Menu

集群

概览

概览

不可变基础设施

不可变基础设施

节点管理

概览

节点是集群的基本构建单元。添加到集群中的节点可以是虚拟机或物理服务器。每个节点包含运行 Pods 所需的基本组件，包括 Kubelet、Kube-proxy 和容器运行时。

向本地集群添加节点

平台管理员可以向平台管理下的自管集群添加节点。

管理节点

支持更新节点标签以及添加或删除自定义节点标签。

节点监控

在节点详情页面查看节点监控数据。

托管集群

概述

导入集群

注册集群

公有云集群初始化

公有云集群初始化。

如何操作

创建本地集群

创建本地集群

托管控制平面

托管控制平面

集群节点规划

集群节点规划

etcd 加密

etcd 加密

如何操作

为内置镜像仓库添加外部访问地址

选择容器运行时

使用 **Manager** 策略优化 **Pod** 性能

更新公共仓库凭证

集群概述

平台支持多种 Kubernetes 集群管理模型，具体取决于底层基础设施的配置方式以及控制平面的部署方式。

目录

Platform-Provisioned 基础设施

描述：

在此模型中，平台负责配置机器和节点操作系统。 所有节点均使用 **Immutable OS**，确保基础设施状态一致、声明式且易于恢复。 该模型实现了整个集群生命周期的全自动化——从配置到扩展和升级。

Immutable OS 示例：

常见的 Immutable OS 包括 **Fedora CoreOS**、**Flatcar Linux** 和 **openSUSE MicroOS**。 目前，平台支持使用 **MicroOS** 进行不可变节点管理。

职责划分：

组件	管理方
机器 / 节点	平台
节点操作系统	平台（仅限 Immutable OS）
Kubernetes	平台

User-Provisioned 基础设施

描述：

在此模型中，用户提供预配置的物理或虚拟机器。 平台在这些节点上安装并管理 Kubernetes，而节点操作系统的管理（包括配置、补丁或替换）由用户负责。

该模型适用于已有成熟基础设施或操作系统管理流程或自动化工具的组织。

职责划分：

组件	管理方
机器 / 节点	用户
节点操作系统	用户
Kubernetes	平台

Hosted Control Plane (HCP)

描述：

Hosted Control Plane (HCP) 是一种部署模型，多个集群共享一个托管于专用管理集群中的控制平面。 仅控制平面组件共享——工作节点仍按照上述两种基础设施模型之一进行配置（Platform-Provisioned 或 User-Provisioned）。

特点：

- 降低控制平面资源消耗。
- 支持混合模型：工作节点可为 Immutable 或 User-Provisioned。
- 适合大型裸金属或资源受限环境。

连接集群

平台还支持连接和管理现有 Kubernetes 集群，无论是公有云集群还是符合 CNCF 标准的 Kubernetes 发行版。

公有云 **Kubernetes**

- 通过云特定提供者（如 *Alauda Container Platform EKS Provider*）连接托管 Kubernetes 服务，如 EKS、AKS 和 GKE。
- 云凭据可安全存储于平台。
- 支持直接从平台创建和管理公有云集群。

CNCF 标准 **Kubernetes**

- 连接任何符合 CNCF 标准的现有 Kubernetes 集群。
- 支持跨环境的统一可视化、策略控制和监控。
- 参见支持的 Kubernetes 版本列表。

基于隧道的连接

- 当 **global** 集群 无法直接访问 **workload** 集群 时，**Tunnel Server**（global 端）和 **Tunnel Agent**（workload 端）建立安全通信。
- 适用于断网或受限网络环境。

选择合适的模型

场景	基础设施配置方	节点操作系统管理方	**Kubernetes** 管理方	自动化程度
Platform-Provisioned 基础设施	平台	平台（仅限 Immutable OS）	平台	完全
User-Provisioned 基础设施	用户	用户	平台	部分
Hosted Control Plane (HCP)	平台	共享节点（平台）	平台	部分
连接集群（云或 CNCF）	外部提供者	外部提供者	部分 / 外部	最低

关于不可变基础设施

不可变基础设施使用不可变操作系统来部署 Kubernetes 集群。与传统的基于操作系统的集群不同，所有节点配置都预先集成在镜像中，部署后保持不变。集群升级和配置更改通过替换带有新镜像的节点来实现，确保了集群生命周期内的一致性、可靠性和简化的运维。

Note

因为 Immutable Infrastructure 的发版周期与灵雀云容器平台不同，所以 Immutable Infrastructure 的文档现在作为独立的文档站点托管在 Immutable Infrastructure ↗。

Menu

节点管理

概览

节点是集群的基本构建单元。添加到集群中的节点可以是虚拟机或物理服务器。每个节点包含运行 Pods 所需的基本组件，包括 Kubelet、Kube-proxy 和容器运行时。

向本地集群添加节点

平台管理员可以向平台管理下的自管集群添加节点。

管理节点

支持更新节点标签以及添加或删除自定义节点标签。

节点监控

在节点详情页面查看节点监控数据。

概览

节点是集群的基本构建单元。添加到集群中的节点可以是虚拟机或物理服务器。每个节点包含运行 Pods 所需的基本组件，包括 Kubelet、Kube-proxy 和容器运行时。

具有平台管理权限的用户可以在集群下管理节点。

注意：不支持向导入的集群添加节点或从导入的集群删除节点。

目录

节点类型

- 控制平面节点：负责运行集群组件，如 kube-apiserver、kube-scheduler、kube-controller-manager、etcd、容器网络以及部分平台管理组件。
  - 当允许在控制平面节点上部署应用时，控制平面节点也可以作为计算节点使用。
  - 必须至少添加 1 个控制平面节点。不支持设置 2 个控制平面节点。设置 3 个或更多控制平面节点时，集群成为高可用集群（对于高可用集群，建议使用奇数个节点，优选 3 或 5 个）。
  - 当控制平面节点数量达到 3 个或更多时，集群具备多副本容灾能力，被视为高可用集群。

- 计算节点：负责承载运行在集群上的业务 Pods。集群中所需的计算节点数量通常可根据业务量进行规划。

Linux 节点可用性检查

如果需要构建本地集群，请首先参考 cluster check 以确保所有节点配置符合要求。所有前提条件必须满足，否则集群部署可能失败。

支持的操作系统和 CPU 型号

请参考 Supported Operating Systems and CPU Models。

向本地集群添加节点

当集群需要扩容或集群中异常节点需要替换为新节点时，可以通过添加节点的方式向平台上现有的本地工作负载集群中添加控制平面节点和计算节点。

目录

约束与限制

- 待添加到集群的节点必须提前准备。请参考节点可用性检查参考准备并检查待添加到集群的节点，确保满足所有条件，否则集群部署可能失败。
- 待添加节点的硬件架构必须与集群的硬件架构保持一致。
- 为避免不可预知的错误，待添加节点的操作系统类型应与集群中其他节点保持一致。
- 在同一添加节点对话框中添加的节点，其 SSH 端口和认证信息必须统一。
- 集群规划指导：集群必须至少有 1 个控制平面节点。不支持设置恰好 2 个控制平面节点。控制平面节点数达到 3 个及以上时，集群即为高可用集群（对于高可用集群，建议使用奇数个

节点，优选 3 个或 5 个）。注意：该要求仅在添加或变更控制平面容量时适用；添加工作/计算节点时无需强制添加控制平面节点。

- 一个节点只能属于一个集群。待添加节点不能被其他集群占用。

前提条件

- 当 global 集群无法通过 SSH 服务直接访问待添加节点，需要通过代理访问时，请提前准备代理服务。目前仅支持 SOCKS5 代理。

操作步骤

1. 在左侧导航栏点击 **Clusters** > **Clusters**。
2. 点击需要添加节点的类型为 **On-Premises** 的*集群名称*。
3. 在 **Nodes** 标签页下，点击 **Add Node**。
4. 参考节点配置参数配置相关参数。
5. 点击 **Add** 对节点进行可用性检查。
   检查通过后，开始添加节点，节点状态为 **Adding**。

后续操作

查看执行进度

在节点列表页面，可以查看已添加节点的列表信息。对于处于 **Adding** 状态的节点，可以查看执行进度。

操作步骤

1. 点击处于 **Adding** 状态节点右侧的 **View Execution Progress**。

2. 在弹出的执行进度对话框中，可以查看节点执行进度（`status.conditions`）。

   提示：当某类型处于执行中或失败状态且带有原因时，可将鼠标悬停在对应的原因（蓝色文字显示）上查看详细的原因信息（`status.conditions.reason`）。

重新添加失败节点

添加节点后，如果部分节点添加失败，节点列表上方会出现提示。点击提示框中的 **Re-add** 按钮即可重新添加失败节点。

Manage Nodes

目录

Update Node Labels

Labels↗ 是附加到节点上的键值对，用于定义节点属性。为节点设置标签后，可以通过标签轻松过滤或选择节点。例如：将 Pods 指定调度到特定节点。

支持对处于正常状态的节点更新节点标签，添加或删除自定义节点标签。

操作步骤

1. 在左侧导航栏，点击 **Cluster Management** > **Clusters**。
2. 点击包含待更新标签节点的 *集群名称*。
3. 在 **Nodes** 选项卡下，点击待更新标签节点右侧的 **Update Node Labels**。
4. 添加、修改或删除节点标签。
5. 点击 **OK**。
   成功更新节点标签后，节点标签数量会发生变化。您可以在 **Node** 信息栏的 **Node Labels** 项中查看该节点的所有标签信息。

Stop/Resume Node Scheduling

通过设置节点的调度状态，可以控制集群中新创建的 Pods 是否允许调度到该节点。

- **Stop Scheduling**：不允许新创建的 Pods 调度到该节点，但不影响已运行在该节点上的 Pods。
- **Resume Scheduling**：允许新创建的 Pods 调度到该节点。

操作步骤

1. 在左侧导航栏，点击 **Clusters** > **Clusters**。
2. 点击包含待停止/恢复调度节点的 *集群名称*。
3. 在 **Nodes** 选项卡下，点击待设置调度状态节点右侧的 **Stop Scheduling/Resume Scheduling**。
4. 点击 **OK**。

Evict Pods

将处于正常状态的节点上除由 DaemonSet（守护进程集）管理的 Pods 以外的所有 Pods 驱逐到集群中的其他节点，并将该节点设置为不可调度状态。

注意：本地存储的 Pods 数据在驱逐后将丢失，请谨慎操作。

操作步骤

1. 在左侧导航栏，点击 **Cluster Management** > **Clusters**。
2. 点击包含待驱逐 Pods 节点的 *集群名称*。
3. 在 **Nodes** 选项卡下，点击待驱逐 Pods 的 *节点名称*。
4. 在右上角，点击 **Actions** > **Evict Pods**。
5. 查看待驱逐 Pods 信息后，点击 **Evict**。

Set Taints

为处于正常状态的节点设置污点信息。

污点是节点的一种属性，允许节点拒绝运行某些类型的 Pods，甚至驱逐 Pods。污点与 Pods 上的容忍度（tolerations）配合使用，防止 Pods 被分配到不合适的节点。每个节点可以应用一个或多个污点，无法容忍这些污点的 Pods 将不会被节点接受。

例如：当发现某节点内存利用率达到 91% 时，不建议继续调度新的 Pods 到该节点，可以为其设置污点。设置污点后，Kubernetes 将不会调度 Pods 到该节点。

了解更多...↗

操作步骤

1. 在左侧导航栏，点击 **Cluster Management** > **Clusters**。
2. 点击包含待设置污点节点的 *集群名称*。

3. 在 **Nodes** 选项卡下，点击待设置污点节点右侧的 **Set Taints**。

4. 参考以下说明设置污点的键、值及效果。节点可添加多个污点。

   污点属性由 `key=value [effect]` 组成。

   `key=value` 用于匹配 Pod 的容忍度。该污点表示节点被 `key=value` 污染，除非 Pod 能容忍（Tolerations）该污点，否则不允许或应避免调度到该节点。

   effect 表示污点的效果，有以下三种选项：

   - **NoSchedule**：表示不允许调度，已调度的资源不受影响。
   - **PreferNoSchedule**：表示尽量不调度。
   - **NoExecute**：表示不允许调度，且已调度的资源将在 `tolerationSeconds` 后被删除。

5. 点击 **OK**。

Label and Taint Management

平台支持对节点批量设置标签和污点。

约束与限制

- 设置设备标签前，需要先在集群中部署设备插件，如 NVIDIA GPU MPS 设备插件、NVIDIA GPU 设备插件、GPU Manager 设备插件等。

  提示：设备标签实际上是节点标签。为方便操作，平台将设备插件依赖的节点标签归类为设备标签，便于快速配置。

操作步骤

1. 在左侧导航栏，点击 **Clusters** > **Clusters**。

2. 点击需要管理标签和污点的 *集群名称*。

3. 在 **Nodes** 选项卡下，多选需要管理的节点，点击 **Label and Taint Management** 按钮。

提示：可在节点列表页的搜索框输入关注的节点标签，快速筛选出需要管理标签和污点的节点列表。

4. 在 **Batch Operations** 中添加并填写要执行的操作，点击 OK 提交批量操作到集群。

   - **Node Labels**：可对选中节点 添加/更新 指定标签，或 删除 指定标签。选择删除时，平台会过滤选中节点上的所有标签列表。当值设置为 **Any**，表示删除所有包含指定标签键的节点标签。

   - **Taints**：可对选中节点 添加/更新 指定污点，或 删除 指定污点。选择删除时，平台会过滤选中节点上的所有污点列表。当值设置为 **Any**，表示删除所有包含指定污点键的节点污点。

