁移、旧版行为以及必需的 `AppRelease` 健康检查。

对于 Kube-OVN `v4.4+` 目标，请勿使用旧版直接 `targetRevision` patch。请先将 DCS provider 升级到 `v1.0.22` 或更高版本，使其能够协调完整的 source 以及相关的 CoreDNS 和 kube-proxy 仓库变更。此 provider 升级不会将现有的外部控制平面端点迁移到 Self-built VIP；常规 Kubernetes 升级期间端点模式保持不变。

升级控制平面基础设施

升级控制平面机器模板可以推出更新的 VM 规格、系统补丁和基础设施设置。

操作步骤

1. 创建更新后的机器模板
   复制 `KubeadmControlPlane` 所引用的现有 `DCSMachineTemplate`，并将其另存为新文件：

   ```
   kubectl get dcsmachinetemplate <current-template-name> -n cpaas-system
   -o yaml > new-cp-template.yaml
   ```

2. 修改模板规格
   根据需要更新新模板：

   - 将 `metadata.name` 设置为 `<new-template-name>`
   - 更新 `spec.template.spec.vmTemplateName`
   - 更新 `spec.template.spec.vmConfig.dcsMachineCpuSpec.quantity`
   - 更新 `spec.template.spec.vmConfig.dcsMachineMemorySpec.quantity`
   - 仅针对系统磁盘和模板本地磁盘更新 `spec.template.spec.vmConfig.dcsMachineDiskSpec`
   - 从复制的清单中移除服务器生成的元数据（`resourceVersion`、`uid`、`generation`、`creationTimestamp`、`managedFields`、`kubectl.kubernetes.io/last-applied-configuration` 注解）以及整个 `status` 字段。

- 保持 `spec.template.spec.providerID` 未设置。DCS provider 会在 VM 创建后将 `providerID` 设置为 `dcs://<machine-name>`；在模板中预先填写该值会破坏控制器的身份绑定。

将由存储池管理的持久磁盘（包括 `/var/cpaas`）保留在 `DCSIpHostnamePool.spec.pool[].persistentDisk` 中。将额外 NIC 保留在 `DCSIpHostnamePool.spec.pool[].additionNic[]` 中。

3. 应用更新后的模板

```
kubectl apply -f new-cp-template.yaml -n cpaas-system
```

4. 更新控制平面引用
   修改 `KubeadmControlPlane` 资源，使其引用新模板：

```
kubectl patch kubeadmcontrolplane <kcp-name> -n cpaas-system --type='merge' -p='{"spec":{"machineTemplate":{"infrastructureRef":{"name":"<new-template-name>"}}}}'
```

5. 监控滚动更新

```
kubectl get kubeadmcontrolplane <kcp-name> -n cpaas-system -w
kubectl get machines -n cpaas-system -l cluster.x-k8s.io/control-plane
```

升级控制平面 **Kubernetes** 版本

在 Alauda OS 上升级控制平面 Kubernetes 版本采用删除并重新创建的工作流。控制平面 VM 会从指向目标 Alauda OS VM 模板的新 `DCSMachineTemplate` 开始逐个替换，并对 `KubeadmControlPlane` 资源进行 patch，使其携带匹配的 Kubernetes 版本、CoreDNS 镜像标签和 etcd 镜像标签。

开始之前，请完成共享的在控制平面之前升级 Kube-OVN 操作步骤，并确认 `cni-kube-ovn` `AppRelease` 已通过 `phase=Success` 位于目标 revision。然后按照 OS Support Matrix 中的必需值中的说明，从目标 ACP 行收集所有必需的控制平面值。

操作步骤

1. 为目标 **Kubernetes** 版本创建新的 `DCSMachineTemplate`
   复制现有控制平面模板，将 `metadata.name` 更新为新名称，并将 `spec.template.spec.vmTemplateName` 更新为在 DCS 上根据目标 ACP 版本发布的 Alauda OS 镜像创建的 VM 模板。将由存储池管理的持久磁盘保留在 `DCSIpHostnamePool.spec.pool[].persistentDisk` 中，将额外 NIC 保留在 `DCSIpHostnamePool.spec.pool[].additionNic[]` 中，而不是重新将它们作为模板字段添加。

   ```
   kubectl get dcsmachinetemplate <current-cp-template-name> -n cpaas-syst
   em -o yaml > new-cp-template.yaml
   # edit metadata.name and spec.template.spec.vmTemplateName
   kubectl apply -f new-cp-template.yaml -n cpaas-system
   ```

2. 使用目标 **Kubernetes** 值 **patch** `KubeadmControlPlane`
   在一次编辑中更新 `KubeadmControlPlane` 资源，使 `spec.version`、CoreDNS 镜像标签、etcd 镜像标签和基础设施模板引用与同一 Alauda OS 版本保持一致：

   - `spec.version` ← OS Support Matrix 行中的 **Kubernetes Version**
   - `spec.kubeadmConfigSpec.clusterConfiguration.dns.imageTag` ← 同一行中的 **coredns** 列
   - `spec.kubeadmConfigSpec.clusterConfiguration.etcd.local.imageTag` ← 同一行中的 **etcd** 列
   - `spec.machineTemplate.infrastructureRef.name` ← 第 1 步中创建的新 `DCSMachineTemplate` 名称
   - 当目标版本为 Kubernetes 1.35 或更高版本时，按照适用于 Kubernetes 1.35 的必需 kubelet patch中的说明，在同一次编辑中更新 `spec.kubeadmConfigSpec.files` 中的 `/etc/kubernetes/patches/kubeletconfiguration0+strategic.json`

     ```
     kubectl edit kubeadmcontrolplane <kcp-name> -n cpaas-system
     ```

   仅更新 `spec.version` 并不足够。CoreDNS 和 etcd 镜像标签必须与 Kubernetes 版本一起更新，因为它们基于同一 Alauda OS 版本构建；如果保留之前的值，可能会导致 CoreDNS 和 etcd pod 与新的 Kubernetes 次要版本不匹配。

3. 监控滚动更新

```
kubectl get kubeadmcontrolplane <kcp-name> -n cpaas-system -w
kubectl get machines -n cpaas-system -l cluster.x-k8s.io/control-plane
kubectl get nodes
```

当集群依赖由存储池管理的持久磁盘时，`KubeadmControlPlane.spec.rolloutStrategy.rollingUpdate.maxSurge` 必须保持为 `0`，以便控制平面 VM 逐个替换。

升级工作节点

工作节点 Kubernetes 升级通过 `MachineDeployment` 资源管理。工作节点升级所携带的字段少于控制平面：CoreDNS 和 etcd 镜像标签属于 `KubeadmControlPlane.spec.kubeadmConfigSpec.clusterConfiguration`，而 `MachineDeployment` 不具有这些标签。工作节点从新 VM 模板继承 Kubernetes 组件版本；`MachineDeployment` 只需包含目标 Kubernetes 版本和新模板引用。

开始之前，请从 OS Support Matrix 中目标 ACP 行读取 **Kubernetes Version** 单元格，并按照 OS Support Matrix 中的必需值中的说明，确认 DCS 上的目标 VM 模板名称。

操作步骤

1. 为工作节点创建新的 **`DCSMachineTemplate`**
   - 创建新的 `DCSMachineTemplate`，并将其 `vmTemplateName` 设置为在 DCS 上根据目标 ACP 版本发布的 Alauda OS 镜像创建的 VM 模板。有关导出、重命名和应用命令，请参阅更改机器基础设施
   - 将 `/var/cpaas` 以及任何其他在升级期间需要保留的磁盘保留在 `DCSIpHostnamePool.spec.pool[].persistentDisk` 中，而不是重新将它们作为模板磁盘添加
   - 将额外 NIC 保留在 `DCSIpHostnamePool.spec.pool[].additionNic[]` 中
2. 更新 **`MachineDeployment`**
   - 将 `spec.template.spec.version` 设置为同一 OS Support Matrix 行中的 **Kubernetes Version** 值
   - 将 `spec.template.spec.infrastructureRef.name` 设置为第 1 步中创建的新 `DCSMachineTemplate` 名称

- 当目标版本为 Kubernetes 1.35 或更高版本时，创建包含所需 kubelet patch 的新 `KubeadmConfigTemplate`（请参阅更新 Bootstrap 模板），并在同一次编辑中将 `spec.template.spec.bootstrap.configRef.name` 设置为该模板；对于更早的版本，仅当需要进行其他 bootstrap 更改时才更新 bootstrap 引用

3. 监控滚动更新

   - 确认滚动更新成功完成
   - 确认新工作节点已使用目标 Kubernetes 版本加入集群

   当集群依赖由存储池管理的持久磁盘时，`MachineDeployment.spec.strategy.rollingUpdate.maxSurge` 必须保持为 `0`，以便工作节点逐个替换。

从失败的第 2 阶段升级中恢复

不要将 Kubernetes 次要版本降级视为普通回滚。请根据 rollout 阶段选择恢复路径：

1. 尚未创建目标版本控制平面 `Machine`：恢复之前的 Kube-OVN 状态以及之前的 `KubeadmControlPlane` 和 `MachineDeployment` 清单值。这样可以在引入新的控制平面数据格式之前取消目标 rollout。
2. 仅机器模板或 **OS** 镜像发生了更改，**Kubernetes** 次要版本未发生更改：将控制资源重新指向之前的模板。Cluster API 会执行另一次替换 rollout。保持 Kubernetes 次要版本不变。
3. 目标 **Kubernetes** 次要版本的控制平面 `Machine` 已加入集群：不要将 Kubernetes、CoreDNS 或 etcd patch 回之前的次要版本。停止进一步 rollout，在目标次要版本上向前修复，或者使用受支持的 ACP 恢复操作步骤从已验证的升级前备份中恢复集群。

如果已创建目标次要版本的控制平面 `Machine` 但它从未加入集群，请先恢复健康的 etcd 仲裁，并确定是否可以安全地移除失败的替换。不要假定仅更改版本字段就足够。

在任何恢复期间，请牢记以下基础设施事实：

- 旧 **VM** 已不存在。 它们在升级期间已被销毁。模板恢复会构建一组全新的替换机器；它不会恢复原始 VM。
- 旧的 `DCSMachineTemplate` 资源必须仍然存在。 在新的 rollout 健康之前，不要删除之前的模板。如果已经删除，请在尝试同一次要版本模板恢复之前，根据版本控制或备份重新创建该模板。

- 存储池管理的磁盘身份会保留，但数据状态不会回溯。`DCSIpHostnamePool.spec.pool[].persistentDisk` 中声明的磁盘会在相同 IP 槽位重新连接到替换机器，但在升级窗口期间写入的数据仍保留在这些磁盘上。

对于阶段 1，请使用阶段 1 恢复期间恢复 Kube-OVN 中的 DCS 规则。恢复操作取决于已安装的 provider 版本：当前 provider 会从注解中恢复完整 source，而旧版 provider 需要使用之前 chart 的 revision patch。请等待恢复后的 `AppRelease` 通过共享验证，然后再更改控制平面清单。

`KubeadmControlPlane` 控制器会在 etcd 不健康时阻止替换。请先恢复仲裁，然后再重试任何安全的替换操作。

Fleet Essentials 边界

Fleet Essentials 1.0.4 及更高版本可以通过 CVO 请求升级 ACP 4.3 及更高版本的 Distribution Version。这是本指南的阶段 1。Fleet Essentials 不会执行阶段 2 中 DCS Kubernetes 和 Alauda OS 的替换，也不管理 `DCSIpHostnamePool.spec.pool[].persistentDisk`。请使用本页的 YAML 操作步骤完成阶段 2 rollout。

通过 Fleet Essentials 创建集群和日常管理节点池不受此升级限制。

Additional Resources

- 将现有 Huawei DCS 集群迁移到池管理的持久磁盘
- Cluster API 升级文档 ↗
- Kubernetes 版本偏差策略 ↗

在华为云 Stack 上升级 Kubernetes

本指南介绍如何在华为云 Stack 上升级 Kubernetes 集群，在尽量缩短停机时间的同时，保持系统稳定性和数据完整性。

INFO

本页在完整 **ACP** 升级流程中的位置

本页仅介绍升级中的 Kubernetes 步骤。完整的 ACP 升级流程包括升级制品同步、通过 CVO 升级 ACP Core、Aligned 插件升级，以及从 Marketplace 升级 Agnostic 插件，相关说明见以下页面。开始本页介绍的 Kubernetes 步骤之前，请先完成这些步骤：

- 升级概览（范围和顺序）
- 升级前准备
- 升级 `global` 集群的发行版（Core、Aligned、Agnostic）
- 升级业务集群的发行版（Core、Aligned、Agnostic）

如果同一集群运行在 Alauda OS 上，请参阅本页，因为此时 Kubernetes 步骤会使用新的 VM 模板替换节点，而不是原地升级二进制文件。

INFO

现有集群迁移

如果集群运行的是 ACP `v4.3.1` 或更高版本，并且仍在 `HCSMachineTemplate.spec.template.spec.dataVolumes[]` 中保留数据，请先按照将现有华为云 Stack 集群迁移到池管理持久磁盘中的操作步骤完成迁移，再依赖升级期间的磁盘保留功能。

目录

概览

升级顺序

前提条件

控制平面升级

基础设施映像更新

操作步骤

Kubernetes 版本升级

OS Support Matrix 中的必需值

在控制平面之前升级 Kube-OVN

升级控制平面 Kubernetes 版本

工作节点升级

升级工作节点 Kubernetes 版本

从失败的第 2 阶段升级中恢复

其他资源

概览

HCS 集群升级涉及多个组件，并遵循结构化方法以确保系统可靠性：

- 控制平面升级：更新 Kubernetes 控制平面组件和底层基础设施
- 工作节点升级：使用新的机器映像和 Kubernetes 版本升级工作节点
- 基础设施更新：修改虚拟机规格、存储和网络配置

升级顺序

按以下顺序升级 HCS 集群：

1. （前提条件） 首先升级管理集群上的 ACP 平台。这会将 `cluster-api-provider-hcs` 控制器和相关 CAPI 组件升级到能够识别新架构的版本。只有在管理侧控制器完成部署并进入 Ready 状态后，才能触发业务集群升级。
2. 升级业务集群的发行版。请参阅升级发行版。
3. 将 Kube-OVN 升级到目标 ACP 版本所需的 chart 版本，并等待 `AppRelease` 达到 `Success`。
4. 升级控制平面的 Kubernetes 版本。
5. 将工作节点升级到目标 Kubernetes 版本。

Cluster API 使用内置安全机制协调滚动更新，以减少服务中断。

> **WARNING**
>
> 跳过步骤 1 可能导致两种故障：旧控制器会静默忽略写入 `HCSMachineConfigPool` / `HCSMachineTemplate` 的新架构字段；或者在滚动升级期间切换控制器映像会中断持久磁盘状态机的推进。操作业务集群升级之前，务必先完成管理侧升级。

前提条件

开始之前，请确保满足以下所有前提条件：

- 已完成发行版升级。
- 控制平面可访问。
- 所有节点均运行正常，且处于 `Ready` 状态。
- 已创建当前的 etcd 备份，并使用受支持的 ACP 备份操作步骤完成验证。
- HCS 环境中已存在使用为目标 ACP 版本发布的 Alauda OS 映像创建的 VM 映像。应用新的 `HCSMachineTemplate` 时，如果该映像不存在，升级将失败。
- 对于跨越多个 Kubernetes 次要版本的跨版本升级，请预先暂存中间版本的 Core 映像和 VM 映像。请参阅跨版本升级准备。
- 目标 Kubernetes 版本与您的工作负载和附加组件兼容。
- 查看 Kubernetes 升级路径和版本偏差策略。

- 必须在 `HCSMachineConfigPool.spec.configs[].persistentDisks[]` 中声明所有需要在替换后保留的节点本地状态，而不是在 `HCSMachineTemplate.spec.template.spec.dataVolumes[]` 中声明。

初始部署请参阅创建集群指南。

> **WARNING**
>
> 单控制平面集群
>
> 本文档中的升级工作流适用于具有高可用控制平面的 HCS 集群。支持创建单控制平面 HCS 集群，但此类集群无法升级。

> **WARNING**
>
> 磁盘保留模型
>
> 升级依赖 Cluster API 的滚动替换机制。HCS 提供商有四种磁盘类别：

磁盘类别	声明位置	升级后留
系统磁盘（根卷）	`HCSMachineTemplate.spec.template.spec.rootVolume`	❌ 从
数据卷	`HCSMachineTemplate.spec.template.spec.dataVolumes`	❌ 从
池管理的持久磁盘	`HCSMachineConfigPool.spec.configs[].persistentDisks[]`	✅ 是
外部 CSI 卷（HCS EVS CSI 等）	工作负载 PVC / CSI 驱动程序	✅ 与 周期无

不要将 HCS `dataVolumes[]` 上的节点本地数据视为已保留状态。在滚动替换之前，请将 `/var/cpaas` 和其他需要保留的节点本地路径迁移到池管理的持久磁盘。

INFO

每个升级步骤都使用新的机器模板

本页上的每个升级步骤都会创建一个具有新 `metadata.name` 的 `HCSMachineTemplate`，应用该模板，然后将 `KubeadmControlPlane` 或 `MachineDeployment` 的 `infrastructureRef.name` 更新为新模板。请保留旧模板，直到新模板的滚动升级正常完成，以备回滚之需。

INFO

Fleet Essentials 适用范围

Fleet Essentials 1.0.4 及更高版本可以通过 CVO 请求 ACP 4.3 及更高版本的发行版升级，但不会执行本页所述的 HCS Kubernetes 和 Alauda OS 替换。请使用 ACP 工作流完成阶段 1，然后使用下方的 YAML 操作步骤完成阶段 2。

控制平面升级

控制平面升级会更新 Kubernetes API server、etcd、scheduler 和 controller manager，以及底层 VM 基础设施。

对于由使用 Pool 管理的持久磁盘的 `HCSMachineConfigPool` 支持的 HCS 控制平面，请在升级期间保留 `KubeadmControlPlane.spec.rolloutStrategy.rollingUpdate.maxSurge: 0`。持久磁盘绑定到固定的 `(hostname, slot)` 标识，因此滚动更新必须先移除旧机器，替换机器才能重新使用相同的磁盘。

基础设施映像更新

升级控制平面节点的底层机器映像可提供安全补丁、性能改进和更新后的系统组件。

操作步骤

1. 创建更新后的机器模板

   复制 `KubeadmControlPlane` 引用的现有 `HCSMachineTemplate`，并修改所需规格：

   ```
   kubectl get hcsmachinetemplate <current-template-name> -n cpaas-system
   -o yaml > new-cp-template.yaml
   ```

2. 修改模板规格

   修改新模板：

   - 将 `metadata.name` 设置为 `<new-template-name>`
   - 从复制的清单中移除服务器生成的元数据和状态字段。
   - 将运行时身份字段留空，包括 `spec.template.spec.providerID` 和 `spec.template.spec.serverId`。HCS provider 在创建实例时会分配这些值。
   - 不要将 `/var/cpaas` 等需要保留的路径放入 `spec.template.spec.dataVolumes[]`。在引用的 `HCSMachineConfigPool.spec.configs[].persistentDisks[]` 中声明这些路径。
   - 根据需要更新：
     - `spec.template.spec.imageName`
     - `spec.template.spec.flavorName`
     - `spec.template.spec.rootVolume.size`
     - 仅针对临时磁盘设置 `spec.template.spec.dataVolumes`

3. 部署更新后的模板

   应用新的机器模板：

   ```
   kubectl apply -f new-cp-template.yaml -n cpaas-system
   ```

4. 更新控制平面引用

   修改 `KubeadmControlPlane` 资源，使其引用新模板：

   ```
   kubectl patch kubeadmcontrolplane <kcp-name> -n cpaas-system --type='me
   rge' -p='{"spec":{"machineTemplate":{"infrastructureRef":{"name":"<new-
   template-name>"}}}}'
   ```

5. 监控滚动更新

控制平面将自动执行滚动更新：

```
kubectl get kubeadmcontrolplane <kcp-name> -n cpaas-system -w
kubectl get machines -n cpaas-system -l cluster.x-k8s.io/control-plane
```

Kubernetes 版本升级

升级 Kubernetes 版本需要同时更新控制平面软件和配套的虚拟机映像。

OS Support Matrix 中的必需值

ACP 版本、Kubernetes 版本以及匹配的 CoreDNS、etcd 和 Kube-OVN 版本之间的权威映射位于 OS Support Matrix 中。开始前，请找到与目标 ACP 版本对应的行；该行提供了以下操作步骤所需的组件值。

ECS 映像名称不在矩阵中。它是将为目标 ACP 版本发布的 Alauda OS 映像上传到 HCS 时使用的名称 — 请参阅下载 Alauda OS — 并且控制平面和工作节点模板中的 `HCSMachineTemplate.spec.template.spec.imageName` 都必须设置为该名称。

从该行读取的单元格与升级清单的对应关系如下：

OS Support Matrix 列	用于设置
Kubernetes Version	`KubeadmControlPlane.spec.version` 和 `MachineDeployment.spec.template.s`
coredns	`KubeadmControlPlane.spec.kubeadmConfigSpec.clusterConfiguration.dns.im`
etcd	`KubeadmControlPlane.spec.kubeadmConfigSpec.clusterConfiguration.etcd.l`
kube-ovn (chart)	`Cluster.metadata.annotations["cpaas.io/kube-ovn-version"]` 和 provider 管 `kube-ovn` `AppRelease`

OS Support Matrix 列	用于设置

CoreDNS 和 etcd 映像标签仅用于控制平面，因为 `clusterConfiguration` 是一个 `KubeadmControlPlane` 字段。工作节点从新的 VM 模板继承容器映像版本；`MachineDeployment` 不包含自身的 `dns` / `etcd` 标签。Kube-OVN 注释放在 `Cluster` 资源上，而不是 `KubeadmControlPlane` 上，因为 HCS provider 会独立于 Kubernetes 控制平面滚动更新监视该资源。

在控制平面之前升级 **Kube-OVN**

按照在控制平面之前升级 Kube-OVN 操作步骤进行，并选择与已安装的 HCS provider 版本匹配的 Huawei Cloud Stack 操作步骤。该共享操作步骤涵盖 provider 版本边界、`v4.4` 处的 Kube-OVN chart 名称迁移、旧版行为以及所需的 `AppRelease` 健康检查。

对于 Kube-OVN `v4.4+` 目标，请勿使用旧版直接 `targetRevision` 补丁。请先将 HCS provider 升级到 `v1.0.4` 或更高版本，以便它协调完整的源以及相关组件仓库变更。

升级控制平面 **Kubernetes** 版本

完成共享的在控制平面之前升级 Kube-OVN 操作步骤，并在继续之前，使用 `phase=Success` 验证 `cni-kube-ovn` `AppRelease` 已处于目标修订版本。

1. 为目标 **Kubernetes** 版本创建新的 **`HCSMachineTemplate`**
   复制现有控制平面模板，并将其应用到新的 `metadata.name` 下，同时设置目标 `imageName`：

   ```
   kubectl get hcsmachinetemplate <current-cp-template-name> -n cpaas-system -o yaml > new-cp-template.yaml
   ```

   在 `new-cp-template.yaml` 中：

   - 将 `metadata.name` 设置为 `<new-template-name>`。
   - 将 `spec.template.spec.imageName` 设置为上传到 HCS 的目标 Alauda OS 映像名称。

- 移除服务器生成的元数据（`resourceVersion`、`uid`、`generation`、`creationTimestamp`、`managedFields`、`kubectl.kubernetes.io/last-applied-configuration` 注释）以及整个 `status` 字段。
- 将运行时身份字段留空，包括 `spec.template.spec.providerID` 和 `spec.template.spec.serverId`。VM 创建后，HCS provider 会将 `providerID` 设置为 `hcs://<cluster-name>/<machine-name>`，并将 `serverId` 设置为 HCS ECS 实例 ID；在模板中预先填入这些值会破坏控制器的身份绑定。

```
kubectl apply -f new-cp-template.yaml -n cpaas-system
```

2. 使用目标 **Kubernetes** 值修补 `KubeadmControlPlane`
   在一次编辑中更新 `KubeadmControlPlane` 资源，使 `spec.version`、CoreDNS 映像标签、etcd 映像标签和基础设施模板引用与同一 Alauda OS 版本保持一致：