   - **Device Labels**：可为选中节点设置所需使用的设备，设备列表来源于您在该集群中部署的设备插件。

Enable/Disable Virtualization Switch

当本地集群中的节点为物理机时，可通过启用/禁用节点虚拟化开关，控制 Kubernetes 是否允许将虚拟机（VMI，VirtualMachineInstance）调度到该节点。

启用开关时，允许新创建的虚拟机调度到物理机节点；禁用开关时，禁止新创建的虚拟机调度到物理机节点，但不影响已运行在该节点上的虚拟机。

提示：相关操作及注意事项，请参见 Prepare Virtualization Environment。

Delete On-Premises Cluster Nodes

支持删除类型为本地部署的集群中的节点。例如：删除本地部署集群中的故障节点。

约束与限制

- 不支持删除导入的集群中的节点。
- 集群中仅有一个控制平面节点时，不支持删除该控制平面节点。

操作步骤

1. 在左侧导航栏，点击 **Cluster Management** > **Clusters**。

2. 点击类型为 **On-Premises** 且包含待删除节点的 *集群名称*。

3. 在 **Nodes** 选项卡下，点击待删除节点右侧的 **Delete**。

   提示：删除 Linux 节点后，如需清理节点上的资源，可点击对话框底部的 **Download Cleanup Script** 下载清理脚本到本地。节点删除成功后，登录该节点执行清理脚本。

4. 输入节点名称，点击 **Delete**。

节点监控

在节点详情页面查看节点监控数据。

TIP

- 当集群中节点数量超过1个时，可以点击节点详情页面资源路径区域中的*当前节点名称*，展开节点下拉列表，然后点击选择节点，快速切换到其他节点详情页面。
- 当集群配置了监控组件后，可以查看节点监控数据，包括资源运行状态、资源使用情况及资源趋势统计。

目录

操作步骤

1. 在左侧导航栏，点击 **Clusters** > **Clusters**。
2. 点击目标节点所在的*集群名称*。
3. 在 **Nodes** 标签页下，点击目标*节点名称*。
4. 点击 **Monitoring** 标签，进入节点监控数据展示页面，查看相关节点监控数据。

TIP

- 鼠标悬停在卡片上，点击 **Details** 图标查看 PromQL 表达式；点击 **Export** 图标导出当前页面所有图表的 PromQL 表达式。
- 当集群中节点数量超过1个时，可以点击节点详情页面资源路径区域中的*当前节点名称*，展开节点下拉列表，然后点击选择节点，快速切换到其他节点详情页面。

TIP

在存储空间统计展示区域，当节点拥有超过4个存储分区时：

- 在分区总使用饼图中，使用率最高的前三个分区单独展示，其余分区合并显示为 **Others**，鼠标悬停时显示其总使用数据；
- 在分区使用柱状图中，使用率最高的前三个分区单独展示，其余分区合并显示为 **Others**，鼠标悬停时显示其总使用量及各自使用率。

监控趋势统计说明如下表。

参数	说明
CPU	指定时间范围内 CPU 的使用率、请求率和限制率。 使用率 = 节点上所有 Pod 的 CPU 使用量 / 节点总 CPU。 注意：若节点 CPU 使用率在某段时间内出现峰值，需先定位消耗 CPU 资源最多的进程。例如，对于 Java 自定义应用，代码中的内存泄漏或死循环可能导致 CPU 使用率升高。 请求率 = 节点上所有 Pod 的 CPU 请求量 / 节点总 CPU。 注意：若节点 CPU 请求率在某段时间内出现峰值，可能是由于集群超额预订比例设置不合理，或节点上运行的 Pod 请求值过高，可能导致资源浪费。 限制率 = 节点上所有 Pod 的 CPU 限制量 / 节点总 CPU。 注意：若节点 CPU 限制率在某段时间内出现峰值，表示节点上 Pod 的限制值设置过高，可能导致 CPU 资源浪费。

参数	说明
Memory	指定时间范围内内存的使用率、请求率和限制率。 使用率 = 节点上所有 Pod 的内存使用量 / 节点总内存。 内存是服务器的重要组成部分，是 CPU 通信的桥梁，因此内存性能对机器影响显著。程序运行时，数据加载、线程并发和 I/O 缓冲均依赖内存。可用内存大小决定程序能否正常运行及运行效率。 请求率 = 节点上所有 Pod 的内存请求量 / 节点总内存。 注意：若节点内存请求率在某段时间内出现峰值，可能是由于集群超额预订比例设置不合理，或节点上运行的 Pod 请求值过高，可能导致资源浪费。 限制率 = 节点上所有 Pod 的内存限制量 / 节点总内存。 注意：若节点内存限制率在某段时间内出现峰值，表示节点上 Pod 的限制值设置过高，可能导致内存资源浪费。
Storage	指定时间范围内的空间使用率和**inode** 使用率。 空间使用率 = 已使用存储空间 / 总存储空间。 通过监控历史磁盘空间数据，可以评估某段时间内的磁盘使用情况。当磁盘使用率较高时，可通过清理无用镜像或容器释放磁盘空间。 **inode** 使用率 = 已使用 inode 数量 / 总 inode 数量。 注意：每个文件必须有一个 inode 来存储文件元数据，如文件创建者和创建日期。inode 也会占用磁盘空间，许多小缓存文件容易导致 inode 资源耗尽。此外，当 inode 耗尽但磁盘未满时，无法在磁盘上创建新文件。
System Load	1 分钟、5 分钟和 15 分钟的平均 CPU 负载。该值为 CPU 当前正在执行和等待执行的进程总数与 CPU 最大可执行进程数的比值，是系统忙闲状态的重要指标。 注意：若 1 分钟/5 分钟/15 分钟曲线在某段时间内相似，表示集群 CPU 负载较为稳定。

参数	说明
	若某段时间或特定时间点 1 分钟值远大于 15 分钟值，表示最近 1 分钟负载上升，需要持续观察。若 1 分钟值超过 CPU 数量，可能表示系统过载，需要进一步分析问题根因。 若某段时间或特定时间点 1 分钟值远小于 15 分钟值，表示系统负载在最近 1 分钟下降，之前 15 分钟内负载较高。
Disk Throughput	指定时间范围内的磁盘吞吐量，指磁盘传输数据的速度，传输数据为读写数据总和。
Disk IOPS	指定时间范围内的磁盘 IOPS，即每秒连续读写操作次数之和，代表磁盘每秒读写操作的性能指标。
Network Traffic Rate	指定时间范围内的网络流入和流出速率，按节点物理网络接口统计。
Network Packet Rate (packets/sec)	指定时间范围内的网络包接收和发送速率，按节点物理网络接口统计。

Menu

托管集群

概述

概述

导入集群

概览

导入标准 **Kubernetes** 集群

导入 **OpenShift** 集群

导入 **Amazon EKS** 集群

导入 **GKE** 集群

导入华为云 **CCE** 集群（公有云）

将已有的 CCE（Cloud Container Engine）集群（公有云）导入平台，实现统一管理。

导入 **Azure AKS** 集群

导入阿里云 **ACK** 集群

导入腾讯云 **TKE** 集群

注册集群

注册集群

公有云集群初始化

网络初始化

公有云集群网络初始化。

存储初始化

公有云集群存储初始化。

如何操作

导入集群的网络配置

获取导入集群信息

信任不安全的镜像仓库

从自定义命名的网卡采集网络数据

从自定义命名的网卡采集网络数据

overview

该平台支持管理现有的标准 Kubernetes 集群、OpenShift、Amazon EKS（Elastic Kubernetes Service）以及华为云 CCE（Cloud Container Engine）集群。

目录

什么是托管集群？

托管集群是指将现有集群整合到一个集中式平台中进行统一治理。它允许企业将各种类型的集群——包括标准 Kubernetes 集群和部分公有云集群——纳入单一控制平面。集中管理提升了可扩展性、可用性和可维护性，使计算资源得到更好利用，构建更高效的云环境。您可以通过 **Access a cluster** 或 **Register a cluster** 将集群接入平台。

两种接入方式有什么区别？

它们仅在接入方式上有所不同，日常运维操作保持一致。

- **Import a cluster**：平台首先获取目标集群的信息，然后主动向其发送访问指令。利用这些信息，平台建立稳定连接，实现集中监控和管理，帮助管理员监督环境，确保资源高效且安全地使用。

- **Register a cluster**：在目标集群中部署反向代理，由其主动向平台发起注册请求。集群通过 CLI 自动建立隧道，与平台安全通信。由于无需披露集群详细信息，安全性更高，流程也更简便高效。

Menu

导入集群

概览

导入标准 **Kubernetes** 集群

导入 **OpenShift** 集群

导入 **Amazon EKS** 集群

导入 **GKE** 集群

导入华为云 **CCE** 集群（公有云）

将已有的 CCE（Cloud Container Engine）集群（公有云）导入平台，实现统一管理。

导入 **Azure AKS** 集群

导入阿里云 **ACK** 集群

导入腾讯云 **TKE** 集群

Menu

概览

选择一个提供商，将现有的托管集群连接到平台。

- Standard Kubernetes
- OpenShift
- AWS EKS
- Google GKE
- Azure AKS
- Alibaba Cloud ACK
- Tencent Cloud TKE

导入标准 **Kubernetes** 集群

支持将使用 **kubeadm** 部署的标准原生 Kubernetes 集群接入平台，实现统一管理。

目录

术语

术语	说明
托管 **Kubernetes** 集群	云厂商提供的一种 Kubernetes 集群，Master 节点及其组件由厂商管理，用户无法登录或管理 Master 节点。
非托管 **Kubernetes** 集群	相对而言，部分云厂商提供由用户管理 Master 节点的集群，如阿里云 ACK 专有版或腾讯云 TKE 独立集群。

前提条件

- 集群中的 Kubernetes 及相关组件需满足版本及参数要求。
- 若运行时为 Containerd，集成前需更新 Containerd 配置，以确保分布式存储能成功部署。

注意事项

平台默认监控匹配 `eth.*|en.*|wl.*|ww.*` 的网卡流量，若您的网卡命名规则不同，集成后请根据 [自定义网卡监控] 更新配置。

获取镜像仓库地址

- 若使用平台在 **global cluster** 安装时部署的镜像仓库，在 global 控制节点执行：

```
if [ "$(kubectl get productbase -o
jsonpath='{.items[].spec.registry.preferPlatformURL}')" = 'false' ]; then
    REGISTRY=$(kubectl get cm -n kube-public global-info -o
jsonpath='{.data.registryAddress}')
else
    REGISTRY=$(kubectl get cm -n kube-public global-info -o
jsonpath='{.data.platformURL}' | awk -F // '{print $NF}')
fi
echo "Registry address: $REGISTRY"
```

- 若使用 外部镜像仓库，请手动设置 **REGISTRY**：

```
REGISTRY=<external-registry-address>  # 例如 registry.example.cn:60080 或
192.168.134.43
echo "Registry address: $REGISTRY"
```

检查是否需要额外的镜像仓库配置

1. 执行以下命令检测仓库是否支持带受信任 CA 证书的 HTTPS：

```
REGISTRY=<registry-address-from-previous-step>

if curl -s -o /dev/null --retry 3 --retry-delay 5 -- "https://${REGISTRY}/v2/"; then
    echo '通过：仓库使用受信任的 CA 证书，无需额外配置。'
else
    echo '失败：仓库不支持 HTTPS 或使用不受信任证书，请参考“信任不安全仓库”。'
fi
```

2. 若检测失败，请参见如何信任不安全仓库？

获取集群信息

请参考如何获取集群信息？。

集成集群

1. 在左侧导航中，进入 集群管理 > 集群。

2. 点击 导入集群。

3. 按下表配置参数：

参数	说明
Registry	存储所需平台组件镜像的仓库。选项包括：平台默认（global 安装时配置）、私有仓库（需填写地址、端口、用户名、密码）、公共仓库（需更新云凭证）。
Cluster Info	可手动输入或从 KubeConfig 文件解析。必填字段：集群地址、**CA** 证书（手动输入时需 Base64 解码）、认证信息（token 或具 cluster-admin 权限的客户端证书）。

4. 点击 检测连通性，平台将验证网络访问并自动识别集群类型。

5. 若检测成功，点击 导入 完成操作。

   可通过 执行进度 对话框（status.conditions）查看进度。集成完成后，集群列表中显示为健康状态。

网络配置

确保 global cluster 与导入集群之间的网络连通。

常见问题

为什么“添加节点”按钮是灰色不可用？

无论托管还是非托管集群，平台 UI 均不支持添加节点，请直接通过集群或云厂商方式添加节点。

支持哪些证书？

1. **Kubernetes** 证书：仅支持查看 API Server 证书，其他证书不支持查看且不会自动轮换。
2. 平台组件证书：支持查看且可自动轮换。

哪些功能不支持？

- 托管集群：不支持审计日志。
- 托管集群：不支持 ETCD、Scheduler、Controller Manager 监控（仅支持 API Server 指标）。
- 所有集群：除 API Server 外的证书不支持。

如何解决 **Containerd** 运行时导致分布式存储部署失败？

使用 Containerd 时，若不调整所有节点的 Containerd 配置，分布式存储部署会失败，调整步骤如下：

1. 编辑 `/etc/systemd/system/containerd.service`，设置 `LimitNOFILE=1048576`。
2. 执行 `systemctl daemon-reload`。
3. 重启 Containerd：`systemctl restart containerd`。
4. 在控制节点重启分布式存储 Pod：

```
kubectl delete pod --all -n rook-ceph
```