   - `spec.version` ← OS Support Matrix 行中的 **Kubernetes Version**
   - `spec.kubeadmConfigSpec.clusterConfiguration.dns.imageTag` ← 同一行中的 **coredns** 列
   - `spec.kubeadmConfigSpec.clusterConfiguration.etcd.local.imageTag` ← 同一行中的 **etcd** 列
   - `spec.machineTemplate.infrastructureRef.name` ← 步骤 1 中创建的新 `HCSMachineTemplate` 名称
   - 当目标为 Kubernetes 1.35 或更高版本时，按照Kubernetes 1.35 所需的 kubelet 补丁中的说明，在同一次编辑中更新 `spec.kubeadmConfigSpec.files` 中的 `/etc/kubernetes/patches/kubeletconfiguration0+strategic.json`

   ```
   kubectl edit kubeadmcontrolplane <kcp-name> -n cpaas-system
   ```

   仅更新 `spec.version` 是不够的。CoreDNS 和 etcd 映像标签必须随 Kubernetes 版本一同更新，因为它们基于同一 Alauda OS 版本构建；保留先前的值可能导致 CoreDNS 和 etcd pod 与新的 Kubernetes 次要版本不匹配。
   如果引用的控制平面 Pool 使用持久磁盘，请保留 `spec.rolloutStrategy.rollingUpdate.maxSurge: 0`。旧机器移除后，替换机器必须重新使用相同的固定主机名和磁盘槽位。

3. 验证升级进度

监控滚动升级过程：

```
# Check control plane status
kubectl get kubeadmcontrolplane <kcp-name> -n cpaas-system

# Monitor individual machines
kubectl get machines -n cpaas-system -l cluster.x-k8s.io/control-plane

# Verify cluster health
kubectl get nodes
```

工作节点升级

工作节点升级通过 `MachineDeployment` 资源进行管理。

升级工作节点 **Kubernetes** 版本

Kubernetes 版本升级要求同时协调更新 `MachineDeployment` 和底层 VM 模板。

注意：确保 VM 模板的 Kubernetes 版本与 `MachineDeployment` 中指定的版本匹配。版本不匹配将导致节点加入集群失败。

操作步骤：

1. 更新 **Machine** 模板
   使用支持目标 Kubernetes 版本的更新后的 `imageName`，创建新的 `HCSMachineTemplate`。有关导出、重命名和应用命令，请参阅更改机器基础设施。
2. 为 **Kubernetes 1.35** 或更高版本创建新的引导模板
   当目标为 Kubernetes 1.35 或更高版本时，在新名称下克隆当前的 `KubeadmConfigTemplate`，并将所需的 kubelet 补丁设置添加到 `/etc/kubernetes/patches/kubeletconfiguration0+strategic.json`。有关导出、编辑和应用步骤，请参阅更新引导模板。对于更早版本，除非该版本发布包含其他引导变更，否则跳过此步骤。
3. 更新 **MachineDeployment**
   修改以下字段：

- `spec.template.spec.version` - 目标 Kubernetes 版本
- `spec.template.spec.infrastructureRef.name` - 新机器模板名称
- `spec.template.spec.bootstrap.configRef.name` - 新引导模板名称；Kubernetes 1.35 或更高版本必需

```
kubectl patch machinedeployment <cluster-name>-md-0 -n cpaas-system \
  --type='merge' -p='{"spec":{"template":{"spec":{"version":"<kubernetes-version>","infrastructureRef":{"name":"<new-template>"},"bootstrap":{"configRef":{"name":"<new-bootstrap-template>"}}}}}}'
```

该命令包含 Kubernetes 1.35 或更高版本所需的引导引用。如果目标版本更早且没有引导变更，请省略 `bootstrap` 对象。

4. 监控升级
   验证新节点是否以正确的 Kubernetes 版本加入集群：

```
kubectl get nodes
```

从失败的第 2 阶段升级中恢复

不要将 Kubernetes 次要版本降级视为普通回滚。根据发布阶段选择恢复路径：

1. 尚未创建目标版本控制平面 `Machine`：恢复之前的 Kube-OVN 状态以及之前的 `KubeadmControlPlane` 和 `MachineDeployment` 清单值。这样可以在引入新的控制平面数据格式之前取消目标版本发布。
2. 仅机器模板或 **OS** 映像发生了更改，**Kubernetes** 次要版本未发生更改：将控制资源指回之前的模板。Cluster API 将执行另一次替换发布。保持 Kubernetes 次要版本不变。
3. 目标 **Kubernetes** 次要版本的控制平面 `Machine` 已加入集群：不要将 Kubernetes、CoreDNS 或 etcd 修补回之前的次要版本。停止进一步发布，在目标次要版本上继续修复，或使用受支持的 ACP 恢复操作步骤从经过验证的升级前备份恢复集群。

如果已创建目标次要版本的控制平面 `Machine` 但从未加入集群，请先恢复健康的 etcd 仲裁，并确定是否可以安全地移除失败的替换实例。不要假设仅更改版本字段就足够。

在任何恢复过程中，请牢记以下基础设施事实：

- 旧 **VM** 已不存在。 它们在升级期间已被销毁。模板恢复会构建一组全新的替换机器，而不会恢复原始 VM。
- 旧的 `HCSMachineTemplate` 资源必须仍然存在。 在新发布恢复健康之前，不要删除之前的模板。如果已删除，请在尝试同一次要版本模板恢复之前，从版本控制或备份中重新创建它。
- 只有由 **Pool** 管理的持久磁盘才能保留节点本地状态。 在升级窗口期间写入 `HCSMachineTemplate.spec.template.spec.dataVolumes[]` 的数据会在替换该 VM 时丢失。写入 `HCSMachineConfigPool.spec.configs[].persistentDisks[]` 中声明的磁盘的数据将被保留，并重新连接到替换 VM。应用程序数据仍应使用外部持久存储（例如 HCS EVS CSI），除非您的运维设计明确依赖节点本地状态。

对于第 1 阶段，请使用在第 1 阶段恢复期间恢复 Kube-OVN 中的 HCS 规则。恢复操作取决于已安装的 provider 版本：当前 provider 会从 annotation 中恢复完整源内容，而旧版 provider 需要应用之前 chart 修订版本的补丁。等待恢复的 `AppRelease` 通过共享验证后，再更改控制平面清单。

当 etcd 不健康时，`KubeadmControlPlane` 控制器可能会阻止替换。请先恢复仲裁，然后再重试任何安全的替换操作。

其他资源

- Cluster API 升级文档↗
- Kubernetes 版本偏差策略↗

在 VMware vSphere 上升级 Kubernetes

本文介绍如何在平台侧发行版升级完成后，升级 VMware vSphere 上的 Kubernetes 集群。本文档所述的工作流重点是通过 Cluster API 资源更新控制平面和工作节点。发布规则请参阅提供商要求。

INFO

本页在完整 **ACP** 升级流程中的位置

本页仅介绍 Kubernetes 升级步骤。完整的 ACP 升级流程包括升级制品同步、通过 CVO 升级 ACP Core、升级 Aligned 插件，以及从 Marketplace 升级 Agnostic 插件，相关说明见以下页面。请先完成这些步骤，再开始本页介绍的 Kubernetes 升级步骤：

- 升级概述（范围和顺序）
- 升级前准备
- 升级 `global` 集群的发行版版本（Core、Aligned、Agnostic）
- 升级业务集群的发行版版本（Core、Aligned、Agnostic）

如果同一集群运行在 Alauda OS 上，请使用本页，因为该环境中的 Kubernetes 升级步骤会通过新的 VM 模板替换节点，而不是原地升级二进制文件。

目录

升级顺序

前提条件

OS Support Matrix 中的必需值

操作步骤
根据准入规则检查清单
在控制平面之前升级 Kube-OVN
创建目标机器模板
升级控制平面
升级工作节点
在不更改 Kubernetes 版本的情况下更新 OS 映像
不代表失败的滚动升级暂停
从失败的 Phase 2 升级中恢复
验证
故障排查
后续步骤

升级顺序

按以下顺序升级 VMware vSphere 集群：

1. （前提条件） 首先升级管理集群上的 ACP 平台。这样可将 `cluster-api-provider-vsphere` 控制器及相关 CAPI 组件升级到能够识别新架构的版本。提供商会更新其 CRD，并自行将现有资源迁移到新的字段布局，无需手动修改 YAML。请参阅升级提供商。只有在管理侧控制器完成发布并变为 Ready 后，才触发业务集群升级。
2. 完成升级发行版版本中所述的发行版版本升级。
3. 确认现有的 `KubeadmControlPlane` 和 `MachineDeployment` 清单符合发布规则。不符合规则的 `KubeadmControlPlane` 可能会阻止升级时的首次编辑，不符合规则的 `MachineDeployment` 则会导致发布停滞。请参阅检查清单是否符合准入规则。
4. 确认控制平面运行正常，且当前集群稳定。
5. 将 Kube-OVN 升级到目标 ACP 版本所需的 chart 版本，并等待 `AppRelease` 达到 `Success`。
6. 升级控制平面的 Kubernetes 版本。
7. 将工作节点升级到目标 Kubernetes 版本。

前提条件

开始之前，请确保满足以下条件：

- 已完成发行版版本升级。
- 控制平面运行正常且可访问。
- 所有节点均处于 `Ready` 状态。
- 已按照受支持的 ACP 备份操作步骤创建并验证最新的 etcd 备份。
- vSphere 环境中已存在由面向目标 ACP 版本发布的 Alauda OS 镜像构建的 VM 模板。如果应用新的 `VSphereMachineTemplate` 时模板不存在，升级将失败。
- 对于跨越多个 Kubernetes 次要版本的跨版本升级，请预先暂存中间版本的 Core 镜像和 VM 模板。请参阅跨版本升级准备。
- 目标 Kubernetes 版本与工作负载及附加组件兼容。
- 机器配置池有足够的容量执行滚动更新。
- `KubeadmControlPlane` 将 `spec.rolloutStrategy.rollingUpdate.maxSurge: 0` 和 `spec.replicas` 设置为 `3` 或更高值，并且每个 `MachineDeployment` 都设置了 `spec.strategy.rollingUpdate.maxSurge: 0`。这些值可让替换操作复用相同的槽位和磁盘标识。在开始之前，请使用检查清单是否符合准入规则进行验证。
- 管理集群不是灾难恢复备用集群。针对备用集群时，提供商会跳过非 `global` 对象，并每 30 秒重新排队一次，因此发布看起来会像是尚未启动，而不是失败。请参阅备用管理集群。
- 你知道此集群的 `KubeadmControlPlane` 名称。请从 `Cluster.spec.controlPlaneRef.name` 中读取；在 VMware vSphere 上创建集群中的清单将其命名为 `<cluster_name>-kcp`。
- 如果集群使用自建 VIP（`controlPlaneLoadBalancer.type: internal`），则 `VSphereCluster.status.conditions[SelfBuiltLoadBalancerReady]` 为 `True`，并且新 VM 模板满足 Alive 内核前提条件（`ip_vs` 模块系列可加载，且允许使用 `net.ipv4.vs.conntrack=1`）。替换后的控制平面节点基于该模板构建，提供商不会注入或验证这些设置。
- 查看 Kubernetes 升级路径和版本偏差策略。

> **WARNING**
>
> 发布规则可能会拒绝对现有集群的首次编辑

`KubeadmControlPlane` 规则同样适用于 `UPDATE` 和 `CREATE`，因此现有集群可能包含不再被接受的值：

- `KubeadmControlPlane` 的 `rolloutStrategy.rollingUpdate.maxSurge` 未设置 — Cluster API 默认值为 `1`；
- `KubeadmControlPlane` 设置了 `replicas: 1`。

拒绝会发生在版本升级操作本身，因此升级会在任何内容被替换之前停止。请在单独的较早编辑中修正 `maxSurge`；请参阅检查清单是否符合准入规则。只有一个控制平面节点（`replicas: 1`）的集群无法升级。

未设置 `strategy.rollingUpdate.maxSurge: 0` 的 `MachineDeployment` 不会被拒绝，但其发布会停滞：当 `maxSurge` 默认值为 `1` 时，发布需要一个满负载池中没有的额外槽位。请在同一较早编辑中修正。