Import OpenShift Cluster

支持将已部署的 OpenShift 集群接入平台，实现统一管理。

目录

前提条件

- 集群的 Kubernetes 版本和参数必须满足标准 Kubernetes 集群要求。
- 集成过程中需要使用 `kubectl` 命令，请在可访问集群的堡垒机上安装该 CLI 工具。
- 为实现对节点、工作负载（Deployment、StatefulSet、DaemonSet）、Pod 和容器等指标的实时监控，确保目标集群已部署 **Prometheus**。

获取 **Registry** 地址

- 若使用平台在 **global** 集群 安装时部署的 Registry，请在 global 控制节点执行以下命令：

```
if [ "$(kubectl get productbase -o
jsonpath='{.items[].spec.registry.preferPlatformURL}')" = 'false' ]; then
    REGISTRY=$(kubectl get cm -n kube-public global-info -o
jsonpath='{.data.registryAddress}')
else
    REGISTRY=$(kubectl get cm -n kube-public global-info -o
jsonpath='{.data.platformURL}' | awk -F // '{print $NF}')
fi
echo "Registry address is: $REGISTRY"
```

- 若使用 外部 **Registry**，请手动设置 **REGISTRY** 变量：

```
REGISTRY=<external-registry-address>  # 例如 registry.example.cn:60080 或
192.168.134.43
echo "Registry address is: $REGISTRY"
```

检查是否需要额外的 **Registry** 配置

1. 执行以下命令，检查 Registry 是否支持 HTTPS 且使用受信任的 CA 证书：

```
REGISTRY=<registry-address-from-previous-step>

if curl -s -o /dev/null --retry 3 --retry-delay 5 -- "https://${REGISTRY}/v2/"; then
    echo 'Pass: Registry uses a trusted CA certificate. No extra config needed.'
else
    echo 'Fail: Registry does not support HTTPS or uses an untrusted certificate.
Follow "Trust Insecure Registry".'
fi
```

2. 若检查未通过，请按照以下步骤操作。

信任不安全的 **Registry**

1. 登录所有 **OCP** 集群节点。

2. 在每个节点上配置 Registry 设置：

```
sudo -i
sudo chattr -i /

sudo mkdir -p /etc/systemd/system/crio.service.d/
cat | sudo tee /etc/systemd/system/crio.service.d/99-registry-cpaas-system.conf <<
'EOF'
[Service]
ExecStart=
ExecStart=/usr/bin/crio \
          --insecure-registry='<registry-address>' \  # 例如 registry.example.cn:60080
或 192.168.134.43
          $CRIO_CONFIG_OPTIONS \
          $CRIO_RUNTIME_OPTIONS \
          $CRIO_STORAGE_OPTIONS \
          $CRIO_NETWORK_OPTIONS \
          $CRIO_METRICS_OPTIONS
EOF
```

3. 重启 `crio` 服务：

```
sudo systemctl daemon-reload && sudo systemctl restart crio
```

配置集群的 DNS

修改 global 集群中的 CoreDNS `ConfigMap` 以配置 DNS。

1. 在堡垒机上获取 OCP 集群的基础域名：

   ```
   oc get dns cluster -o jsonpath='{.spec.baseDomain}'
   ```

   示例输出：

   ```
   ocp.example.com
   ```

2. 登录平台管理控制台，切换到 **global** 集群，进入 集群管理 > 资源管理。

3. 编辑 `kube-system` 命名空间下的 `cpaas-coredns` `ConfigMap`。

   使用 OCP 基础域名和 DNS 服务器地址（取自集群节点的 `/etc/resolv.conf`）添加新的配置块。

   示例：

   ```
   Corefile: |
   ocp.example.com:1053 {
       log
       forward . 192.168.31.220
   }
   .:1053 {
       log
       forward . 192.168.31.220
   }
   ```

获取集群信息

可选择以下方式之一：

方法 1（推荐）：获取 KubeConfig 文件

1. 在堡垒机上查找 `kubeconfig` 文件，确认包含管理员上下文。

2. 将 kubeconfig 文件从堡垒机复制到本地机器：

```
scp root@<bastion-ip>:</path/to/kubeconfig> <local-path>
```

方法 2：使用 Token、API Server 地址和 CA 证书

参见如何获取集群信息？。

导入集群

1. 在左侧导航中，进入 集群管理 > 集群。

2. 点击 导入集群。

3. 配置参数：

参数	说明
Registry	存储平台组件镜像的 Registry。平台默认：global 安装时配置的 Registry。私有 **Registry**：需填写 Registry 地址、端口、用户名和密码。公共 **Registry**：需更新云端凭据。
集群信息	上传 KubeConfig 文件或手动输入。集群地址：API Server 地址。**CA** 证书：Base64 解码后的 CA 证书。认证方式：token 或具备 cluster-admin 权限的客户端证书。

4. 点击 检测连通性。

5. 若成功，点击 导入。可在执行日志中查看进度。导入完成后，集群状态显示正常。

网络配置

确保 global 集群与导入集群之间的网络连通。详见导入集群的网络配置。

部署插件

集成成功后，进入 **Marketplace** 部署所需插件，如监控、日志采集和日志存储。

部署日志采集前，请确保 `/var/cpaas/` 有超过 50GB 的可用空间：

```
df -h /var/cpaas
```

更新审计策略

可修改集群的审计策略（`spec.audit.profile`）：

- **Default**：记录读写请求的元数据（OAuth 访问令牌创建时记录请求体）。
- **WriteRequestBodies**：记录所有请求的元数据及写请求的请求体。
- **AllRequestBodies**：记录所有请求的元数据和请求体。

敏感资源（如 Secrets、Routes、OAuthClient）仅记录元数据。

更新命令：

```
oc edit apiserver cluster
```

常见问题

为什么“添加节点”按钮是禁用状态？

平台 UI 不支持通过界面添加节点，请使用厂商提供的方法。

支持哪些证书？

1. **Kubernetes** 证书：仅可见 API Server 证书，无自动轮换功能。
2. 平台组件证书：可见且支持自动轮换。

OpenShift 集群不支持哪些功能？

- 审计数据采集。
- ETCD、Scheduler、Controller Manager 监控（仅支持 API Server 指标）。
- API Server 以外的证书管理。

导入 Amazon EKS 集群

将现有的 Amazon EKS（Elastic Kubernetes Service）集群连接到平台，实现统一管理。

目录

前提条件

- 集群的 Kubernetes 版本和设置符合导入标准 Kubernetes 集群的版本兼容性中的要求。
- 镜像仓库必须支持 HTTPS，并提供由公共 CA 签发的有效 TLS 证书。

准备环境

为符合 AWS EKS 的安全规范，请在 AWS CloudShell 中执行以下步骤。

1. 确保网络能够访问 AWS 管理控制台。
2. 搜索 `cloudshell`，然后打开 CloudShell↗。
3. 确认所选地域与目标集群的地域一致，如有需要请切换。
4. CloudShell 准备就绪后，清空终端并运行：

```
# 列出当前地域的集群并验证权限
aws eks list-clusters

# <region-code> 是集群所在地域，例如 us-west-1
# <my-cluster> 是上一步输出的集群名称
aws eks update-kubeconfig --region <region-code> --name <my-cluster>

# kubeconfig 文件保存于 "${HOME}/.kube/config"
# 将其内容保存到文件，然后上传至平台进行解析
cat "${HOME}/.kube/config"
```

5. 环境准备完成。后续如获取集群信息和导入集群等步骤，请在 CloudShell 中针对目标集群执行命令。

获取集群信息

获取导入令牌

来自公有云集群的 KubeConfig 不能直接用于导入。

请参考如何获取集群信息？获取集群导入令牌。

导入集群

1. 在左侧导航中，进入 集群管理 > 集群。

2. 点击 导入集群。

3. 按照以下说明配置参数。

参数	说明
镜像仓库	存储集群所需平台组件镜像的仓库。- 平台默认：global 集群部署时配置的仓库。- 私有仓库：预先准备好的托管所需镜像的仓库，需填写私有仓库地址、端口、用户名和密码。- 公共仓库：公网仓库，使用前请按照更新公共仓库云凭据获取凭据。
集群信息	提示：上传 kubeconfig 文件，平台将自动解析。集群端点：目标集群暴露的外部 API Server 地址。**CA** 证书：集群的 CA 证书。认证方式：使用上一步创建的具有集群管理员权限的令牌。

4. 点击 检查连通性，验证网络连通性并自动检测集群类型。检测结果会以徽章形式显示在表单右上角。

5. 连通性检查通过后，点击 导入，然后确认。

   提示：

   - 对于处于 导入中 状态的集群，点击详情图标可在 执行进度 对话框中查看进度（`status.conditions`）。
   - 导入成功后，集群列表会显示关键信息，集群状态为正常，且可进行集群操作。

网络配置

确保 global 集群与导入集群之间具备网络连通性。详见导入集群的网络配置。

后续步骤

初始化 **Ingress** 和存储

如需 Ingress 和存储功能，请参见 为 AWS EKS 初始化 Ingress 和 为 AWS EKS 初始化存储。

常见问题

导入后“添加节点”按钮不可用，如何添加节点？

平台 UI 不支持添加节点，请通过您的集群提供商添加节点。

导入集群的证书管理支持哪些证书？

1. **Kubernetes** 证书：仅可查看 API Server 证书，其他 Kubernetes 证书不可见且不支持自动轮换。
2. 平台组件证书：在平台中可见，支持自动轮换。

导入的 **AWS EKS** 集群 不支持哪些功能？

- 不提供审计数据。
- 不支持 ETCD、Scheduler 和 Controller Manager 的指标，仅提供部分 API Server 的图表。
- 除 Kubernetes API Server 证书外，不提供其他证书详情。

Import GKE Cluster

平台支持导入 Google GKE 集群。

目录

前提条件

- 集群上的 Kubernetes 版本及组件满足导入公有云集群的版本要求。
- 确保集群类型为标准集群，且账号具有维护控制平面的权限。当前不支持 Autopilot 集群。

- 镜像仓库必须支持 HTTPS 访问，并提供由公有认证机构认证的有效 TLS 证书。

准备操作环境

为符合 GKE 安全规范，以下步骤必须使用 Cloud Shell 执行。

1. 确保与 Google 的网络连通。
2. 访问 Kubernetes Engine 功能中的 **Clusters** 页面↗；找到待导入集群，点击集群详情，选择 **Connect** 按钮。
3. 在弹出对话框中，复制配置 kubectl 命令行访问权限的命令，点击 **Run in Cloud Shell** 按钮。
4. 等待 Cloud Shell 准备完成，清空命令行，粘贴上一步复制的内容并执行。
5. 环境准备完成。后续在导入集群环境中执行的所有命令（如获取集群信息、导入集群等步骤）均应在 Cloud Shell 中执行。

获取集群信息

获取目标集群的 **API Server** 地址和 **CA** 证书

1. 访问 Kubernetes Engine 功能中的 **Clusters** 页面↗，点击进入目标集群详情页。
2. 在 **External endpoints** 区域可查看 API Server 地址。
3. 在 Cloud Shell 中使用以下任一方法获取 CA 证书：

   方法 **A**：从 kubeconfig 中获取 CA 证书：

   ```
   gcloud container clusters get-credentials <cluster-name> --zone <zone>
   kubectl config view --raw -o jsonpath='{.clusters[0].cluster.certificate-authority-data}' | base64 -d
   ```

   方法 **B**：直接从集群获取 CA 证书：

```
gcloud container clusters describe <cluster-name> --zone <zone> --
format='get(masterAuth.clusterCaCertificate)' | base64 -d
```

注意：证书需 Base64 解码后再粘贴至导入表单。

获取目标集群 **Token**

公有云集群的 KubeConfig 文件不能直接用于导入集群。

请参考 FAQ 如何获取集群信息？ 获取目标集群 Token。

导入集群

1. 在左侧导航栏点击 **Clusters** > **Clusters**。
2. 点击 **Manage Cluster** > **Import Cluster**。
3. 按照以下说明配置相关参数。

参数	说明
Image Repository	存储集群所需平台组件镜像的仓库。**-** 平台默认：全局部署时配置的镜像仓库。**-** 私有仓库：预先搭建的存储平台所需组件的仓库。需填写访问镜像仓库的私有镜像仓库地址、端口、用户名和密码。**-** 公共仓库：使用互联网上的公共镜像仓库服务。使用前需先参考更新公共仓库云凭据获取仓库认证权限。
Cluster Information	集群信息：包括目标集群 Token 以及目标集群的 API Server 地址和 CA 证书。集群地址：目标集群对外暴露 API Server 的访问地址，供平台访问集群 API Server。**CA** 证书：目标集群的 CA 证书。注意：手动输入时需填写 Base64 解码后的证书。认证方式：目标集群的认证方式，需使用前面步骤创建的具有集群管理权限的**token**（Token）进行认证。

4. 点击 **Check Connectivity** 验证与目标集群的网络连通性，并自动识别集群类型，识别结果会以徽章形式显示在表单右上角。

5. 连通性检查通过后，点击 **Import** 并确认。

> **TIP**
>
> - 点击处于 **Importing** 状态的集群右侧的 **Details** 图标，可在弹出的 执行进度 对话框中查看集群执行进度（status.conditions）。
> - 集群导入成功后，可在集群列表中查看关键集群信息，集群状态显示正常，且可进行集群相关操作。