WARNING

磁盘保留模型

每个集群有五种磁盘类别；只有由池管理的持久类磁盘和外部 CSI 卷会在删除并重建后保留。

磁盘类别	声明位置	升留
系统磁盘（根卷）	用于 `spec.template.spec.template` 的 VM 模板	✗
模板本地磁盘	`VSphereMachineTemplate.spec.template.spec.*`（模板中声明的其他磁盘）	✗
池管理的临时磁盘	`VSphereMachineConfigPool.spec.configs[].ephemeralDisks[]`	✗ V V 并仓盘
池管理的持久磁盘	`VSphereMachineConfigPool.spec.configs[].persistentDisks[]`	✓ 离

磁盘类别	声明位置	升 留
		相 至
外部 CSI 卷（vSphere CSI 等）	工作负载 PVC / CSI 驱动程序	 原

与池管理的持久磁盘不同，临时磁盘不参与回收，也不会占用槽位释放，因此不会延迟发布。请参阅选择持久磁盘或临时磁盘。

磁盘所属类别在集群创建时即已确定。 `ephemeralDisks[]` 是 provider `v1.0.17` 中新增的。使用较早版本的 provider 创建的集群会将 `/var/cpaas`、`/var/lib/containerd` 和 `/var/lib/etcd` 声明为持久磁盘，并在 etcd 磁盘上设置 `wipeFilesystem: true`；升级不会改变这一点，因为 provider 不会迁移磁盘。升级时不要重新调整池的磁盘列表 — 标准的持久磁盘与临时磁盘拆分仅适用于新创建的集群。

“保留”表示重新连接的是同一磁盘标识 — 并不意味着磁盘内容会回到过去。升级期间写入池管理磁盘的任何内容都会在升级后保留，回滚后也会保留。

WARNING

每次更改都创建新模板

本页中的每个升级步骤都会创建一个具有新 `metadata.name` 的新 `VSphereMachineTemplate`，应用该模板，然后将控制资源的 `infrastructureRef.name` 更新为新模板。不要编辑现有模板。请保留旧模板，直到新发布运行正常，以备需要回滚。

INFO

Fleet Essentials 适用范围

Fleet Essentials 1.0.4 及更高版本可通过 CVO 请求 ACP 4.3 及更高版本的 Distribution Version 升级。它不会执行本页所述的 vSphere Kubernetes 和 Alauda OS 替换。请先使用升级发行版版本完成阶段 1，然后使用以下 YAML 操作步骤执行阶段 2。

WARNING

已升级的集群无法使用自建 **VIP**

自建 VIP 仅适用于在 ACP `v4.4` 或更高版本上使用 provider `v1.0.17` 或更高版本创建的集群。使用较早版本的 provider 创建的集群在 provider 升级后仍会保留其外部负载均衡器，且无法切换到自建 VIP：provider 会在第一个控制平面节点启动时注入 VIP，运行中的控制平面无法进行原地迁移。请参阅控制平面端点模式。

端点本身不会迁移，升级也不是更改端点的时机。控制平面端点 — 模式以及 VIP 或负载均衡器地址 — 在部署集群时选定，此后便固定不变；集群运行期间或升级过程中均无法调整。现有集群无需执行任何操作，也不会发生任何变化：`spec.controlPlaneLoadBalancer` 仍保持未设置，这正是外部负载均衡器的状态，而现在写入该字段会被拒绝。此字段为可选字段正是出于这个原因 — 在自建 VIP 出现之前创建的集群，其端点配置在升级过程中会保持不变。请参阅升级提供商。

OS Support Matrix 中的必需值

ACP 版本、Kubernetes 版本以及对应的 CoreDNS、etcd 和 Kube-OVN 版本之间的权威映射位于 OS Support Matrix。开始之前，请找到与目标 ACP 版本对应的行；该行提供了以下操作步骤所需的组件值。

VM 模板名称不在矩阵中。它是使用针对目标 ACP 版本发布的 Alauda OS 镜像在 vSphere 中创建模板时所用的名称 — 请参阅下载 Alauda OS — `VSphereMachineTemplate.spec.template.spec.template` 必须在控制平面和工作节点模板中都设为该名称。

该行中各列的值在升级清单中的对应关系如下：

OS Support Matrix 列	用于设置
Kubernetes Version	`KubeadmControlPlane.spec.version` 和 `MachineDeployment.spec.template.s`
coredns	`KubeadmControlPlane.spec.kubeadmConfigSpec.clusterConfiguration.dns.im`

OS Support Matrix 列	用于设置
etcd	`KubeadmControlPlane.spec.kubeadmConfigSpec.clusterConfiguration.etcd.l`
kube-ovn (chart)	`Cluster.metadata.annotations["cpaas.io/kube-ovn-version"]` 和由 provider `kube-ovn` `AppRelease`

CoreDNS 和 etcd 镜像标签仅用于控制平面，因为 `clusterConfiguration` 是 `KubeadmControlPlane` 字段。工作节点从新的 VM 模板继承容器镜像版本；`MachineDeployment` 不包含自己的 `dns` / `etcd` 标签。Kube-OVN 注解位于 `Cluster` 资源上，而不是 `KubeadmControlPlane` 上。

操作步骤

1 根据准入规则检查清单

更改任何内容之前，请先读取当前值：

```
kubectl -n <namespace> get kubeadmcontrolplane <cluster_name>-kcp \
  -o jsonpath='replicas={.spec.replicas} maxSurge={.spec.rolloutStrat
egy.rollingUpdate.maxSurge}{"\n"}'

kubectl -n <namespace> get machinedeployment -o custom-columns='NAM
E:.metadata.name,TYPE:.spec.strategy.type,SURGE:.spec.strategy.rollin
gUpdate.maxSurge,UNAVAIL:.spec.strategy.rollingUpdate.maxUnavailable'
```

`maxSurge` 值为空表示该字段未设置；未设置时将采用 Cluster API 默认值 `1`，而该值不适用于由池支持的控制器。`MachineDeployment` 示例使用 `maxUnavailable: 1`。请在提升版本之前更正这些值：

```
# replicas is 3 or more: correct maxSurge.
kubectl -n <namespace> patch kubeadmcontrolplane <cluster_name>-kcp -
-type='merge' \
  -p='{"spec":{"rolloutStrategy":{"type":"RollingUpdate","rollingUpda
te":{"maxSurge":0}}}}'

kubectl -n <namespace> patch machinedeployment <cluster_name>-md-0 --
type='merge' \
  -p='{"spec":{"strategy":{"type":"RollingUpdate","rollingUpdate":{"m
axSurge":0,"maxUnavailable":1}}}}'
```

如果 `replicas` 小于 3，则无法升级集群。

使用 `strategy.type: OnDelete` 的 `MachineDeployment` 无需设置 surge，因为 `OnDelete` 已采用先删除再替换机器的方式。

开始之前，请确认池运行正常：

```
kubectl -n <namespace> get vspheremachineconfigpool
kubectl -n <namespace> get vspheremachineconfigpool <cluster_name>-cp
-pool \
  -o jsonpath='{range .status.conditions[*]}{.type}={.status} ({.reas
on}){"\n"}{end}'
```

预期结果为 `Ready=True`、`MembersValid=True`、`MembersUnique=True` 和 `PersistentDisksReady=True`。在已完全分配的池中，`SlotAvailable` 合理地显示为

`False`；它不属于 `Ready` 摘要的一部分。

2 在控制平面之前升级 Kube-OVN

按照在控制平面之前升级 Kube-OVN 操作步骤进行，并选择与已安装的 provider 版本匹配的 VMware vSphere 选项卡：仍使用 `v1.0.15` 或更早版本的集群应遵循旧版选项卡；若目标为 Kube-OVN `v4.4` 或更高版本，则必须先将 provider 升级到 `v1.0.16` 或更高版本。通用操作步骤包含 vSphere 专用的网络注解检查、Kube-OVN `v4.4` 中的 chart 名称迁移，以及所需的 `AppRelease` 健康检查。请勿手动修补 chart 源；provider 会根据注解协调该配置。

3 创建目标机器模板

开始滚动升级之前，请为控制平面和工作节点创建新的 `VSphereMachineTemplate` 资源。

1. 导出现有控制平面模板

```
kubectl get vspheremachinetemplate <cluster_name>-control-plane -n
<namespace> -o yaml > new-cp-template.yaml
```

2. 修改控制平面模板

   编辑 `new-cp-template.yaml`：

   - 将 `metadata.name` 设置为新的唯一名称（例如 `<cluster_name>-control-plane-v2`）
   - 将 `spec.template.spec.template` 更新为目标 VM 模板名称
   - 如有需要，更新 CPU、内存或磁盘设置
   - 删除服务器生成的字段：`metadata.resourceVersion`、`metadata.uid`、`metadata.generation`、`metadata.creationTimestamp`、`metadata.managedFields`、`metadata.annotations["kubectl.kubernetes.io/last-applied-configuration"]` 和 `status`
   - 保持 `spec.template.spec.providerID` 未设置。VM 创建后，vSphere provider 会将 `providerID` 设置为该 VM 的 BIOS UUID；在模板中预先填入该值会破坏控制器的身份绑定。

3. 导出并修改工作节点模板

```
kubectl get vspheremachinetemplate <cluster_name>-worker -n <names
pace> -o yaml > new-worker-template.yaml
```

编辑 `new-worker-template.yaml`：

- 将 `metadata.name` 设置为新的唯一名称（例如 `<cluster_name>-worker-v2`）
- 将 `spec.template.spec.template` 更新为目标 VM 模板名称
- 如有需要，更新 CPU、内存或磁盘设置
- 删除上文列出的相同服务器生成字段

4. 应用两个新模板

```
kubectl apply -f new-cp-template.yaml
kubectl apply -f new-worker-template.yaml
```

4 升级控制平面

开始之前，请完成通用的在控制平面之前升级 Kube-OVN 操作步骤，并使用 `phase=Success` 验证 `cni-kube-ovn` `AppRelease` 已处于目标修订版本。然后，按照 OS Support Matrix 中的必需值所述，从 OS Support Matrix 中目标 ACP 对应的行收集所有必需的控制平面值。

1. 使用目标 **Kubernetes** 值修补 `KubeadmControlPlane`
   在一次编辑中更新 `KubeadmControlPlane` 资源，以确保 `spec.version`、CoreDNS 镜像标签、etcd 镜像标签和基础设施模板引用与同一个 VM 模板保持一致：

   - `spec.version` ← OS Support Matrix 行中的 **Kubernetes Version**
   - `spec.kubeadmConfigSpec.clusterConfiguration.dns.imageTag` ← 同一行中的 **coredns** 列
   - `spec.kubeadmConfigSpec.clusterConfiguration.etcd.local.imageTag` ← 同一行中的 **etcd** 列
   - `spec.machineTemplate.infrastructureRef.name` ← 上文创建的新 `VSphereMachineTemplate` 名称

- 如果目标为 Kubernetes 1.35 或更高版本，请按照 Kubernetes 1.35 所需的 kubelet 补丁所述，在本次编辑中更新 `spec.kubeadmConfigSpec.files` 中的 `/etc/kubernetes/patches/kubeletconfiguration0+strategic.json`

```
kubectl edit kubeadmcontrolplane <cluster_name>-kcp -n <namespac
e>
```

仅更新 `spec.version` 并不足够。CoreDNS 和 etcd 镜像标签必须与 Kubernetes 版本一同更新，因为它们基于同一版本构建；如果仍使用旧值，CoreDNS 和 etcd pod 的版本可能与新的 Kubernetes 次版本不匹配。

> **INFO**
>
> provider 完成协调后，预期会在刚才编辑的对象上看到 CoreDNS 和 kube-proxy 跳过注解，以及由控制器设置的 `dns.imageRepository`。请勿删除这些内容，也不要重新应用不包含这些内容的已存储清单。业务集群的 kube-proxy DaemonSet 和 CoreDNS Deployment 仓库会分别协调，并且仅在此滚动升级完成后进行协调，因此这些资源出现延迟是预期情况。请参阅清单中由 Provider 管理的字段。
>
> 在自建 VIP 集群上，Alive 会在整个滚动升级期间保留 VIP。引导 VIP 仅注入原始 `kubeadm init` 节点；替换后的控制平面节点通过 VIP 加入，且不会获得该 VIP。随着控制平面成员发生变化，provider 会根据当前控制平面节点地址更新 Alive 后端列表。请保持 `maxSurge: 0` — VIP 和 etcd 仲裁都依赖逐个替换节点。

2. 监控控制平面滚动升级

```
kubectl -n <namespace> get kubeadmcontrolplane <cluster_name>-kcp
-w
kubectl -n <namespace> get machine -l cluster.x-k8s.io/control-pla
ne
```

5 升级工作节点

控制平面升级完成后，更新 `MachineDeployment`，使其引用新的工作节点模板和目标 Kubernetes 版本。

常见更改包括：

- `spec.template.spec.version` — 目标 Kubernetes 版本
- `spec.template.spec.infrastructureRef.name` — 新的 `VSphereMachineTemplate` 名称
- `spec.template.spec.bootstrap.configRef.name` — 新的 `KubeadmConfigTemplate` 名称。Kubernetes 1.35 或更高版本必须进行此项设置，以便工作节点获得所需的 kubelet 补丁；对于更早版本，仅在需要更改其他引导设置时才进行修改。请参阅更新引导模板。

以下补丁仅更改 `spec.template.spec`。如果预检显示需要更正 `MachineDeployment` 策略，则必须在修补之前完成更正。

应用更改：

以下命令包含 Kubernetes 1.35 或更高版本所需的引导引用。如果目标版本更早且无需更改引导配置，请省略 `bootstrap` 对象。

```
kubectl patch machinedeployment <cluster_name>-md-0 -n <namespace> \
  --type='merge' -p='{
    "spec": {
      "template": {
        "spec": {
          "version": "<target_kubernetes_version>",
          "infrastructureRef": {
            "name": "<new-worker-template-name>"
          },
          "bootstrap": {
            "configRef": {
              "name": "<new-bootstrap-template-name>"
            }
          }
        }
      }
    }
  }'
```

监控工作节点滚动升级：

```
kubectl -n <namespace> get machinedeployment <cluster_name>-md-0 -w
kubectl -n <namespace> get machine
kubectl --kubeconfig=/tmp/<cluster_name>.kubeconfig get nodes -o wide
```

在不更改 Kubernetes 版本的情况下更新 OS 映像

使用此操作步骤，在 Kubernetes 版本保持不变的情况下，将节点迁移到新的 Alauda OS 映像，例如获取 OS 安全修复，或将Pod 日志目录迁移应用到现有节点。

1. 按照创建目标机器模板中的导出、清理和重命名操作步骤，创建新的 `VSphereMachineTemplate` 资源。仅将 `spec.template.spec.template` 更改为新的 VM 模板名称。
2. 应用两个新模板。
3. 修补 `KubeadmControlPlane.spec.machineTemplate.infrastructureRef.name`，等待控制平面滚动升级完成，然后修补 `MachineDeployment.spec.template.spec.infrastructureRef.name`。

不要修改 `spec.version`、`clusterConfiguration.dns.imageTag`、`clusterConfiguration.etcd.local.imageTag` 或 `cpaas.io/kube-ovn-version` 注释。OS 映像滚动升级并非 Kubernetes 版本升级，因此无需遵循先升级 Kube-OVN 的顺序。

同样的滚动升级规则仍然适用：两个控制器上的 `maxSurge: 0`。由池管理的持久磁盘会像版本升级时一样重新挂载；由池管理的临时磁盘则会被销毁并重新创建为空磁盘。

如需恢复，将控制资源指回之前的模板。这是从失败的第 2 阶段升级中恢复的第 2 阶段。

不代表失败的滚动升级暂停

某些滚动升级暂停是 Provider 为保护数据而设置的。不要手动清除这些暂停。

附件安全暂停。 删除 `VSphereVM` 时，Provider 会扫描数据中心，查找仍挂载着该槽位持久磁盘后端存储的 VM。只要仍有任何磁盘处于挂载状态，Provider 就会保留 finalizer 并重新排队。相同的检查也会保护池回收。如果滚动升级似乎在删除第一个旧机器时停滞，通常是触发了此暂停，而非出现故障。不要移除 **finalizer**。

读取当前状态：

```
kubectl -n <namespace> get vspheremachineconfigpool <pool_name> \
  -o jsonpath='{range .status.persistentDiskStatuses[*]}{.hostname}{" "}
{.name}{" "}{.phase}{" owner="}{.ownerMachineName}{" err="}{.lastError}
{"\n"}{end}'
kubectl -n <namespace> get vspherevm -o custom-columns='NAME:.metadata.na
me,DELETED:.metadata.deletionTimestamp,FINALIZERS:.metadata.finalizers'
```

正常的交接过程是先执行 `Attached`，待旧 VM 消失后执行 `Available`，然后在替换机器上再次执行 `Attached`。删除和重新创建过程中会保留 `ownerMachineName` 和 `ownerMachineUID`，因此替换机器会重新挂载相同的磁盘。

`Reclaimed` 是墓碑记录。 不要删除 `phase: Reclaimed` 条目来解除滚动升级停滞。有关删除该条目会导致重新播种循环的原因，请参阅集群创建页面中的故障排除。

`SlotAvailable=False` 属于预期情况。 使用 `maxSurge: 0` 且池已完全分配时，只有在删除旧机器后才会有空闲槽位。 `SlotAvailable` 会从池的 `Ready` 摘要中排除，正是为了避免将这种情况显示为失败。

在 **vCenter** 中关机的节点会保持关机状态。 `VSphereVM` 一旦锁定 `InitialPowerOnCompleted=True`，控制器就不再强制启动已关机的 VM。滚动升级期间创建的替换 VM，其锁定位未设置，因此升级前主动关机的节点会在替换后以开机状态恢复。

从失败的 **Phase 2** 升级中恢复

不要将 Kubernetes 次版本降级视为普通回滚。请根据滚动升级阶段选择恢复路径：

1. 尚未创建目标版本的控制平面 **`Machine`**：恢复之前的 Kube-OVN 注解，以及之前的 `KubeadmControlPlane` 和 `MachineDeployment` 清单值。这样可以在引入新的控制平面数据格式之前取消目标版本的滚动升级。
2. 仅更改了机器模板或 **OS** 镜像，**Kubernetes** 次版本未变：将控制资源指回之前的模板。Cluster API 会执行另一轮替换滚动升级。保持 Kubernetes 次版本不变。
3. 目标 **Kubernetes** 次版本的控制平面 **`Machine`** 已加入集群：不要将 Kubernetes、CoreDNS 或 etcd 回退到之前的次版本。停止进一步滚动升级，在目标次版本上向前修复，或使用受支持的 ACP 恢复操作步骤，从经过验证的升级前备份恢复集群。

如果已创建目标次版本的控制平面 `Machine`，但它从未加入集群，请先恢复健康的 etcd 仲裁，并确定是否可以安全移除失败的替换节点。不要假设仅更改版本字段就足够了。

恢复期间，请牢记以下基础设施事实：

- 旧 **VM** 已不存在。 它们已在升级过程中销毁。模板恢复会创建一组全新的替换机器，不会恢复原始 VM。
- 旧的 **`VSphereMachineTemplate`** 资源必须仍然存在。 在新一轮滚动升级恢复正常之前，不要删除之前的模板。如果已将其删除，请在尝试恢复相同次版本的模板之前，从版本控制或备份中重新创建。
- 池管理的磁盘身份会保留，但数据状态不会。 在 `VSphereMachineConfigPool.spec.configs[].persistentDisks[]` 中声明的磁盘会在相同槽位重新挂载到替换机器上，但升级期间写入的数据仍保留在这些磁盘上。
- 临时磁盘已丢失。 每个被替换 VM 上的 `VSphereMachineConfigPool.spec.configs[].ephemeralDisks[]` 数据都会随 VM 一起销毁，无法从池中恢复。只有池管理的持久磁盘会重新挂载。
- 不要撤销滚动升级规则的修正。 如果阶段 1 要求恢复之前的清单值，请将 `maxSurge` 保持为 `0`。

对于阶段 1，请使用在阶段 1 恢复期间恢复 Kube-OVN 中的 vSphere 规则。Provider 会根据注解恢复 chart。请等待恢复后的 `AppRelease` 通过共享验证，然后再更改控制平面清单。

如果 etcd 不健康，`KubeadmControlPlane` 控制器可能会阻止替换操作。重试任何安全的替换操作之前，请先恢复仲裁。

验证

升级后重新读取池状态：

```
kubectl -n <namespace> get vspheremachineconfigpool
kubectl -n <namespace> get vspheremachineconfigpool <pool_name> \
  -o jsonpath='{range .status.persistentDiskStatuses[*]}{.hostname}{" "}
{.name}{" "}{.phase}{"\n"}{end}'
```

确认以下结果：

- `KubeadmControlPlane` 达到目标版本和所需的从节点数。
- `MachineDeployment` 达到目标版本和所需的从节点数。
- 控制平面和工作节点恢复到 `Ready` 状态。
- vSphere CPI daemonset 在业务集群中仍然可用。
- 每个 `VSphereMachineConfigPool` 均报告 `Ready=True` 和 `PersistentDisksReady=True`；其 `ALLOCATED` 数量等于从该池调度运行的 Machine 数量，并且其 `TOTAL` 数量保持不变。
- 已分配槽位上的每个持久磁盘均报告 `phase: Attached`。

故障排查

问题	检查内容
`KubeadmControlPlane` 上的 `kubectl edit` 或 `kubectl patch` 被拒绝	运行升级前检查。有关完整的消息与原因对照表，请参阅准入拒绝。
删除第一台机器后，滚动升级停滞	通常是附件安全保持机制，而非缺陷。请参阅不会导致失败的滚动升级保持机制。
池报告 `Ready=True`，但 `SlotAvailable=False`	在 `maxSurge: 0` 滚动升级期间，已完全分配的池出现这种情况是正常的。`SlotAvailable` 不计入 `Ready` 汇总。
被替换节点的磁盘为空	确认磁盘声明在哪个列表中。对于在 `v1.0.17` 或更高版本上创建的集群，`/var/lib/kubelet`、`/var/lib/containerd` 和 `/var/lib/etcd` 按设计属于临时磁盘，每次替换时都会从空状态开始。对于较旧的集群，它们属于持久磁盘，因此还要检查 `wipeFilesystem`；按设计，它在 etcd 磁盘上为 `true`。
没有任何资源协调、条件没有变化，也没有出现错误	管理集群可能是灾难恢复备用集群。请参阅备用管理集群。
业务集群节点已关闭且无法恢复	一旦 `InitialPowerOnCompleted` 已锁存，这是预期行为。请在 vCenter 中将其开机。请参阅不会导致失败的滚动升级保持机制。

问题	检查内容
新的控制平面机器始终无法引导	查看 `VSphereVM` 和 `VSphereMachine` 上的 `BootstrapReady` 条件；原因分别为 `BootstrapSecretGetFailed` 和 `BootstrapSecretContentInvalid`。
Kube-OVN 注解已更改，但 chart 名称或修订版本未收敛	按照共享的 Kube-OVN 操作步骤进行操作。请勿手动修补 chart 源。
控制平面达到目标版本后，业务 `kube-proxy` 或 `coredns` 仍在使用旧镜像仓库	这是预期的堆积量——只有在控制平面滚动升级完成后，才会对这些资源进行协调。请参阅清单中由 Provider 管理的字段。
`VSphereMachineTemplate` 更新因不可变而被拒绝	使用新名称创建新的模板。请参阅本页上方的警告。

后续步骤

Kubernetes 升级完成后，请继续参阅 在 VMware vSphere 上管理节点，执行常规节点操作。

将现有 Huawei DCS 集群迁移到池管理的持久磁盘

当您将现有 Huawei DCS 集群从旧版模板磁盘布局升级到当前的池管理持久磁盘模型时，请使用本指南。

此迁移由 YAML 驱动，因为 `DCSIpHostnamePool.spec.pool[].persistentDisk` 未在 Web UI 中公开。

INFO

版本

当集群运行 ACP `v4.2.1` 或更高版本时，请使用此操作步骤。

此操作步骤目前假设满足以下所有条件：

- DCS VM 模板为 `4.2.1` 或更高版本。
- Guest tools（`vmtools`）在 guest OS 中正常工作，以便完成安全关机和磁盘分离。

目录

概述

开始之前

检查当前磁盘布局

确定哪些磁盘可声明

示例

更新 `DCSMachineTemplate`

更新 `DCSIpHostnamePool`

触发滚动升级

验证声明、分离、转换和重新连接

限制和恢复说明

相关主题

概述

较早创建的 DCS 集群通过 `DCSMachineTemplate` 创建可重复使用的数据磁盘。该布局无法为控制器提供足够的信息，以便在删除并重新创建实例时安全地保留磁盘。