网络配置

确保 global 集群与导入集群之间的网络连通。详见导入集群的网络配置。

导入后操作

Ingress 和存储初始化

导入集群后，如需使用 Ingress 和存储相关功能，请参考Google GKE Ingress Controller 配置和Google GKE 存储配置。

常见问题

导入集群后“添加节点”按钮变灰，如何添加节点？

平台界面不支持通过添加节点操作，请联系集群提供方添加节点。

导入集群的证书管理功能支持哪些证书？

1. **Kubernetes** 证书：所有导入集群仅支持在平台证书管理界面查看 APIServer 证书信息，其他 Kubernetes 证书不可查看且不支持自动轮换。

2. 平台组件证书：所有导入集群可在平台证书管理界面查看平台组件证书信息，支持自动轮换。

导入华为云 **CCE** 集群（公有云）

将已有的 CCE（Cloud Container Engine）集群（公有云）导入平台，实现统一管理。

目录

前提条件

- 集群上的 Kubernetes 版本及参数符合标准 Kubernetes 集群组件版本及参数要求。

- 确保集群类型为华为云 CCE 集群，且账号拥有维护控制平面的权限。目前不支持 Turbo 集群。
- 华为云 CCE 集群创建后默认无法访问外部网络资源。导入集群前，请确保待导入集群能够访问平台访问地址。

获取镜像仓库地址

- 若使用来自 global 集群部署的平台部署镜像仓库，请在**global** 集群的控制节点执行以下命令获取地址：

```
if [ "$(kubectl get productbase -o
jsonpath='{.items[].spec.registry.preferPlatformURL}')" = 'false' ]; then
    REGISTRY=$(kubectl get cm -n kube-public global-info -o
jsonpath='{.data.registryAddress}')
else
    REGISTRY=$(kubectl get cm -n kube-public global-info -o
jsonpath='{.data.platformURL}' | awk -F \// '{print $NF}')
fi
echo "Image registry address is: $REGISTRY"
```

- 若使用外部镜像仓库，请手动设置 **REGISTRY** 变量。

```
REGISTRY=<external image registry address>  # 有效示例：registry.example.cn:60080 或
192.168.134.43
echo "Image registry address is: $REGISTRY"
```

判断镜像仓库是否需要额外配置

1. 执行以下命令判断指定镜像仓库是否支持 HTTPS 访问且使用受信任 CA 机构颁发的证书：

```
REGISTRY=<从“获取镜像仓库地址”部分获得的镜像仓库地址>

if curl -s -o /dev/null --retry 3 --retry-delay 5 -- "https://${REGISTRY}/v2/"; then
    echo '测试通过：镜像仓库使用受信任 CA 机构颁发的证书，无需执行“信任不安全镜像仓库”部分内容。'
else
    echo '测试失败：镜像仓库不支持 HTTPS 或证书不受信任，请参考“信任不安全镜像仓库”部分进行配置。'
fi
```

2. 若测试失败，请参考 FAQ 如何信任不安全的镜像仓库？。

获取集群信息

1. 确保与华为云控制台的网络连通。
2. 访问 `Cloud Container Engine CCE` 功能的集群管理页面↗；找到待导入集群，点击集群名称进入详情页。
3. 如下图所示，依次导航找到下载 KubeConfig 文件按钮：`集群信息 - 连接信息 - kubectl - 配置`，下载 KubeConfig 文件。

获取导入集群令牌

公有云集群的 KubeConfig 文件不能直接用于集群导入。

请参考 FAQ 如何获取集群信息？ 获取导入集群令牌。

导入集群

1. 在左侧导航栏点击 集群管理 > 集群。
2. 点击 导入集群。
3. 按照以下说明配置 `Image Registry` 相关参数。

参数	说明
Image Registry	存储集群所需平台组件镜像的仓库。 - 平台默认：global 集群部署时配置的镜像仓库。 - 私有仓库：预构建的仓库存储平台所需组件。需填写访问镜像仓库的私有镜像仓库地址、端口、用户名和密码。 - 公共仓库：使用位于公网的镜像仓库服务。使用前需先参考更新公共镜像仓库云凭证获取仓库认证权限。
集群信息	提示：请上传 KubeConfig 文件，由平台自动解析并填写。 集群地址：导入集群暴露的 API Server 访问地址，用于平台访问导入集群的 API Server。 **CA** 证书：导入集群的 CA 证书。 认证方式：导入集群的认证方式，需使用前面步骤创建的具有集群管理权限的**token**进行认证。

4. 点击 `解析 KubeConfig 文件` 按钮，提交上一步下载的 KubeConfig 文件，平台将自动解析并填写 `集群信息` 相关参数。

5. 点击 检查连通性，检测与导入集群的网络连通性，并自动识别导入集群类型。集群类型会以徽章形式显示在表单右上角。

6. 连通性检测通过后，点击 导入 并确认。

   提示：

   - 点击处于 导入中 状态的集群右侧的
     图标，可在弹出的 执行进度 对话框中查看集群执行进度（status.conditions）。
   - 集群导入成功后，可在集群列表查看集群关键信息。集群状态显示正常，且可进行集群相关操作。

网络配置

为确保 global 集群与导入集群间的网络连通，需参考导入集群网络配置。

后续操作

Ingress（入站规则）及存储初始化

导入集群后，如需使用 Ingress（入站规则）及存储相关功能，请参考华为云 CCE 集群 Ingress 初始化配置和华为云 CCE 集群存储初始化配置。

FAQ

导入集群后，添加节点按钮灰显，如何添加节点？

平台界面不支持添加节点。请联系集群提供方添加节点。

证书管理功能支持导入集群哪些证书？

1. **Kubernetes** 证书：所有导入集群仅支持在平台证书管理界面查看 APIServer 证书信息。不支持查看其他 Kubernetes 证书及自动轮换。
2. 平台组件证书：所有导入集群可在平台证书管理界面查看平台组件证书信息，支持自动轮换。

导入的华为云 CCE 集群还不支持哪些功能？

- 不支持审计数据获取。
- 不支持 ETCD、Scheduler 和 Controller Manager 相关监控信息。支持 APIServer 部分监控图表。
- 不支持获取除 Kubernetes APIServer 证书外的集群证书相关信息。

Import Azure AKS Cluster

将现有的 Azure AKS 集群导入平台，实现统一管理。

目录

前提条件

- 集群上的 Kubernetes 版本和参数必须满足标准 Kubernetes 集群组件版本及参数要求。

> **TIP**
>
> - 如果 AKS 节点无法访问 global 集群，请参考 FAQ：如何配置 AKS 节点外网 IP 安全组规则。

- 镜像仓库必须支持 HTTPS 访问，并提供由公有认证机构认证的有效 TLS 证书。

准备操作环境

为符合 Azure AKS 安全规范，以下步骤需使用 Cloud Shell 执行。

1. 确保与 Azure 控制台的网络连通性。
2. 打开 Kubernetes 服务页面↗，定位要导入的集群，点击进入集群概览页面。
3. 点击 `Connect` 按钮，将弹出标题为 `Connect to <import cluster name>` 的浮动窗口，按照提示打开 Cloud Shell 并配置操作环境。

获取集群信息

获取导入集群的 **Token**

公有云集群的 KubeConfig 文件不能直接用于集群导入。

请参考 FAQ 如何获取集群信息？ 获取导入集群的 Token。

导入集群

1. 在左侧导航栏点击 集群管理 > 集群。
2. 点击 导入集群。

3. 按照以下说明配置相关参数。

参数	说明
镜像仓库	存储集群所需平台组件镜像的仓库。- 平台默认：部署 global 集群时配置的镜像仓库。- 私有仓库：预先搭建的存储平台所需组件镜像的仓库，需要填写访问镜像仓库的私有镜像仓库地址、端口、用户名和密码。- 公共仓库：使用互联网公共镜像仓库服务，使用前需先参考更新公共镜像仓库云端凭据获取仓库认证权限。
集群信息	提示：请上传 KubeConfig 文件，平台将自动解析并填写信息。集群地址：导入集群暴露的 API Server 访问地址，平台通过该地址访问导入集群的 API Server。**CA** 证书：导入集群的 CA 证书。认证方式：导入集群的认证方式，需使用前面步骤创建的具有集群管理权限的**Token**进行认证。

4. 点击 检查连通性，验证与导入集群的网络连通性，并自动识别导入集群类型。集群类型将以徽章形式显示在表单右上角。

5. 连通性检查通过后，点击 导入 并确认。

TIP

- 点击处于 导入中 状态的集群右侧的 详情 图标，可在弹出的 执行进度 对话框中查看集群执行进度（status.conditions）。
- 集群导入成功后，可在集群列表查看集群的关键信息，集群状态显示正常，且可进行集群相关操作。

网络配置

确保 global 集群与导入集群之间的网络连通。详见导入集群的网络配置。

导入后操作

Ingress（入站规则）及存储初始化

导入集群后，如需使用 Ingress（入站规则）和存储相关功能，请参考Azure AKS 集群 Ingress 初始化配置及Azure AKS 集群存储初始化配置。

常见问题

如何配置 AKS 节点外网 IP 安全组规则

节点默认仅有内网 IP，外网 IP 配置在前端负载均衡器（LB）上，默认用于出站流量。该 LB 由 AKS 主体控制，直接手动修改可能导致异常。可通过 **Kubernetes** > 属性 > 基础设施资源组 > 网络安全组 > 添加出站/入站全部规则 来放行流量。

如何访问 AKS 节点

查看 Kubelet、CNI、内核等系统组件日志，需要先 SSH 登录节点。推荐使用 `kubectl-node-shell` 插件，避免为每个节点分配公网 IP。

方案一：使用 **kubectl node-shell**

官方链接↗

方案二：使用 **debug**

官方链接↗

> **NOTE**
>
> 本示例需 kubectl 版本 1.25 及以上，包含 GA 版本的 `kubectl debug` 命令。

```
kubectl debug node/aks-newadd-41368356-vmss000002 -it --
image=mcr.microsoft.com/dotnet/runtime-deps:6.0
chroot /host
```

Azure ALB 使用内部负载均衡器

参考官方链接↗

```yaml
apiVersion: v1
kind: Service
metadata:
  name: internal-app
  namespace: cpaas-system
  annotations:
    service.beta.kubernetes.io/azure-load-balancer-internal: "true"
spec:
  type: LoadBalancer
  ports:
  - name: http-port
    port: 80
    protocol: TCP
  - name: https-port
    port: 443
    protocol: TCP
  selector:
    service.cpaas.io/name: deployment-aks-alb
    service_name: alb2-aks-alb
```

Azure ALB 使用外部负载均衡器

部署高可用 ALB，访问地址配置为外部 LB。

```yaml
apiVersion: v1
kind: Service
metadata:
  name: azure-alb
  namespace: cpaas-system
spec:
  type: LoadBalancer
  ports:
  - name: http-port
    port: 80
    protocol: TCP
  - name: https-port
    port: 443
    protocol: TCP
  - name: prom-port
    port: 11780
    protocol: TCP
  - name: prom2-port
    port: 11781
    protocol: TCP
  - name: prom3-port
    port: 15012
    protocol: TCP
  selector:
    service_name: alb2-cpaas-system
```

若已提前部署，可使用以下命令修改。

```
kubectl edit helmrequest -n cpaas-system uat-cluster-aks-alb
```

导入集群后添加节点按钮变灰，如何添加节点？

平台界面不支持添加节点，请联系集群提供方添加节点。

证书管理功能支持导入集群哪些证书？

1. **Kubernetes** 证书：所有导入集群仅支持在平台证书管理界面查看 APIServer 证书信息，其他 Kubernetes 证书不可查看且不支持自动轮换。

2. 平台组件证书：所有导入集群可在平台证书管理界面查看平台组件证书信息，支持自动轮换。

导入的 **AKS** 集群 还不支持哪些功能？

- 不支持审计数据获取。
- 不支持 ETCD、Scheduler 和 Controller Manager 相关监控信息，仅支持 APIServer 部分监控图表。
- 不支持获取除 Kubernetes APIServer 证书外的集群证书相关信息。

导入阿里云 **ACK** 集群

导入已有的阿里云 ACK 托管集群（Managed Kubernetes）或阿里云 ACK 专属集群（Dedicated Kubernetes），实现统一平台管理。

TIP

有关 ACK 托管集群（Managed Kubernetes）或阿里云 ACK 专属集群（Dedicated Kubernetes）的产品信息，请参见官方文档↗。

目录

前提条件

- 集群上的 Kubernetes 版本和参数需满足导入标准 Kubernetes 集群的组件版本和参数要求。

获取镜像仓库地址

- 若使用 global 集群部署的平台部署镜像仓库，在**global** 集群的控制节点执行以下命令获取地址：

```
if [ "$(kubectl get productbase -o
jsonpath='{.items[].spec.registry.preferPlatformURL}')" = 'false' ]; then
    REGISTRY=$(kubectl get cm -n kube-public global-info -o
jsonpath='{.data.registryAddress}')
else
    REGISTRY=$(kubectl get cm -n kube-public global-info -o
jsonpath='{.data.platformURL}' | awk -F \// '{print $NF}')
fi
echo "镜像仓库地址为: $REGISTRY"
```

- 若使用外部镜像仓库，请手动设置 **REGISTRY** 变量。

```
REGISTRY=<外部镜像仓库地址>  # 有效示例：registry.example.cn:60080 或 192.168.134.43
echo "镜像仓库地址为: $REGISTRY"
```

判断镜像仓库是否需要额外配置

1. 执行以下命令判断指定镜像仓库是否支持 HTTPS 访问且使用受信任 CA 机构颁发的证书：

```
REGISTRY=<从“获取镜像仓库地址”部分获得的镜像仓库地址>

if curl -s -o /dev/null --retry 3 --retry-delay 5 -- "https://${REGISTRY}/v2/"; then
    echo '测试通过：镜像仓库使用受信任 CA 机构颁发的证书，无需执行“信任不安全镜像仓库”部分内容。'
else
    echo '测试失败：镜像仓库不支持 HTTPS 或证书不受信任，请参考“信任不安全镜像仓库”部分进行配置。'
fi
```

2. 若测试失败，请参见常见问题如何信任不安全的镜像仓库？。

获取 KubeConfig

1. 登录阿里云容器服务管理平台。
2. 在控制台左侧导航栏点击 集群。
3. 在集群列表页，点击目标集群名称或目标集群右侧操作列下的详情。
4. 在集群信息页，点击连接信息标签页，然后点击生成临时 **KubeConfig**。
5. 在临时 **KubeConfig**对话框中，设置临时凭证的有效期及访问集群的方式（包括公网访问和内网访问）。
6. 点击生成临时 **KubeConfig**，然后点击复制，将内容保存到本地计算机的 **KubeConfig** 文件中。
7. 集群导入成功后，可以撤销临时凭证。

导入集群

1. 在左侧导航栏点击 集群管理 > 集群。
2. 点击 导入集群。

3. 按照以下说明配置相关参数。

参数	说明
镜像仓库	存储集群所需平台组件镜像的仓库。- 平台默认：global 集群部署时配置的镜像仓库。- 私有仓库：预先搭建的存储平台所需组件镜像的私有仓库。需填写访问镜像仓库的私有镜像仓库地址、端口、用户名和密码。- 公共仓库：使用互联网的公共镜像仓库服务，使用前需参考更新公共仓库云凭证获取仓库认证权限。
集群信息	提示：可手动填写，也可通过上传 KubeConfig 文件由平台自动解析填写。解析 **KubeConfig** 文件：上传获取的 KubeConfig 文件后，平台自动解析并填写集群信息，可修改自动填写的信息。集群地址：集群对外暴露的 API Server 访问地址，平台通过该地址访问集群 API Server。**CA** 证书：集群的 CA 证书。注意：手动输入时需填写 Base64 解码后的证书。认证方式：访问集群的认证方式，需使用具有集群管理权限的**token**或**证书认证（客户端证书和密钥）**进行认证。

4. 点击 检查连通性，检测与待导入集群的网络连通性，并自动识别待导入集群类型。集群类型会以徽章形式显示在表单右上角。

5. 连通性检测通过后，点击 导入 并确认。

TIP

- 点击处于导入中状态集群右侧的详情图标，可在弹出的执行进度对话框中查看集群执行进度（status.conditions）。
- 集群导入成功后，可在集群列表查看集群关键信息，集群状态显示正常，且可进行集群相关操作。

网络配置

确保 global 集群与待导入集群之间的网络连通性。详见导入集群的网络配置。

常见问题

阿里云监控与平台监控组件端口冲突如何处理？

阿里云内置监控与平台监控组件共存时会发生端口冲突，建议卸载阿里云监控，仅保留平台监控。

阿里云集群如何使用公网访问？

若使用阿里云集群的公网访问，可在阿里云绑定公网 IP。

导入集群后添加节点按钮灰显，如何添加节点？

阿里云 **ACK** 托管集群和**ACK** 专属集群均不支持通过平台界面添加节点，请在后台添加或联系集群提供方添加。

导入集群的证书管理功能支持哪些证书？

1. **Kubernetes** 证书：所有导入集群仅支持在平台证书管理界面查看 APIServer 证书信息，不支持查看其他 Kubernetes 证书，也不支持自动轮换。
2. 平台组件证书：所有导入集群可在平台证书管理界面查看平台组件证书信息，并支持自动轮换。

导入的阿里云 **ACK** 托管集群和**ACK** 专属集群还不支持哪些功能？

- 阿里云 **ACK** 托管集群不支持获取审计数据。
- 阿里云 **ACK** 托管集群不支持 ETCD、Scheduler、Controller Manager 相关监控信息，但支持部分 APIServer 监控图表。
- 阿里云 **ACK** 托管集群和**ACK** 专属集群均不支持获取除 Kubernetes APIServer 证书外的集群证书相关信息。

导入腾讯云 TKE 集群

将已有的腾讯云 TKE 专属集群或腾讯云 TKE 托管集群导入平台，实现统一管理。

TIP

关于 TKE 专属集群或腾讯云 TKE 托管集群的产品介绍，请参见官方文档↗。

目录

前提条件

- 集群上的 Kubernetes 版本及参数满足导入标准 Kubernetes 集群的组件版本及参数要求。

- 镜像仓库必须支持 HTTPS 访问，并提供由公有证书机构颁发的有效 TLS 证书。

获取镜像仓库地址

- 若使用平台部署的镜像仓库（即全局集群部署时配置的镜像仓库），请在全局集群的控制节点执行以下命令获取地址：

```
if [ "$(kubectl get productbase -o
jsonpath='{.items[].spec.registry.preferPlatformURL}')" = 'false' ]; then
    REGISTRY=$(kubectl get cm -n kube-public global-info -o
jsonpath='{.data.registryAddress}')
else
    REGISTRY=$(kubectl get cm -n kube-public global-info -o
jsonpath='{.data.platformURL}' | awk -F \// '{print $NF}')
fi
echo "镜像仓库地址为: $REGISTRY"
```

- 若使用外部镜像仓库，请手动设置 **REGISTRY** 变量。

```
REGISTRY=<外部镜像仓库地址>  # 有效示例：registry.example.cn:60080 或 192.168.134.43
echo "镜像仓库地址为: $REGISTRY"
```

判断镜像仓库是否需要额外配置

1. 执行以下命令判断指定的镜像仓库是否支持 HTTPS 访问且使用受信任 CA 颁发的证书：

```
REGISTRY=<从“获取镜像仓库地址”部分获得的镜像仓库地址>

if curl -s -o /dev/null --retry 3 --retry-delay 5 -- "https://${REGISTRY}/v2/"; then
    echo '验证通过：镜像仓库使用受信任 CA 颁发的证书，无需执行“信任不安全镜像仓库”部分内容。'
else
    echo '验证失败：镜像仓库不支持 HTTPS 或证书不受信任，请参考“信任不安全镜像仓库”部分进行配置。'
fi
```

2. 若验证失败，请参见 FAQ 如何信任不安全的镜像仓库？。

获取 **KubeConfig**

1. 登录腾讯云容器服务管理平台。

2. 在 集群详情 > 基本信息 中查看 **Cluster APIServer** 信息。

3. 根据客户实际网络选择 公网访问 或 内网访问，然后下载 **Kubeconfig** 并保存到本地电脑。

导入集群

1. 在左侧导航栏点击 集群管理 > 集群。

2. 点击 导入集群。

3. 按照以下说明配置相关参数。

参数	说明
镜像仓库	存储集群所需平台组件镜像的仓库。- 平台默认：全局部署时配置的镜像仓库。- 私有仓库：预先搭建的存储平台所需组件镜像的私有仓库，需要输入访问镜像仓库的私有镜像仓库地址、端口、用户名和密码。- 公共仓库：使用位于公网的镜像仓库服务，使用前需先参考更新公共仓库云端凭证获取仓库认证权限。
集群信息	提示：可手动填写，也可通过上传 KubeConfig 文件由平台自动解析填写。解析 **KubeConfig** 文件：上传获取的 KubeConfig 文件后，平台会自动解析并填写集群信息，且可修改自动填写的信息。集群地址：集群对外暴露的 API Server 访问地址，平台通过该地址访问集群 API Server。**CA** 证书：集群的 CA 证书。注意：手动输入时需输入 Base64 解码后的证书。认证方式：访问集群的认证方式，需使用具有集群管理权限的**token**（令牌）或证书认证（客户端证书和密钥）。

4. 点击 检查连通性，验证与待导入集群的网络连通性，并自动识别待导入集群的类型。集群类型会以徽章形式显示在表单右上角。

5. 连通性检查通过后，点击 导入 并确认。

   提示：

   - 点击处于 导入中 状态的集群右侧的 详情 图标，可在弹出的 执行进度 对话框中查看集群执行进度（status.conditions）。
   - 集群导入成功后，可在集群列表中查看集群的关键信息，集群状态显示正常 ，且可进行集群相关操作。

网络配置

确保全局集群与待导入集群之间的网络连通性。请参照导入集群的网络配置。

常见问题

导入集群后，“添加节点”按钮置灰，如何添加节点？

TKE 专属集群和**TKE** 托管集群均不支持通过平台界面添加节点，请在后台添加或联系集群提供方添加。

导入集群的证书管理功能支持哪些证书？

1. **Kubernetes** 证书：所有导入集群仅支持在平台证书管理界面查看 APIServer 证书信息，不支持查看其他 Kubernetes 证书，也不支持自动轮换。
2. 平台组件证书：所有导入集群均可在平台证书管理界面查看平台组件证书信息，且支持自动轮换。

导入的 **TKE** 托管集群 和 **TKE** 专属集群 还不支持哪些功能？

- **TKE** 托管集群不支持获取审计数据。
- **TKE** 托管集群不支持 ETCD、Scheduler、Controller Manager 相关监控信息，但支持部分 APIServer 监控图表。
- **TKE** 托管集群和**TKE** 专属集群均不支持获取除 Kubernetes APIServer 证书外的集群证书相关信息。

Register Cluster

这是一种在被管理集群中部署反向代理服务的方法，由被管理集群主动发起注册请求到平台。

目录

前提条件

- 根据被管理集群的类型，被管理集群上 Kubernetes 及其他组件的版本和参数必须满足被管理集群的版本和参数要求。
- 镜像仓库必须支持 HTTPS 访问，并提供由公有证书机构认证的有效 TLS 证书。如无法满足此条件，请参考常见问题如何信任不安全的镜像仓库？

  注意：平台在公网提供的公共仓库已满足 HTTPS 访问要求，您只需确认平台默认和私有仓库是否支持 HTTPS 访问。
- 如果待连接集群的运行时组件为 Containerd，需在连接集群前修改 Containerd 配置，以确保分布式存储部署成功。

重要说明

平台的网卡流量监控默认识别名称匹配 `eth\.\|en\.\|wl\.*\|ww\.*` 的网卡。因此，如果您使用其他命名规则的网卡，请参考从自定义命名网卡采集网络数据文档，在集群连接后修改对应资源，确保平台能正确监控网卡流量。

注册集群

1. 在左侧导航栏点击 **Clusters** > **Clusters**。

2. 点击 **Managed Clusters** > **Register Cluster**。

3. 按照以下说明配置注册集群所需的存储平台组件镜像的镜像仓库参数。

参数	说明
Platform Default	部署全局组件时使用的镜像仓库。
Private Registry	您提前搭建的外部镜像仓库，需要填写访问镜像仓库的私有镜像仓库地址、端口、用户名和密码。
Public Registry	通过平台提供的公共镜像仓库拉取所需镜像，需确保集群能访问公网。使用前需先参考更新公网镜像仓库云端凭据获取仓库认证权限。

4. 点击 **Create**，在注册命令页面获取注册命令，并在待注册集群中运行该命令。

   注意：注册命令有效期为 24 小时，过期后请重新获取。

查看注册命令

您可以在集群列表中找到待注册集群，点击查看注册命令，请在过期时间前完成注册操作。

常见问题

连接集群运行时组件为 **Containerd** 时，如何解决分布式存储部署失败？

当连接集群的运行时组件为 Containerd 时，分布式存储部署会失败。为解决此问题，需手动修改集群所有节点上的 Containerd 配置信息，并重启 Containerd。

注意：如果您在部署分布式存储前已按照以下步骤修改 Containerd 配置，则无需执行第四步。

1. 登录集群节点，编辑 `/etc/systemd/system/containerd.service` 文件，将 `LimitNOFILE` 参数值修改为 `1048576`。

2. 执行命令 `systemctl daemon-reload` 重新加载配置。

3. 执行命令 `systemctl restart containerd` 重启 Containerd。

4. 在集群控制节点执行命令 `kubectl delete pod --all -n rook-ceph`，重启 rook-ceph 命名空间下的所有 Pod 以使配置生效。

Menu

公有云集群初始化

Menu

网络初始化

AWS EKS 集群网络初始化配置

目录

支持概览

功能	支持状态	需求
LoadBalancer Service	支持	可选地部署 AWS Load Balancer Controller。未部署该控制器时，LoadBalancer 功能受限。

功能	支持状态	需求
Ingress	支持	可选地部署 AWS Load Balancer Controller。可选启用 **Ingress Class** 功能（启用后，可在通过表单创建 ingress 时手动选择 ingress class）。

前提条件

- 准备两个带有 **kubernetes.io/role/elb** 标签的子网。对于共享子网，添加 **kubernetes.io/cluster/<cluster-name>: shared** 标签。详见为子网添加标签。
- 如果已创建 EKS 集群，请导入 Amazon EKS 集群。
- 在部署 AWS Load Balancer Controller 之前，确保已安装并可用 kubectl、Helm、AWS CLI 和 eksctl 工具。

  注意：安装工具后，使用创建集群的用户通过 AWS CLI 配置登录信息，并测试 AWS CLI 和 eksctl 工具是否正确安装。
- 事先获取 ACCOUNT_ID、**REGION** 和 **CLUSTER_NAME**，并在文档中将 `<ACCOUNT_ID>`、`<REGION>` 和 `<CLUSTER_NAME>` 替换为实际值。