当前模型会将升级期间保留的磁盘移入 `DCSIpHostnamePool.spec.pool[].persistentDisk`。每个磁盘都绑定到一个 `(ip, slot)` 身份。在滚动替换期间，控制器会执行以下操作：

1. 从旧 VM 声明现有磁盘。
2. 安全停止旧 VM。
3. 分离磁盘。
4. 在需要时将 stock 卷转换为独立持久卷。
5. 删除旧 VM。
6. 将磁盘重新连接到替换 VM。
7. 启动替换 VM；替换 VM 会挂载现有文件系统，而不会重新格式化该文件系统。

这也是平台要求的 `/var/cpaas` 磁盘所采用的文档化模型。

开始之前

开始前，请确认以下所有事项：

- 集群运行正常且当前保持稳定。
- 由于池管理的持久磁盘需要逐个替换，因此相关控制平面和工作节点的发布策略使用 `maxSurge: 0`。
- 您可以从 DCS UI 中识别旧 VM 上当前的磁盘 `sequenceNum` 值，或通过 DCS API 查询 VM 详细信息。
- 您知道哪些磁盘必须保留，以及哪些磁盘仍可随 VM 一起重新创建。
- 目标 `DCSIpHostnamePool` 已存在，并将每个节点映射到固定的 IP 槽位。

检查当前磁盘布局

首先，确定管理集群对象以及为每个节点提供支持的 DCS VM：

```
kubectl get kubeadmcontrolplane -n cpaas-system
kubectl get machinedeployment -n cpaas-system
kubectl get machine -n cpaas-system
kubectl get dcsmachine -n cpaas-system
kubectl get dcsiphostnamepool -n cpaas-system
```

对于计划迁移的任何 `DCSMachine`，请检查当前 VM 详细信息，并记录每个要保留磁盘的磁盘 `sequenceNum`、大小、数据存储和 PCI 类型。

您可以通过以下方式收集这些信息：

- DCS 平台 UI。
- 封装 `QueryVmInfo` 的现有运维工具。
- 如果您的环境已提供相应工作流，也可以直接检查 API。

每个要保留的磁盘都需要以下值：

- 旧 `sequenceNum`
- `quantityGB`
- `datastoreName` 或 `datastoreClusterName`
- `path`
- `format`

- `pciType`

确定哪些磁盘可声明

现有集群只能声明位于旧 VM 磁盘布局尾部连续区域中的磁盘。

使用以下公式：

```
slot = oldSequenceNum - systemDiskCount - newTemplateDataDiskCount - 1
```

应用公式时使用以下常量：

- `systemDiskCount = 1`
- `newTemplateDataDiskCount` = 新 `DCSMachineTemplate` 中保留的非系统磁盘数量

计算得出的 `slot` 必须：

- 大于或等于 `0`
- 在同一 IP 条目中唯一

如果磁盘不在尾部连续区域中，则必须采取以下措施之一：

- 将它与旧模板末尾之间的磁盘也移入池管理的持久磁盘列表，或
- 接受该不可声明的磁盘仍会随旧 VM 一起丢失

示例

假设旧模板中的磁盘顺序如下：

旧序号	旧磁盘
1	系统磁盘
2	`/var/lib/kubelet`
3	`/var/lib/etcd`

旧序号	旧磁盘
4	`/var/lib/containerd`
5	`/var/cpaas`

如果新模板只保留 `system + /var/lib/kubelet + /var/lib/containerd`，那么 `newTemplateDataDiskCount = 2`。

要保留的磁盘	旧 `sequenceNum`	新模板数据磁盘数量	计算得出的 `slot`	可声明
`/var/cpaas`	5	2	`5 - 1 - 2 - 1 = 1`	是
`/var/lib/containerd` 和 `/var/cpaas`	4, 5	1	`4 - 1 - 1 - 1 = 1`、`5 - 1 - 1 - 1 = 2`	是
仅 `/var/lib/etcd`	3	2	`3 - 1 - 2 - 1 = -1`	否

更新 `DCSMachineTemplate`

就地编辑当前引用的 `DCSMachineTemplate`，使其不再声明要保留的磁盘。

1. 导出当前模板：

```
kubectl get dcsmachinetemplate <template-name> -n cpaas-system -o yaml
> current-template.yaml
```

2. 更新导出的清单：

- 保留系统磁盘。
- 仅保留仍应随 VM 一起重新创建的模板本地磁盘。
- 移除要通过 IP 池保留的所有磁盘。

- 如果只有同时移动末尾磁盘才能声明目标磁盘，也要从模板中移除这些末尾磁盘。
- 保留原始 `metadata.name`，因为此迁移会就地更新当前引用的模板。
- 移除 `resourceVersion`、`uid`、`creationTimestamp` 和 `managedFields` 等临时元数据字段。

3. 应用更新后的模板：

```
kubectl apply -f current-template.yaml -n cpaas-system
```

更新 `DCSIpHostnamePool`

为每个保留的磁盘，在匹配的 IP 槽位中添加 `persistentDisk` 条目。

此规范会通过以下三种方式与实时磁盘属性交互：

严格声明匹配。其中任一字段不匹配都会导致声明失败，并设置带有 `lastError` 的 `phase=Error`。控制器会以较慢的循环持续重试，直到规范得到更正：

- `quantityGB` — 必须与实时磁盘大小完全匹配
- `datastoreName` 或 `datastoreClusterName` — 必须指向与实时磁盘相同的存储目标
- `pciType` — 必须与实时磁盘的 PCI 类型匹配。如果省略，provider 将使用默认值 `VIRTIO`；省略此字段前，请先验证实时磁盘的 PCI 类型，因为非 `VIRTIO` 的实时磁盘可能无法通过严格声明匹配

`isThin` 仅用于创建。只有在 provider 创建新的 DCS 持久卷时才会发送此字段。现有卷声明期间不会比较此字段，也不会转换现有卷。

文件系统（影响来宾端初始化，不影响声明检查）：

- `format` — 仅在初始化新磁盘时使用。如果实时磁盘已有文件系统，则保留现有格式，并跳过 `mkfs`。

来宾端（仅应用于替换 **VM**，不属于声明检查的一部分）：

- `path` — 来宾内部的挂载路径
- `mountOptions` — 挂载选项

- `options` — 仅在首次格式化时应用的 `mkfs` 选项

对于平台要求的 `/var/cpaas` 磁盘，请在此迁移过程中将其移动到由池管理的布局中。

将 `slot` 设置为上一节中计算出的值。对于不同的磁盘布局，不要重复使用固定的示例值。

示例：

```
apiVersion: infrastructure.cluster.x-k8s.io/v1beta1
kind: DCSIpHostnamePool
metadata:
  name: <iphostname-pool-name>
  namespace: cpaas-system
spec:
  pool:
  - ip: "<node-ip>"
    mask: "<mask>"
    gateway: "<gateway>"
    dns: "<dns>"
    hostname: "<hostname>"
    machineName: "<machine-name>"
    persistentDisk:
    - slot: <calculated-slot>
      quantityGB: 40
      datastoreClusterName: <datastore-cluster-name>
      path: /var/cpaas
      format: xfs
      pciType: VIRTIO
```

应用池更新：

```
kubectl apply -f <updated-pool-file>.yaml -n cpaas-system
```

触发滚动升级

触发替换之前：

- 确认 `KubeadmControlPlane.spec.rolloutStrategy.rollingUpdate.maxSurge = 0`

- 确认每个 `MachineDeployment.spec.strategy.rollingUpdate.maxSurge = 0`

这些设置是执行迁移以及在后续升级期间重新使用由池管理的持久磁盘的前提条件。

然后触发发布：

```
kubectl patch kubeadmcontrolplane <kcp-name> -n cpaas-system \
  --type='merge' \
  -p='{"spec": {"rolloutAfter": "'"$(date -u +%Y-%m-%dT%H:%M:%SZ)"'"}}'
```

```
kubectl patch machinedeployment <md-name> -n cpaas-system \
  --type='merge' \
  -p='{"spec": {"rolloutAfter": "'"$(date -u +%Y-%m-%dT%H:%M:%SZ)"'"}}'
```

验证声明、分离、转换和重新连接

在发布期间监控管理集群资源：

```
kubectl get kubeadmcontrolplane <kcp-name> -n cpaas-system -w
kubectl get machinedeployment <md-name> -n cpaas-system -w
kubectl get machine -n cpaas-system -w
```

检查池状态，确认控制器已声明并跟踪这些磁盘：

```
kubectl get dcsiphostnamepool <iphostname-pool-name> -n cpaas-system -o y
aml
```

在转换期间，每条记录都会显示在 `status.persistentDiskStatus` 下。需要关注的稳定阶段包括：

- 旧 VM 仍拥有磁盘时为 `phase: Attached`
- 磁盘分离后为 `phase: Available`（如果需要，还会从 stock 卷转换为独立持久卷）
- 替换 VM 重新连接磁盘后再次为 `phase: Attached`

在相应操作期间，可能会短暂出现临时阶段（`Attaching`、`Detaching`）；永久移除磁盘时会出现 `Deleting`，例如清理池或集群期间。完整的阶段集合为 `Creating`、`Available`、`Attaching`、`Attached`、`Detaching`、`Deleting`、`Error`。

如果磁盘进入 `phase: Error`，请先检查 `lastError`，然后再重试。

限制和恢复说明

- 在现有集群迁移路径中，只有尾部连续的磁盘可被声明。
- 控制器只保护在 `persistentDisk` 中声明的磁盘。任何未声明的磁盘仍遵循 VM 生命周期，并可能随旧 VM 一起被删除。
- 此迁移会更改保留磁盘的所有权模型。不要在 `DCSMachineTemplate` 和 `DCSIpHostnamePool` 中同时定义同一个磁盘。
- 如果需要保留 `/var/cpaas`，请在此次迁移中将其移入 IP 池，而不是将其留在模板中。
- 本操作指南适用于 ACP `v4.2.1` 或更高版本上的集群。

相关主题

- Huawei DCS 的基础设施资源
- 在 Huawei DCS 上升级 Kubernetes

将现有 **Huawei Cloud Stack** 集群迁移到由池管理的持久磁盘

当您将现有 Huawei Cloud Stack (HCS) 集群从旧版 `HCSMachineTemplate` 数据卷布局升级到由池管理的持久磁盘模型时，请使用本指南。

必须在节点替换后保留的磁盘在 `HCSMachineConfigPool.spec.configs[].persistentDisks[]` 中声明。其中包括平台所需的 `/var/cpaas` 磁盘。

> **INFO**
>
> 版本
>
> 当集群运行 ACP `v4.3.1` 或更高版本时，请使用此操作步骤。

目录

概述

开始之前

检查当前磁盘布局

决定要保留哪些磁盘

创建新的机器模板

准备滚动发布策略

更新机器配置池

触发滚动替换

验证迁移

故障处理

限制和恢复说明

相关主题

概述

旧版 HCS 集群通常会将 `/var/cpaas` 放置在 `HCSMachineTemplate.spec.template.spec.dataVolumes[]` 中。该布局会随 ECS 一同创建数据卷。在滚动替换期间，旧 ECS 及其模板拥有的数据卷可能会一并删除。

由池管理的模型会将升级期间需保留的磁盘移至 `HCSMachineConfigPool.spec.configs[].persistentDisks[]`。每个持久磁盘都绑定到固定的 `(hostname, slot)` 标识。在滚动替换期间，提供程序会：

1. 当旧 ECS 上的现有 EVS 磁盘与池声明匹配时，认领该磁盘。
2. 停止旧 ECS。
3. 分离 EVS 磁盘，并等待其变为可用状态。
4. 删除旧 ECS。
5. 创建替换 ECS，并在其首次启动前挂载相同的 EVS 磁盘。
6. 启动替换 ECS，替换 ECS 会挂载现有文件系统，而不会对其重新格式化。

开始之前

开始前，请验证以下所有事项：

- HCS provider 已安装在管理集群上。
- 业务集群运行正常，并且所有节点均为 `Ready`。
- 集群使用 `HCSMachineConfigPool` 分配固定的主机名和 IP 地址。