  注意：**ACCOUNT_ID** 是创建集群用户的账户 ID，**REGION** 是集群所在地域，**CLUSTER_NAME** 是集群名称。
- 更新并验证 Kubeconfig 配置文件。

配置步骤

部署 **AWS Load Balancer Controller**

注意：有关部署 AWS Load Balancer Controller 的详细信息，请参见官方文档↗。

配置 **OIDC Provider**

Kubernetes 集群使用 OpenID Connect (OIDC) 进行身份管理，并关联一个 OIDC 发行者 URL。为启用集群中的 AWS 身份并允许为 Service Account 使用 IAM 角色，需要创建一个与集群 OIDC 发行者 URL 关联的 IAM OIDC Provider。

在 eksctl 中执行以下命令配置 OIDC Provider：

```
eksctl utils associate-iam-oidc-provider --region=<REGION> --cluster=<CLUSTER_NAME> --
approve
```

配置 **Service Account**

执行以下命令创建 IAM 策略，并创建名为 `aws-load-balancer-controller` 的 Service Account，将其与 IAM 角色关联：

```
curl -o aws-load-balancer-controller-iam-policy.json
https://raw.githubusercontent.com/kubernetes-sigs/aws-load-balancer-
controller/v2.4.7/docs/install/iam_policy.json
aws iam create-policy \
    --policy-name <CLUSTER_NAME>-AWSLoadBalancerControllerIAMPolicy \
    --policy-document file://aws-load-balancer-controller-iam-policy.json

eksctl create iamserviceaccount \
  --cluster=<CLUSTER_NAME> \
  --namespace=kube-system \
  --name=aws-load-balancer-controller \
  --role-name AmazonEKSLoadBalancerControllerRole \
  --attach-policy-arn=arn:aws:iam::<ACCOUNT_ID>:policy/<CLUSTER_NAME>-
AWSLoadBalancerControllerIAMPolicy \
  --approve
```

将 **AWS Load Balancer Controller** 部署到集群

在 eksctl 中执行以下命令部署 AWS Load Balancer Controller：

1. 添加 eks-charts 仓库：

   ```
   helm repo add eks https://aws.github.io/eks-charts
   ```

2. 更新本地仓库：

```
helm repo update eks
```

3. 将 AWS Load Balancer Controller Helm Chart 部署到集群：

   注意：`aws-load-balancer-controller` 是在配置 Service Account中创建的 **Service Account**。

```
helm install aws-load-balancer-controller eks/aws-load-balancer-controller \
  -n kube-system \
  --version=v2.4.7 \
  --set ingressClassConfig.default=true \
  --set clusterName=<CLUSTER_NAME> \
  --set serviceAccount.create=false \
  --set serviceAccount.name=aws-load-balancer-controller
```

创建 Ingress 和 LoadBalancer 服务

您可以同时创建 ingress 和 LoadBalancer 服务，也可以根据需求选择其中一种。

创建 **Ingress**

1. 在 容器平台，点击左侧导航的 网络 > **Ingress**。

2. 点击 创建 **Ingress**，并在 **Ingress Class** 中选择 **EKS Ingress Class**。

3. 选择 协议。默认是 **HTTP**。若选择 **HTTPS**，请先创建证书并选择该证书。

4. 切换到 **YAML**，添加以下注解。详情见注解文档↗：

```
alb.ingress.kubernetes.io/scheme: internet-facing ## 指定公网访问
alb.ingress.kubernetes.io/target-type: ip ## 流量直接路由到 Pod
```

5. 点击 创建。

创建 **LoadBalancer** 服务

1. 在 容器平台，点击左侧导航的 网络 > 服务。

2. 点击 创建服务，在 外部访问 中选择 **LoadBalancer**。

3. 展开 **annotations**，根据需要填写LoadBalancer 服务注解。

4. 点击 创建。

相关操作

测试 AWS CLI 和 eksctl 安装

- 执行以下命令，若返回集群列表，则 AWS CLI 安装正确：

```
aws eks list-clusters
```

- 执行以下命令，若返回集群列表，则 eksctl 安装正确：

```
eksctl get clusters
```

获取 ACCOUNT_ID

执行 `aws sts get-caller-identity` 获取 **ACCOUNT_ID**。响应中的 `651168850570` 即为 **ACCOUNT_ID**：

```
{
  "ARN": "arn:aws:iam::651168850570:user/jwshi"
}
```

Kubeconfig 配置文件

1. 执行以下命令更新指定地域的 Kubeconfig 文件：

```
aws eks --region <REGION>  update-kubeconfig --name <CLUSTER_NAME>
```

2. 执行以下命令验证 Kubeconfig 文件，若正常返回信息，则配置正确：

```
kubectl get svc -n cpaas-system
```

为子网添加标签

1. 执行以下命令获取集群子网：

```
eksctl get cluster --name <CLUSTER_NAME>
```

2. 执行以下命令获取子网详情：

```
aws ec2 describe-subnets
```

3. 执行以下命令为子网添加标签。将 `<subnet-id>` 替换为实际值。详见子网自动发现↗：

- 为子网添加 `kubernetes.io/role/elb` 标签：

```
aws ec2 create-tags --resources <subnet-id> --tags
Key=kubernetes.io/role/elb,Value="1"
```

- 为共享子网添加 `kubernetes.io/cluster/<CLUSTER_NAME>: shared` 标签：

```
aws ec2 create-tags --resources <subnet-id> --tags
Key=kubernetes.io/cluster/<CLUSTER_NAME>,Value="shared"
```

创建证书

使用 HTTPS 协议时，需提前将 HTTPS 证书凭据以 Secret（TLS 类型）形式保存。

1. 在 容器平台，点击左侧导航的 配置 > **Secrets**。

2. 点击 创建 **Secret**。

3. 选择 **TLS** 类型，按需导入或填写 证书 和 私钥。

4. 点击 创建。

AWS EKS 补充信息

目录

术语

缩写	全称	说明
eks-clb	Classic Load Balancer	AWS 默认负载均衡器。在某些情况下存在问题，不推荐使用。
eks-nlb	Network Load Balancer	AWS 第4层负载均衡器，在 TCP/UDP 层进行负载均衡，适用于需要更高层网络控制的场景。
eks-alb	Application Load Balancer	AWS 第7层负载均衡器。相比 eks-nlb，eks-alb 能解析 HTTP/HTTPS 协议并更智能地分发请求，适合 Web 应用。
aws-lb	AWS Load Balancer	安装在 Kubernetes 上的负载均衡器，能根据 Kubernetes 中的 LoadBalancer 类型 Service 和

缩写	全称	说明
		Ingress 自动创建 eks-nlb 和 eks-alb，以满足应用负载均衡需求。
Platform Load Balancer	-	平台自研的第7层负载均衡器。
Service Annotations	-	以键值对形式附加在对象上的元数据。这些附加信息可以被识别和利用，用于增强和简化 Kubernetes 资源的各方面管理。Annotations 可以是无具体功能的说明性文本，也可以指定云厂商配置或行为，或指定配置参数和工具，功能非常强大。

重要说明

创建负载均衡器时，建议手动配置 Service Annotations，确保平台负载均衡器正确使用 aws-lb。如果未正确配置相应的 Service Annotations，平台将默认使用 eks-clb，而 eks-clb 存在 UDP 相关问题，可能导致异常情况。

EKS 使用 aws-lb 为容器网络负载均衡器提供外部访问

Service Annotation 配置说明

1. 在对应集群中，使用 kubectl 执行以下命令，查找 kube-system 命名空间中名称包含 "aws-load" 的所有 Pod：

   ```
   kubectl get pod -n kube-system |grep aws-load
   ```

2. 创建负载均衡器；详细创建步骤和参数请参见 AWS EKS Service Annotation Instructions 中的负载均衡器创建部分。

- 如果上述命令未返回相关 Pod，说明集群未安装 AWS Load Balancer Controller，无需配置 Service Annotations，直接创建负载均衡器即可。

- 如果上述命令返回相关 Pod，说明集群已安装 AWS Load Balancer Controller。创建负载均衡器时，在对应集群中需添加以下 Service Annotations。注释详情请参见 AWS EKS Service Annotation Instructions：

  - `service.beta.kubernetes.io/aws-load-balancer-type: external //必填`
  - `service.beta.kubernetes.io/aws-load-balancer-nlb-target-type: ip //必填`
  - `service.beta.kubernetes.io/aws-load-balancer-scheme: internet-facing // 可选，若需要公网支持则添加此注释。`

访问地址获取方式

- 创建容器网络类型负载均衡器时，填写的 Service Annotations 会设置在对应平台负载均衡器的 LoadBalancer 类型 Service 上。

- 在公有云环境中，带有相应 Service Annotations 的 LoadBalancer 类型 Service 会被公有云识别并分配地址。平台负载均衡器会读取该地址，并将其设置为自身的访问地址。

华为云 CCE 集群网络初始化配置

目录

Support Overview

Feature	Support Status	Requirements
LoadBalancer Service	Default Support	无需额外部署。
Ingress	Default Support	可选启用 **Ingress Class** 功能（启用后，可通过表单界面创建 ingress 时手动选择 ingress 类）。无需额外部署。

Prerequisites

如果您已创建 CCE 集群，请导入 CCE 集群（公有云）。

Configuration Steps

您可以同时创建 ingress 和 LoadBalancer 服务，也可以根据需求选择其中一种。

Create Ingress

创建 ingress 有两种方式。推荐使用 方式一：手动选择 **Ingress Class**。

注意：避免创建两个具有相同 **path** 的 ingress 资源。

（推荐）方式一：手动选择 **Ingress Class**

1. 在 **Container Platform** 左侧导航中点击 **Network** > **Ingress**。
2. 点击 **Create Ingress**，在 **Ingress Class** 中选择 **CCE Ingress Class**。
3. 选择 **Protocol**，默认是 **HTTP**。若选择 **HTTPS**，请先创建证书并选择该证书。
4. 切换到 **YAML**，根据您的默认 Ingress Controller 类型添加以下注解。注解详情请参见使用注解配置负载均衡器↗：

   注意：请将下表注解中的 **values** 替换为实际环境值。

默认 **Ingress Controller** 类型	注解
Shared（自动创建）	`kubernetes.io/elb.autocreate: '{"type":"public","bandwidth_name":"{random}","bandwidth_chargemode":"traffic","bandwidth_size":5,"bandwidth_sh` `kubernetes.io/elb.class: union`

默认 **Ingress Controller** 类型	注解

Shared（复用）	`kubernetes.io/elb.class: union`
	`kubernetes.io/elb.id: <Load Balancer Instance ID>`
	`kubernetes.io/elb.port: '80'`

Dedicated（自动创建）	`kubernetes.io/elb.autocreate: '{"type":"public","bandwidth_name":"<Bandwi`
	`Name>","bandwidth_chargemode":"traffic","bandwidth_size":5,"bandwidth_share`
	`["<AZ A>","<AZ B>","<AZ C>"],"elb_virsubnet_ids":["<ELB Virtual Subnet`
	`ID>"],"l7_flavor_name":"L7_flavor.elb.s1.small","l4_flavor_name":"L4_flavo`
	`kubernetes.io/elb.class: performance`
	`kubernetes.io/elb.port: "80"`

Dedicated（复用）	`kubernetes.io/elb.class: performance`
	`kubernetes.io/elb.id: <Load Balancer Instance ID>`
	`kubernetes.io/elb.port: "80"`

5. 点击 **Create**。创建完成后，即可通过 ELB 访问集群服务。

方式二：使用默认 **Ingress Class**

1. 创建一个 IngressClass YAML 文件，内容如下。详情请参见默认 Ingress Class↗：

```yaml
apiVersion: networking.k8s.io/v1
kind: IngressClass
metadata:
  annotations:
    ingressclass.kubernetes.io/is-default-class: "true"
  name: cce
spec:
  controller: alauda/cce
```

2. 保存文件并应用到导入的集群，替换 `<filename.yaml>` 为实际 YAML 文件名：

```
kubectl apply -f <filename.yaml>
```

3. 在 **Container Platform** 左侧导航中点击 **Network** > **Ingress**。

4. 选择 **Protocol**，默认是 **HTTP**。若选择 **HTTPS**，请先创建证书并选择该证书。

5. 点击 **Create**。创建完成后，即可通过 ELB 访问集群服务。

Create LoadBalancer Service

1. 在 **Container Platform** 左侧导航中点击 **Network** > **Services**。

2. 点击 **Create Service**，在 **External Access** 中选择 **LoadBalancer**。

3. 展开 **annotations**，根据需要填写 LoadBalancer 服务注解。

4. 点击 **Create**。

Related Operations

Create Certificate

使用 HTTPS 协议时，请提前将 HTTPS 证书凭据以 Secret（TLS 类型）形式保存。

1. 在 **Container Platform** 左侧导航中点击 **Configuration** > **Secrets**。

2. 点击 **Create Secret**。

3. 选择 **TLS** 类型，导入或填写 **Certificate** 和 **Private Key**。

4. 点击 **Create**。

Azure AKS 集群网络初始化配置

目录

支持概览

功能	支持状态	需求
LoadBalancer Service	默认支持	无需额外部署。
Ingress	支持	可选 部署 Ingress Controller。可选启用 **Ingress Class** 功能（启用后，可在通过表单界面创建 ingress 时手动选择 ingress 类）。

前提条件

如果您已创建 AKS 集群，导入 Azure AKS 集群。

配置步骤

部署 **Ingress Controller**

AKS 使用 容器网络模式，并利用 **Nginx Ingress Controller** 管理负载均衡器，同时通过 **LoadBalancer** 类型的 **Services** 为容器内部网络中的虚拟 IP 地址（VIP）提供外部访问地址。

1. 登录 Microsoft Azure 并进入您创建的 AKS 集群。
2. 在左侧导航中，点击 **Kubernetes** 资源 > 服务和 **Ingresses**。
3. 点击 创建，从下拉菜单中选择 **Ingress (Preview)**，系统将提示并自动创建 Ingress Controller。
4. 点击 启用 并等待完成。

创建 **Ingress** 和 **LoadBalancer** 服务

您可以同时创建 ingress 和 LoadBalancer 服务，或根据需求选择其中之一。

创建 **Ingress**

1. 在 容器平台，左侧导航点击 网络 > **Ingress**。
2. 点击 创建 **Ingress**，并为 **Ingress Class** 选择 **webapprouting.kubernetes.azure.com**。
3. 选择 协议。默认是 **HTTP**。若选择 **HTTPS**，请先创建证书并选择该证书。
4. 点击 创建。

创建 **LoadBalancer** 服务

1. 在 容器平台，左侧导航点击 网络 > 服务。

2. 点击 创建服务，并为 外部访问 选择 **LoadBalancer**。

3. 展开 **annotations**，根据需要填写 LoadBalancer 服务注解。

4. 点击 创建。

相关操作

创建证书

使用 HTTPS 协议时，请提前将 HTTPS 证书凭据以 Secret（TLS 类型）形式保存。

1. 在 容器平台，左侧导航点击 配置 > **Secrets**。

2. 点击 创建 **Secret**。

3. 选择 **TLS** 类型，按需导入或填写 证书 和 私钥。

4. 点击 创建。

Google GKE 集群网络初始化配置

目录

支持概览

功能	支持状态	需求
LoadBalancer Service	默认支持	无需额外部署。
Ingress	默认支持	可选启用 **Ingress Class** 功能（启用后，可通过表单界面创建 ingress 时手动选择 ingress 类）。无需额外部署。

前提条件

如果您已创建 GKE 集群，请导入 GKE 集群。

配置步骤

部署 **Ingress Controller**

无需手动部署。GKE 提供了一个托管的内置 Ingress controller，称为 GKE Ingress。该 controller 将 Ingress 资源映射到 Google Cloud 负载均衡器，用于处理 GKE 中的 HTTP(S) 工作负载，使配置更简单且更自动化。

创建 **Ingress** 和 **LoadBalancer** 服务

您可以根据需求同时创建 ingress 和 LoadBalancer 服务，或选择其中之一。

创建 **Ingress**

1. 在 容器平台，点击左侧导航的 网络 > **Ingress**。
2. 点击 创建 **Ingress**，并为 **Ingress Class** 选择 **GKE Ingress Class**。
3. 选择 协议。默认是 **HTTP**。若选择 **HTTPS**，请先创建证书并选择该证书。
4. 点击 创建。等待约 5 分钟，GKE 平台会自动为 ingress 分配公网 IP 地址。

   注意：不同的 ingress 资源会分配不同的公网 IP 地址。

创建 **LoadBalancer** 服务

1. 在 容器平台，点击左侧导航的 网络 > 服务。
2. 点击 创建服务，并为 外部访问 选择 **LoadBalancer**。
3. 展开 **annotations**，根据需要填写 LoadBalancer 服务注解。
4. 点击 创建。

相关操作

在 **Google Cloud** 中查看 **Ingress** 资源

1. 进入 **Google Cloud > Kubernetes Engine**，点击左侧导航的 服务和 **Ingress**。

2. 点击 **INGRESS**。

3. 在列表中查看对应 Ingress 资源的信息。

创建证书

使用 HTTPS 协议时，请提前将 HTTPS 证书凭据以 Secret（TLS 类型）形式保存。

1. 在 容器平台，点击左侧导航的 配置 > **Secrets**。

2. 点击 创建 **Secret**。

3. 选择 **TLS** 类型，按需导入或填写 证书 和 私钥。

4. 点击 创建。

Menu

存储初始化

概览

- Amazon Elastic Kubernetes Service (Amazon EKS) 是 Amazon 提供的托管 Kubernetes 服务，用于在 AWS Cloud 和本地数据中心运行 Kubernetes。在云端，Amazon EKS 自动管理负责调度容器、管理应用可用性、存储集群数据及其他关键任务的 Kubernetes 控制平面节点的可用性和可扩展性，提供一致且完全支持的 Kubernetes 解决方案。
- Huawei Cloud Container Engine (CCE) 提供高可靠性、高性能的企业级容器应用管理服务，支持 Kubernetes community 原生应用和工具，简化云上自动化容器运行环境的构建。
- Azure Kubernetes Service (AKS) 提供在 Azure、数据中心或边缘快速开始开发和部署云原生应用的最快方式，内置从代码到云的流水线和护栏，实现对本地、边缘和多云 Kubernetes 集群的统一管理和治理。
- Google Kubernetes Engine (GKE) 提供极具可扩展性、全自动的 Kubernetes 服务，几乎无需 Kubernetes 专业知识即可使用。其优势包括更快的速度、更低的风险和更低的总体拥有成本，内置安全态势和可观测性工具，以及业界领先的自动扩缩容解决方案，支持扩展至 15,000 个节点。

目录

存储类支持

AWS EKS 集群

存储类型	默认存储类	创建带 **RWO** 访问模式的 **PVC**	创建带 **RWX** 访问模式的 **PVC**	**PVC** 扩容	**PVC** 快照
文件存储	efs-sc	支持	支持	不支持	不支持
块存储	ebs-sc	支持	不支持	支持	不支持

Huawei Cloud CCE 集群

存储类型	默认存储类	创建带 **RWO** 访问模式的 **PVC**	创建带 **RWX** 访问模式的 **PVC**	**PVC** 扩容	**PVC** 快照
文件存储	csi-nas	不支持	支持	支持	不支持
块存储	csi-disk	支持	不支持	支持	不支持

Azure AKS 集群

存储类型	默认存储类	创建带 **RWO** 访问模式的 **PVC**	创建带 **RWX** 访问模式的 **PVC**	**PVC** 扩容	**PVC** 快照
文件存储	azurefile	支持	支持	支持	不支持
块存储	default	支持	不支持	支持	不支持

Google GKE 集群

存储类型	默认存储类	创建带 **RWO** 访问模式的 **PVC**	创建带 **RWX** 访问模式的 **PVC**	**PVC** 扩容	**PVC** 快照
文件存储	standard-rwx	支持	支持	支持	不支持
块存储	standard-rwo	支持	不支持	支持	不支持

AWS EKS 集群存储初始化配置

平台与 AWS EKS 的集成及存储初始化配置。

目录

约束与限制

- 默认的 efs-sc 文件存储类在挂载后可能不支持权限修改，可能导致 PostgreSQL、Jenkins 等部分应用无法正常运行。
- AL2023 AMI 不支持 A1 系列实例，导致 EBS 块存储插件（Amazon EBS CSI Driver）无法正常部署。EBS CSI 驱动已支持 GA 多架构/ARM，因此限制在于 AMI/实例支持，而非驱动本身。如果需要使用 EBS 块存储类，请避免使用以下实例类型，建议使用 Graviton2/3 替代方案：
  - a1.medium

- a1.large
- a1.xlarge
- a1.2xlarge
- a1.4xlarge

推荐替代方案：使用 Graviton2/3 实例系列，如 m6g、c6g、r6g、t4g 等，提供更优性能及完整的 EBS CSI 驱动支持。

前提条件

- 确保已安装并可用 kubectl↗ 和 AWS CLI 工具。
- 若已创建 EKS 集群，请导入 Amazon EKS 集群；若未创建，则需创建 AWS EKS 集群。
- 在 EKS 集群中部署 EFS 文件存储插件 **Amazon EFS CSI Driver** 和 EBS 块存储插件 **Amazon EBS CSI Driver**。

  注意：使用 EFS 文件存储时，请在 EKS 所在地域创建文件存储，并记录 文件系统 **ID**。

配置步骤

创建存储类

1. 进入 平台管理，点击左侧导航的 存储管理 > 存储类。
2. 点击 创建存储类 旁的下拉菜单 > 从 **YAML** 创建。
3. 在 YAML 文件中添加以下内容，根据需要创建默认存储类。默认文件存储类名称为 **efs-sc**，默认块存储类名称为 **ebs-sc**。

   - EFS 文件存储

     注意：将 `<File System ID>` 替换为实际的 文件系统 **ID**，例如 `fileSystemId: fs-05aef9e1edd309f2b`。

```
kind: StorageClass
apiVersion: storage.k8s.io/v1
metadata:
  name: efs-sc
provisioner: efs.csi.aws.com
parameters:
  provisioningMode: efs-ap
  fileSystemId: <File System ID>
  directoryPerms: "755"
```

- EBS 块存储

```
allowVolumeExpansion: true
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: ebs-sc
provisioner: ebs.csi.aws.com
reclaimPolicy: Delete
volumeBindingMode: WaitForFirstConsumer
```

4. 点击 创建。

注意：若默认存储类不满足需求，可按上述步骤创建新的存储类并根据需要修改参数。详见可用存储类参数配置。

修改存储类项目分配

1. 在左侧导航点击 存储管理 > 存储类。

2. 点击名为 **efs-sc** 或 **ebs-sc** 的存储类旁的三点按钮 > 更新项目。

3. 根据需要选择 项目分配 方式，点击 更新，将存储类分配给项目。

相关操作

配置可用存储类参数

- EFS 文件存储可用参数

参数	可选值	默认值	可选	描述
az		""	是	用于跨账户挂载。若指定，则使用与 az 关联的挂载目标进行跨账户挂载；若不指定，则随机选择挂载目标进行跨账户挂载。
basePath			是	动态创建访问点时的路径。若不指定，访问点将创建在文件系统根目录下。
directoryPerms			否	创建 访问点根目录↗ 的目录权限。
uid			是	创建 访问点根目录↗ 的 POSIX 用户 ID。
gid			是	创建 访问点根目录↗ 的 POSIX 组 ID。
gidRangeStart		50000	是	创建 访问点根目录↗ 时应用的 POSIX 组 ID 起始范围。若已设置 uid/gid，则无需设置。
gidRangeEnd		7000000	是	POSIX 组 ID 结束范围。若已设置 uid/gid，则无需设置。
subPathPattern			是	构造动态创建访问点所在子路径的模板。可由固定字符串和有限变量组成，类似于 nfs-subdir-external-provisioner chart 中的 "subPathPattern" 变量。可选参数包括

参数	可选值	默认值	可选	描述
				.PVC.name、.PVC.namespace 和 .PV.name。
ensureUniqueDirectory		true	是	动态创建启用时使用。设置为 true 时，会在 subPathPattern 指定的模式后追加 UID，确保访问点不会意外指向同一目录。注意：仅在确定需要此行为时才设置为 false。
provisioningMode	efs-ap		否	EFS 卷类型，目前支持访问点。
fileSystemId			否	创建访问点的文件系统 ID。

- EBS 块存储可用参数

  注意：不同卷类型的性能参数详见 Amazon EBS 卷类型↗。

参数	可选值	默认值	描述
"allowAutoIOPSPerGBIncrease"	true, false	false	设置为 "true" 时，当 iopsPerGB * <卷大小> 过低，无法满足 AWS 支持的 IOPS 范围时，CSI 驱动会自动增加卷的 IOPS，确保动态创建始终成功，即使用户指定的 PVC 容量或 iopsPerGB 值过小，但可能产生额外费用，因为此类卷的 IOPS 高于 iopsPerGB 要求。

参数	可选值	默认值	描述
"blockExpress"	true, false	false	通过将 io2 卷的 IOPS 限制提升至 256000，创建 io2 Block Express 卷，但创建 IOPS 超过 64000 的卷无法挂载到不支持 io2 Block Express 的实例上。
"blockSize"			格式化底层文件系统时使用的块大小。仅适用于 Linux 节点且文件系统类型为 ext2、ext3、ext4 或 xfs。
"bytesPerINode"			格式化底层文件系统时每个 inode 的字节数。仅适用于 Linux 节点且文件系统类型为 ext2、ext3 或 ext4。
"csi.storage.k8s.io/fstype"	xfs, ext2, ext3, ext4	ext4	创建卷时格式化的文件系统类型，区分大小写。
"encrypted"	true, false	false	是否需要对卷进行加密。
"inodeSize"			格式化底层文件系统时使用的 inode 大小。仅适用于 Linux 节点且文件系统类型为 ext2、ext3、ext4 或 xfs。inode 是文件系统中存储文件和目录元数据的数据结构。

参数	可选值	默认值	描述
"iops"			每秒 I/O 操作数，适用于 IO1、IO2 和 GP3 卷。
"iopsPerGB"			每 GiB 每秒 I/O 操作数，适用于 IO1、IO2 和 GP3 卷。
"kmsKeyId"			用于加密卷的密钥完整 ARN。若未指定，AWS 会使用卷所在区域的默认 KMS 密钥，并自动生成名为 /aws/ebs 的密钥。
"numberOfINodes"			格式化底层文件系统时指定的 inode 数量。仅适用于 Linux 节点且文件系统类型为 ext2、ext3 或 ext4。
"throughput"		125	吞吐量，单位 MiB/s。仅在指定 gp3 卷类型时有效。若为空，默认为 125 MiB/s。详见 Amazon EBS 卷类型↗ 。
"type"	io1, io2, gp2, gp3, sc1, st1, standard, sbp1, sbg1	gp3	EBS 卷类型。