- 需保留的磁盘在旧版 `HCSMachineTemplate.spec.template.spec.dataVolumes[]` 中具有非空的挂载路径。
- 相关的滚动更新策略使用 `maxSurge: 0`。
- 您已为逐个节点替换安排维护窗口。
- 您拥有一份已验证的 etcd 和平台配置备份。

> **WARNING**
>
> 不要在 `HCSMachineTemplate.spec.template.spec.dataVolumes[]` 和 `HCSMachineConfigPool.spec.configs[].persistentDisks[]` 中声明相同的挂载路径。

检查当前磁盘布局

识别用于控制集群的管理集群对象：

```
kubectl get cluster -n cpaas-system
kubectl get kubeadmcontrolplane -n cpaas-system
kubectl get machinedeployment -n cpaas-system
kubectl get hcsmachinetemplate -n cpaas-system
kubectl get hcsmachineconfigpool -n cpaas-system
kubectl get hcsmachine -n cpaas-system
```

检查当前机器模板：

```
kubectl get hcsmachinetemplate <template-name> -n cpaas-system -o yaml
```

记录所有必须保留的 `dataVolumes[]` 条目。对于每个磁盘，请记录：

字段	来源
`mountPath`	`HCSMachineTemplate.spec.template.spec.dataVolumes[].mountPath`
`size`	`HCSMachineTemplate.spec.template.spec.dataVolumes[].size`

字段	来源
`type`	`HCSMachineTemplate.spec.template.spec.dataVolumes[].type`
`format`	`HCSMachineTemplate.spec.template.spec.dataVolumes[].format`

决定要保留哪些磁盘

仅将节点替换后必须保留的磁盘迁移到由池管理的模型。

按以下方式划分：

路径	建议的声明
`/var/cpaas`	`HCSMachineConfigPool.spec.configs[].persistentDisks[]`
存储在 `/var/cpaas` 下的监控或日志数据	`HCSMachineConfigPool.spec.configs[].persistentDisks[]`
`/var/lib/kubelet`	`HCSMachineTemplate.spec.template.spec.dataVolumes[]`，除非您的运维设计要求保留它
`/var/lib/containerd`	`HCSMachineTemplate.spec.template.spec.dataVolumes[]`，除非您的运维设计要求保留它
`/var/lib/etcd`	使用 etcd 备份和还原操作步骤进行灾难恢复。不要仅依赖节点本地磁盘保留。

对于自动迁移，provider 会通过匹配 `mountPath` 来认领现有数据卷。如果要保留的磁盘没有 `mountPath`，则无法自动认领。请使用受支持的运维操作步骤，将现有 EVS `volumeID` 记录到 `status.persistentDiskStatus[]` 中，或者在触发替换前将数据迁移到声明了挂载路径的磁盘。

创建新的机器模板

为替换节点创建新的 `HCSMachineTemplate` 资源。不要直接修改现有模板。

复制当前模板：

```
kubectl get hcsmachinetemplate <current-template-name> -n cpaas-system -o yaml > new-template.yaml
```

编辑 `new-template.yaml`：

1. 将 `metadata.name` 设置为新的模板名称。
2. 删除服务器生成的元数据，例如 `resourceVersion`、`uid`、`creationTimestamp`、`managedFields` 和 `status`。
3. 保持运行时身份字段未设置，包括 `spec.template.spec.providerID` 和 `spec.template.spec.serverId`。
4. 从 `spec.template.spec.dataVolumes[]` 中删除要保留的路径。
5. 仅保留可随每个 ECS 重新创建的临时数据卷。
6. 当此迁移属于 Kubernetes 或镜像升级的一部分时，更新 `spec.template.spec.imageName` 和其他升级字段。

例如，在 `/var/cpaas` 移入池后，模板仅保留临时磁盘：

hcs-machine-template-temporary-data-volumes.yaml

```yaml
apiVersion: infrastructure.cluster.x-k8s.io/v1beta1
kind: HCSMachineTemplate
metadata:
  name: <new-template-name>
  namespace: cpaas-system
spec:
  template:
    spec:
      imageName: <vm-image-name>
      flavorName: <instance-flavor>
      availabilityZone: <availability-zone>
      rootVolume:
        type: SSD
        size: 100
      configPoolRef:
        name: <pool-name>
      dataVolumes:
        - size: 100
          type: SSD
          mountPath: /var/lib/containerd
          format: xfs
```

应用新模板：

```
kubectl apply -f new-template.yaml -n cpaas-system
```

准备滚动发布策略

更新池之前，确认每个将使用更新后池的控制器的滚动发布策略。如果此次迁移没有工作节点池，请跳过 `MachineDeployment` 命令。

```
kubectl get kubeadmcontrolplane <kcp-name> -n cpaas-system \
  -o jsonpath='{.spec.rolloutStrategy.rollingUpdate.maxSurge}{"\n"}'

kubectl get machinedeployment <md-name> -n cpaas-system \
  -o jsonpath='{.spec.strategy.rollingUpdate.maxSurge}{"\n"}'
```

对于使用持久化磁盘的池，每个返回值都必须为 `0`。

如果任何返回值不是 `0`，请在更新池之前修补受影响的滚动发布策略：

```
kubectl patch kubeadmcontrolplane <kcp-name> -n cpaas-system \
  --type='merge' \
  -p='{"spec":{"rolloutStrategy":{"type":"RollingUpdate","rollingUpdate":
{"maxSurge":0}}}}'

kubectl patch machinedeployment <md-name> -n cpaas-system \
  --type='merge' \
  -p='{"spec":{"strategy":{"type":"RollingUpdate","rollingUpdate":{"maxSu
rge":0}}}}'
```

更新机器配置池

编辑由旧的和新的 `HCSMachineTemplate.spec.template.spec.configPoolRef.name` 引用的 `HCSMachineConfigPool`。

在每个必须保留磁盘的主机名下添加一个 `persistentDisks[]` 条目：

hcs-pool-persistent-disk.yaml

```yaml
apiVersion: infrastructure.cluster.x-k8s.io/v1beta1
kind: HCSMachineConfigPool
metadata:
  name: <pool-name>
  namespace: cpaas-system
  labels:
    cluster.x-k8s.io/cluster-name: <cluster-name>
spec:
  configs:
    - hostname: <node-hostname>
      networks:
        - subnetName: <subnet-name>
          ipAddress: <node-ip>
      persistentDisks:
        - slot: 0
          size: 40
          type: SSD
          mountPath: /var/cpaas
          format: xfs
          mountOptions:
            - defaults
            - noatime
```

编辑池时遵循以下规则：

- 对于每个主机名，将 `slot` 从 `0` 开始设置。
- 对于每个主机名，保持槽位连续。
- 将 `size`、`type`、`mountPath` 和 `format` 设置为要认领的旧数据卷的对应值。
- 每个池 `mountPath` 必须与现有数据卷的 `mountPath` 完全匹配。否则会创建新的空磁盘，而不是使用旧磁盘，并且旧磁盘会随旧 ECS 一起删除。
- 如果池尚未包含 `cluster.x-k8s.io/cluster-name: <cluster-name>`，请添加它。
- 对于必须保留相同路径的所有节点，使用相同的持久化磁盘声明。

仅当替换模板已存在、已从 `dataVolumes[]` 中删除要保留的路径，并且滚动发布策略使用 `maxSurge: 0` 后，才应用更新后的池：

```bash
kubectl apply -f hcs-pool-persistent-disk.yaml -n cpaas-system
```

触发滚动替换

在替换任何旧主机之前应用更新后的池。如果替换旧主机时池尚未声明 `persistentDisks[]`，其旧数据磁盘会与 ECS 一起删除。

应用更新后的池后，立即在同一维护窗口内将控制平面或工作节点控制器指向新模板。请勿在活动滚动发布仍指向旧模板，且该旧模板也在 `dataVolumes[]` 中声明 `/var/cpaas` 的情况下，保留声明 `/var/cpaas` 持久化磁盘的池。

对于控制平面迁移，将 `KubeadmControlPlane` 指向新模板：

```
kubectl patch kubeadmcontrolplane <kcp-name> -n cpaas-system \
  --type='merge' \
  -p='{"spec":{"machineTemplate":{"infrastructureRef":{"name":"<new-templ
ate-name>"}}}}'
```

对于工作节点迁移，将 `MachineDeployment` 指向新模板：

```
kubectl patch machinedeployment <md-name> -n cpaas-system \
  --type='merge' \
  -p='{"spec":{"template":{"spec":{"infrastructureRef":{"name":"<new-temp
late-name>"}}}}}'
```

如果模板引用已指向目标模板，并且需要强制逐一替换，请设置 `rolloutAfter`：

```
kubectl patch kubeadmcontrolplane <kcp-name> -n cpaas-system \
  --type='merge' \
  -p='{"spec": {"rolloutAfter": "'"$(date -u +%Y-%m-%dT%H:%M:%SZ)"'"}}'

kubectl patch machinedeployment <md-name> -n cpaas-system \
  --type='merge' \
  -p='{"spec": {"rolloutAfter": "'"$(date -u +%Y-%m-%dT%H:%M:%SZ)"'"}}'
```

验证迁移

监控滚动替换：

```
kubectl get kubeadmcontrolplane <kcp-name> -n cpaas-system -w
kubectl get machinedeployment <md-name> -n cpaas-system -w
kubectl get machine -n cpaas-system -w
kubectl get hcsmachine -n cpaas-system -w
```

检查池状态：

```
kubectl get hcsmachineconfigpool <pool-name> -n cpaas-system -o yaml
```

每个已迁移的磁盘都会显示在 `status.persistentDiskStatus` 下：

阶段	含义
`Creating`	提供商正在为此主机名和槽位创建 EVS 卷。
`Available`	EVS 卷已就绪，尚未连接到任何 ECS。
`Attaching`	替换 ECS 已存在；提供商正在确认连接状态。
`Attached`	EVS 磁盘已连接到当前 ECS。
`Detaching`	正在移除旧 ECS 前卸载磁盘。
`Deleting`	正在删除池；EVS 卷正在被移除。
`Error`	提供商无法安全继续。请检查 `lastError`。

确认替换节点可以从保留的路径读取数据：

```
kubectl --kubeconfig <workload-kubeconfig> debug node/<node-name> --image
=busybox -- \
  chroot /host sh -c 'mount | grep /var/cpaas && ls -la /var/cpaas'
```

如需进行更严格的数据保留检查，请在滚动发布前写入标记，并在替换节点成为 `Ready` 后读取该标记：

```
kubectl --kubeconfig <workload-kubeconfig> debug node/<node-name> --image
=busybox -- \
  chroot /host sh -c 'cat /var/cpaas/<marker-file>'
```

故障处理

当磁盘进入 `phase: Error` 时，使用池状态决定下一步操作。

症状	可能原因	操作
`lastError` 报告挂载路径冲突	`dataVolumes[]` 和 `persistentDisks[]` 中存在相同路径	从新的 `HCSMachineTemplate` 中移除保留路径，然后重试滚动发布。
`lastError` 报告大小或类型不匹配	池声明与现有 EVS 磁盘不匹配	更正池条目，使 `size` 和 `type` 与现有磁盘匹配。
`lastError` 报告可用区冲突	现有 EVS 磁盘与目标 ECS 位于不同的可用区	使用与 EVS 磁盘匹配的 ECS 可用区，或在重试前将磁盘迁移到目标可用区。
`lastError` 报告 EVS 磁盘已连接到其他 ECS	磁盘在当前替换流程之外处于连接状态	仅在确认所有权后手动断开磁盘连接，然后让提供商再次进行协调。
`lastError` 报告未找到 `volumeID`	被跟踪的 EVS 磁盘已被删除	在重试前，使用正确的所有权标签恢复或重新创建预期的 EVS 磁盘。

当持久化磁盘处于未解决的错误状态时，请勿删除旧的 `HCSMachine`，也不要强制移除 finalizer。提供商会阻止删除操作，以避免在能够安全声明并卸载磁盘之前删除旧 ECS。

限制和恢复说明

- 此操作步骤适用于使用 `HCSMachineConfigPool`、且主机名和 IP 地址固定的集群。
- 由池管理的持久化磁盘必须逐个替换。对于使用持久化磁盘的每次控制平面或工作节点滚动发布，请保留 `maxSurge: 0`。

- 提供商会通过匹配非空的 `mountPath` 自动声明现有数据卷。
- 未在 `persistentDisks[]` 中声明的 `dataVolumes[]` 仍由模板管理，可能会随旧 ECS 一起删除。
- 提供商接受持久化磁盘条目后，请将 `slot`、`size`、`type`、`format` 和 `mountPath` 视为不可变。
- `mountOptions` 可以更改，但更改仅会在替换 VM 时生效。
- 单控制平面 HCS 集群仅支持创建，无法升级。请勿对单控制平面集群使用此滚动迁移操作步骤。

相关主题

- Huawei Cloud Stack 的基础设施资源
- 在 Huawei Cloud Stack 上创建集群
- 在 Huawei Cloud Stack 上升级 Kubernetes

升级验证

在关闭维护窗口前，验证每个已升级的集群。在 global DR 环境中，分别验证备用和主 global 集群，然后再升级业务集群。对于 Alauda OS 集群，在 `global` 集群 或业务集群完成在 Alauda OS 上升级 Kubernetes 后，执行此验证。

目录

确认 Distribution 版本和 CVO 完成状态

验证 Kubernetes、节点、Core 和对齐插件

验证已安装扩展的可用性

验证 Web UI 和正在使用的 Agnostic 插件

确认 **Distribution** 版本和 **CVO** 完成状态

从 global 集群检查目标集群的 `ClusterVersionShadow` ：

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

当前 Distribution 版本和期望的 Distribution 版本都必须与目标版本匹配。`Ready` 必须为 `True`，`Reconciling` 不得再为 `True`，并且 `Stalled` 不得为 `True`。最新的历史记录项必须显示目标版本已成功完成。

验证 Kubernetes、节点、Core 和对齐插件

从 global 集群管理上下文中，确认 Core 和已安装 Aligned 模块的状态：

```
kubectl get clustermodule <cluster> -o yaml
kubectl get moduleinfo -l cpaas.io/cluster-name=<cluster>
```

然后，对已升级的集群使用 `kubectl` 上下文，确认 Kubernetes 服务器版本、节点就绪状态和 Pod：

```
kubectl version
kubectl get nodes
kubectl get pods --all-namespaces \
  | awk '{if ($4 != "Running" && $4 != "Completed")print}' \
  | awk -F'[/ ]+' '{if ($3 != $4)print}'
```

所有节点都必须为 `Ready`；Core 和已安装的 Aligned 模块必须在健康阶段报告目标版本；关键 Pod 必须为 `Running` 或 `Completed`。

验证已安装扩展的可用性

如果需要确认已升级的应用仍可被发现，请从 global 集群查看集群插件目录：

```
kubectl get moduleplugins
```

升级验收适用于已安装在升级集群上的 Core 和 Aligned 应用。使用前几节中的 `ClusterVersionShadow` 完成状态和 `ModuleInfo` 版本作为权威结果，并确认没有相关 Pod 报告 `ImagePullBackOff` 或 `ErrImagePull`。

无需要求 `ProductBase.status.artifacts` 中的每个条目都为 `Ready`。`ProductBase` 包含可选或未安装的目录软件包，包括 Agnostic 应用；集群成功升级后，这些软件包可以合法地保持为 `Absent`。

验证 Web UI 和正在使用的 Agnostic 插件

打开平台 Web UI，并确认登录页面、集群页面和 Marketplace 页面均可正常加载。升级目标 Kubernetes 版本所需的每个正在使用的 Agnostic 插件，然后验证其 UI 和主要功能。

从早于 ACP 4.3 的版本升级时，请使用与目标版本兼容的以下 Agnostic 插件版本，否则其页面可能无法打开：

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

完成适用的产品专属操作：

- 升级 Alauda AI
- 升级 Alauda DevOps

如果源版本早于 ACP 4.3，且尚未完成 PKCE 加固，请等待 global 层和所有业务集群达到目标 Distribution Version，然后按照禁用 PKCE Plain 方法中的说明操作。

如果任何验收检查失败，请保持维护窗口开启，并参阅升级故障排除。

升级故障排查

当 preflight 报告阻塞性检查、CVO 报告 `Ready=False`、`Reconciling=True` 或 `Stalled=True`，或者 Core 或 Aligned 模块未到达其目标版本时，请使用本页。

目录

检查升级状态
处理 Preflight 阻塞
解决其他常见的 Preflight 阻塞
处理管理员确认门禁
恢复缺失的 Aligned 包
Aligned 集群插件
Aligned operators
诊断无法继续推进的模块
在升级升级前收集证据

检查升级状态

将 `<cluster>` 替换为 `global` 或业务集群名称：

```
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.preflig
ht.checks[*]}{.name}{"\t"}{.policy}{"\t"}{.state}{"\t"}{.reason}{"\t"}{.m
essage}{"\n"}{end}'

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

在更改升级请求之前，先解决第一个失败的 preflight 检查或相关的 `Ready`、`Reconciling` 或 `Stalled` 条件。

处理 **Preflight** 阻塞

`ResourcePatch` 记录了应用于平台资源的自定义项。升级不会覆盖此类自定义项，除非目标版本已对其豁免，因此在已自定义的环境中，`ResourcePatchUpgradeable` 是最常见的阻塞项。

如果 `ResourcePatchUpgradeable` 失败并显示 `reason=UnexemptResourcePatches`，请检查指定的 `ResourcePatch`，并且仅在审查该 patch 之后再添加所需的目标版本豁免：

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

kubectl get resourcepatches <rp-name> -o yaml
kubectl annotate resourcepatches <rp-name> \
  config.cpaas.io/exempt-for-ver=<target-version> \
  --overwrite
```

豁免注解的 key 是 `config.cpaas.io/exempt-for-ver`，其 value 为目标版本。豁免是按 major 和 minor 版本匹配的，因此，为某个 patch 版本记录的豁免也适用于同一 minor release 的其他 patch 版本。

对 patch 进行豁免意味着升级会保留该自定义项不变。它不会验证该自定义项在目标版本中是否仍然有效——请向提供该自定义项的技术支持联系人确认。

解决其他常见的 **Preflight** 阻塞

使用 检查升级状态 中的命令列出所有不是 `Passed` 的检查，然后逐一解决：

检查	验证内容	处理方式
`KubernetesVersionSupported`	集群的 Kubernetes 版本位于目标 release 支持的范围内。	请参阅 Kubernetes Support Matrix；不要禁用该检查。
`VersionUpgradePath`	当前 patch 能通过受支持的升级路径到达请求的目标版本。	使用 `availableUpdates` 中的目标版本，或遵循提供的中间目标版本。
`ClusterRunning`	集群健康且已完成协调。	检查 `kubectl get clusterview <cluster>`，并解决不健康的节点或组件。
`DockerRuntimeUnsupported`	没有节点使用不受支持的 Docker runtime。	将受影响的节点迁移到 `containerd`。
`ClusterModuleStable`，`ModuleInfoStable`	Core 和已安装的 Aligned 模块处于稳定状态。	请遵循 在升级前验证模块稳定性。

不要强行绕过失败的检查继续升级。`VersionUpgradePath`、`KubernetesVersionSupported` 和 `AdminAckRequired` 是支持的升级路径本身的门禁；绕过它们的升级会进入平台未验证的状态。如果无法通过本页解决阻塞，请在继续之前联系技术支持。

处理管理员确认门禁

如果 `AdminAckRequired` 失败，请检查目标 release 提供的 keys：

```
kubectl -n cpaas-system get configmap admin-gates -o yaml
```

完成适用于相应门禁所描述的操作。对于 Kubernetes 1.35 或更高版本的 node-readiness 门禁，请在每个生产节点上完成 Kubernetes 1.35 或更高版本节点就绪状态。

针对该要求引入的第一个门禁，也就是 ACP 4.4 和 Kubernetes 1.35 目标所使用的门禁，是 `ack-4.4-kubernetes-1.35-kernel-update`。请将其视为示例——后续目标 release 可能提供不同的 key，因此始终应从 `admin-gates` 中读取 key，而不要重复使用这里的示例。

从 `admin-gates` 复制适用的 key，在 `admin-acks` 中记录确认，然后重新运行 preflight：

```
ACK_KEY='<key-from-admin-gates>'
kubectl -n cpaas-system patch configmap admin-acks --type merge \
  -p "{\"data\":{\"${ACK_KEY}\":\"true\"}}"

bash upgrade.sh --preflight
```

确认该检查现在通过：

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

恢复缺失的 Aligned 包

不可用的 Aligned 包不会改变应用当前的已安装状态。根据包类型和协调阶段，CVO 可能通过 `Ready=False` 或 `Reconciling=True` 报告问题，而不是 `Stalled=True`。

Aligned 集群插件

如果某个 `ClusterVersionShadow` 条件包含 `required ready ModulePluginConfig for installed platform-aligned component`，请识别目标 `ProductManifest` 和指定的 `ModulePluginConfig`：

```
kubectl get productmanifest v<target-version> -o yaml

kubectl get modulepluginconfig <modulepluginconfig-name> \
  -o jsonpath='{.spec.image}{"\n"}{range .status.conditions[*]}{.type}
{"\t"}{.status}{"\t"}{.reason}{"\t"}{.message}{"\n"}{end}'
```

对于 ModulePlugin channel，目标 `ProductManifest` 中的 `artifactStatus: Absent` 表示匹配的目标 `ModulePluginConfig` 尚未完全 Ready。它并不表示当前已安装的应用已变为 absent，也不能单凭这一点证明镜像在 registry 中缺失。

使用 `ModulePluginConfig` 条件和 `.spec.image` 中的确切镜像来选择恢复方式：

证据	恢复方式
指定的 `ModulePluginConfig` 不存在	检查目标 `ProductManifest` 的条件和控制器协调情况。对象缺失本身并不能证明包镜像不存在。
`ResolveDigestFailed` 报告 `manifest unknown` 或 `not found`，或者 registry 确认 `.spec.image` 不存在	使用下面适用的方法重新发布确切的目标包。
该条件报告 `unauthorized`、`denied` 或其他身份验证错误	修正 registry 凭据或仓库权限，然后让现有请求重试。
该条件报告 `x509` 或证书信任错误	修正 registry 证书链或节点信任配置。
该条件报告 DNS、路由、超时或连接错误	从执行解析的组件恢复到 registry 的可达性。
`LoadModulePluginFailed`	检查包内容和 `module-plugin.yaml`；仅在确认包无效后，发布修正后的包。

- 对于 **ACP Upgrade to v4.5** 中的应用，将其包复制到目标 Core Package 的 `plugins/` 目录中。使用平台内置 Registry 时，重新运行 `upgrade.sh --only-sync-image`。使用外部 registry 时，重新运行 Prepare the Target-Version Payload 中的两种模式。
- 对于其他 Aligned 集群插件，使用 `violet push` 将目标包发布到 global 层级。

Aligned operators

Aligned operators 不使用 `ModulePluginConfig`。如果某个已安装的 operator 未继续升级，请在受影响的集群上检查其 `Subscription`、目标 `InstallPlan`、catalog 和 `ModuleInfo`：

```
kubectl get subscription -A
kubectl get installplan -A
kubectl get moduleinfo -l cpaas.io/cluster-name=<cluster>
```

如果目标 operator 包缺失，请将其发布到安装了该 operator 的每个集群。如果只有一个业务集群缺失该包，请使用 `--clusters "<workload-cluster-name>"` 将其推送到该集群。如果包已存在，请解决 CVO 实际报告的 `Subscription`、`InstallPlan`、catalog 或 `ModuleInfo` 条件。

在修复问题后，继续观察现有请求。CVO 会自动重试；不要清除或重新提交 `desiredUpdate`。

诊断无法继续推进的模块

```
kubectl get moduleinfo -l cpaas.io/cluster-name=<cluster> \
  -o custom-columns='MODULE:.metadata.labels.cpaas\.io/module-name,CURREN
T:.status.version,TARGET:.spec.version,NEW:.status.availableVersions[0].v
ersion,PHASE:.status.phase'

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

检查指定资源及相关 Events。对于镜像拉取失败，请验证 registry 可达性、CA 信任、拉取凭据，以及受影响节点上引用的 manifest。

在升级升级前收集证据

记录 `cvsh` 条件、preflight 检查、阶段和历史、受影响的 `ModuleInfo` 或 `ModulePluginConfig`、相关 Events，以及确切的源版本和目标版本。不要包含平台 token、registry 密码或 Secret 内容。