华为云 CCE 集群存储初始化配置

平台与华为云 CCE 的集成及存储初始化配置。

目录

约束与限制

集群 PVC 数量有限制，且账户存储容量有配额。您可以通过工单申请提升。

前提条件

- 如果您已创建 CCE 集群，请导入 CCE 集群（公有云）。

配置步骤

1. 进入 平台管理，点击左侧导航中的 存储管理 > 存储类。
2. 点击名为 **csi-nas** 或 **csi-disk** 的存储类旁的三点 > 更新项目。

   注意：导入 CCE 集群后，会生成默认存储类。**csi-disk** 推荐用于块存储，**csi-nas** 推荐用于文件存储。详见默认存储类说明。
3. 根据需要选择 项目分配 方式，点击 更新，将 **csi-nas** 或 **csi-disk** 存储类分配给项目。

默认存储类说明

存储类名称	存储类类型	说明
（推荐）**csi-disk**	块存储	推荐使用。
（推荐）**csi-nas**	文件存储	推荐使用，仅在支持 csi-nas 服务的区域可用。
csi-disk-topology	块存储	节点跨可用区时自动云盘拓扑。
csi-local	本地存储	
csi-local-topology	本地存储	
csi-obs	对象存储	
csi-sfsturbo	超高速文件存储	超高速文件存储无法动态创建持久卷。

常见问题

PVC 创建失败

出现以下错误是因为达到 PVC 数量限制。您可以通过工单申请提升：

```
message: "ShareLimitExceeded: Maximum number of shares allowed (10) exceeded."
```

账户欠费

由于欠费导致 PVC 创建失败，出现以下错误，请及时缴费：

```
message: "Your account is suspended and resources cannot be used"
```

Azure AKS 集群存储初始化配置

平台与 Azure AKS 的集成及存储初始化配置。

目录

约束与限制

默认的 azurefile 文件存储类可能不支持挂载后修改权限，这可能导致某些应用如 PostgreSQL 和 Jenkins 无法正常运行。

前提条件

- 如果您已创建 AKS 集群，请导入 Azure AKS 集群。

配置步骤

1. 进入 平台管理，点击左侧导航中的 存储管理 > 存储类。
2. 点击名为 **default** 或 **azurefile** 的存储类旁的三个点 > 更新项目。

   注意：导入 AKS 集群后，会生成以下默认存储类。推荐使用 **default** 作为块存储，**azurefile** 作为文件存储。详见默认存储类说明。
3. 根据需要选择 项目分配 方式，点击 更新，将 **default** 或 **azurefile** 存储类分配给项目。

   注意：如果默认存储类不满足需求，可按照上述步骤创建新的存储类并根据需要修改参数。详见可用存储类参数。

相关信息

默认存储类说明

存储类名称	存储类类型	说明
（推荐）**azurefile**	文件存储	使用 Azure 标准存储创建 Azure 文件共享。
（推荐）**default**	块存储	使用 Azure StandardSSD 存储创建托管磁盘。
azurefile-csi	文件存储	使用 Azure 标准存储创建 Azure 文件共享。
azurefile-csi-nfs	文件存储	使用 Azure 标准存储创建 Azure 文件共享，支持 NFS 协议。
azurefile-csi-premium	文件存储	使用 Azure 高级存储创建 Azure 文件共享。
azurefile-premium	文件存储	使用 Azure 高级存储创建 Azure 文件共享。

存储类名称	存储类类型	说明
managed	块存储	使用 Azure StandardSSD 存储创建托管磁盘。
managed-csi	块存储	使用 Azure StandardSSD 本地冗余存储（LRS）创建托管磁盘。
managed-csi-premium	块存储	使用 Azure 高级本地冗余存储（LRS）创建托管磁盘。
managed-premium	块存储	使用 Azure 高级存储创建托管磁盘。

可用存储类参数

- 有关块存储可选参数及含义，请参见在 Azure Kubernetes Service (AKS) 中使用 Azure 磁盘创建和使用卷↗。
- 有关文件存储可选参数及含义，请参见在 Azure Kubernetes Service (AKS) 中使用 Azure 文件创建和使用卷↗。

Google GKE 集群存储初始化配置

平台与 Google GKE 的集成及存储初始化配置。

目录

约束与限制

- 默认文件存储类型 PVC 最小容量为 1T。创建时如果容量设置小于 1T，会自动扩展到 1T。
- 默认文件存储存在容量限制，可通过工单申请扩容。
- 默认文件存储的创建和删除操作耗时较长，如长时间处于创建中状态，请耐心等待。

前提条件

- 创建集群时，在 Google Cloud Platform **Cluster > Features** 页面 **Other** 区域，勾选 **Enable Compute Engine Persistent Disk CSI Driver** 和 **Enable Filestore CSI Driver** 选项。
- 在 Google Cloud Platform 上启用 **Cloud Filestore API** 和 **Google Kubernetes Engine API**。详见 使用 Filestore CSI 驱动访问 Filestore 实例↗。
- 在 Google Cloud Platform 上调整区域文件存储配额。详见 资源配额和限制↗。
- 如果已创建 GKE 集群，请导入 GKE 集群。

配置步骤

1. 进入 平台管理，点击左侧导航的 存储管理 > 存储类。
2. 点击名为 **standard-rwx** 或 **standard-rwo** 的存储类旁的三点按钮 > 更新项目。

   注意：导入 GKE 集群后会生成默认存储类。推荐文件存储使用 **standard-rwx**，块存储使用 **standard-rwo**。详见 默认存储类说明。
3. 根据需要选择 项目分配 方式，点击 更新，将 **standard-rwx** 或 **standard-rwo** 存储类分配给项目。

   注意：若默认存储类不满足需求，可按上述步骤创建新的存储类并根据需要修改参数。详见 可用存储类参数。

相关信息

默认存储类说明

存储类名称	存储类类型	说明
（推荐）**standard-rwx**	文件存储	使用 Basic HDD Filestore 服务层↗。
（推荐）**standard-rwo**	块存储	使用平衡型持久磁盘。
premium-rwx	文件存储	使用 Basic SSD Filestore 服务层↗。
premium-rwo	块存储	SSD 持久磁盘。
enterprise-rwx	文件存储	使用 Enterprise Filestore 层↗。
enterprise-multishare-rwx	文件存储	使用 Enterprise Filestore 层↗。详见 Google Kubernetes Engine 的 Filestore 多共享↗。

可用存储类参数

- 块存储可选参数及含义，详见 存储选项↗。
- 文件存储可选参数及含义，详见 服务层↗。

常见问题

文件存储类型存储类 PVC 创建失败

- 出现以下错误是因为项目未启用 Cloud Filestore API 或缺少相应权限。请参见 前提条件 解决：

```
failed to provision volume with StorageClass "standard-rwx": rpc error: code =
PermissionDenied desc = googlespi: Error 403: Cloud Filestore API has not been used in
project alauda-proj-1234 before or it is disabled.
...
resion: SERVICE_DISABLED
```

- 出现以下错误是因为超出存储配额。请参见 前提条件 解决：

```
failed to provision volume with StorageClass "standard-rwx": rpc error: code =
ResourceExhausted desc = googlespi: Error 429: Quora limit
'StandardStorageGbPerRegion' has been exceeded. Limit 2048 in region asia-east1.
 rateLimitExceeded
```

块存储类型存储类 PVC 无法正常绑定

出现以下错误是因为节点的 CSINode 缺少 pd.csi.storage.gke.io 驱动的配置。可通过重启 **Compute Engine Persistent Disk CSI Driver** 解决。

注意：更新该插件会导致集群不可用，更新过程约需 5-10 分钟。

```
Warning ProvisioningFailed 18m (x14 over 39m) pd.csi.storage.gke.io_gke-
5cb9bddae4d1430eb8ad-01f4-2084-vm_4b4e70bd-e2db-4779-9102-fee83a657ced failed to
provision volume with StorageClass "standard": error generating accessibility
requirements: no available topology found
Normal ExternalProvisioning 4m35s (x143 over 39m) persistentvolume-controller waiting for
a volume to be created, either by external provisioner "pd.csi.storage.gke.io" or
manually created by system administrator
Normal Provisioning 3m19s (x18 over 39m) pd.csi.storage.gke.io_gke-5cb9bddae4d1430eb8ad-
01f4-2084-vm_4b4e70bd-e2db-4779-9102-fee83a657ced External provisioner is provisioning
volume for claim "acp-gke-test/standard"
```

Menu

如何操作

导入集群的网络配置

获取导入集群信息

信任不安全的镜像仓库

从自定义命名的网卡采集网络数据

从自定义命名的网卡采集网络数据

导入集群的网络配置

目录

场景描述

在集群导入前，仅保证单向连通性，即 global 集群可以访问导入集群。集群导入后，为确保导入集群能够正常访问 global 集群，实现双向连通，需要额外的网络配置，以保证平台功能组件的正常运行。

前提条件

请提前准备好导入集群可访问的域名、该域名指向的**IP** 地址以及对应的有效证书。

注意：

- 该域名不得与平台访问地址相同。
- 确保该域名的端口（HTTPS 端口，与平台访问地址端口相同）能够将流量转发至 global 集群的所有控制节点。

为导入集群添加注解信息

具体来说，为确保告警和日志采集组件能够正常访问平台，必须为导入集群添加注解信息。

1. 在左侧导航栏点击 **Cluster Management** > **Clusters**。

2. 点击 **global**。

3. 点击 **Operations** > **CLI Tools**，并使用以下命令替换相关参数：

```
kubectl annotate --overwrite -n cpaas-system clusters.cluster.x-k8s.io <imported
cluster name> \
        cpaas.io/platform-url=<准备好的域名地址，例如：https://www.domain.cn>
```

代码示例：

```
kubectl annotate --overwrite -n cpaas-system clusters.cluster.x-k8s.io cluster-
imported \
        cpaas.io/platform-url=https://www.domain.cn
```

如何获取导入集群信息？

目录

问题描述

获取连接导入集群所需的配置信息，以便平台能够被授权访问和管理该集群。本节提供获取导入集群信息的操作步骤。

前提条件

- 已配置可用的 `kubectl` 环境。对于公有云集群，强烈建议使用云厂商提供的 **CloudShell**。如果无法使用 CloudShell，建议使用 Linux 或 macOS。
- 已获取导入集群的 KubeConfig 文件并复制到安装有 `kubectl` 的环境中。为避免误操作其他环境，强烈建议采用以下非破坏性方式之一：
  - 备份方式：先备份现有 kubeconfig 到安全位置：`cp "${HOME}/.kube/config" "${HOME}/.kube/config.backup"`

- 隔离方式：设置 `KUBECONFIG` 环境变量指向导入的 kubeconfig：`export KUBECONFIG="/path/to/imported/kubeconfig"`
- 合并方式：使用 kubectl 的合并/扁平化功能，且不丢失现有上下文：
  1. `export KUBECONFIG="/path/to/imported/kubeconfig:${HOME}/.kube/config"`
  2. `kubectl config view --flatten > "${HOME}/.kube/merged.kubeconfig"`
  3. `export KUBECONFIG="${HOME}/.kube/merged.kubeconfig"`

获取集群信息

获取集群 **token**

1. 执行以下命令：

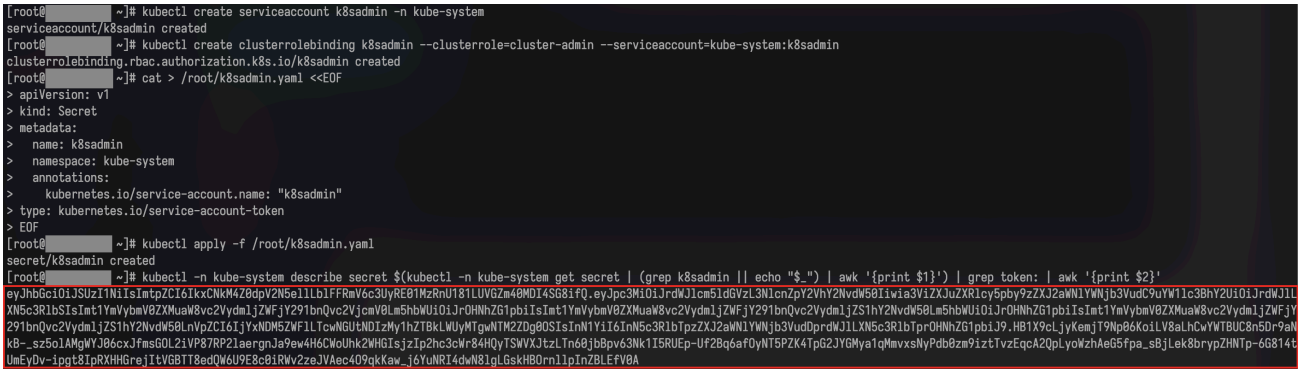
```
# [重要] 以下操作仅支持 bash

# 手动创建 secret, 绑定服务账号, 并生成一个不失效的 token
kubectl get ns cpaas-system > /dev/null 2>&1 || kubectl create namespace cpaas-system
kubectl create serviceaccount k8sadmin -n cpaas-system
kubectl create clusterrolebinding k8sadmin --clusterrole=cluster-admin --
serviceaccount=cpaas-system:k8sadmin

cat | kubectl apply -f - <<EOF
apiVersion: v1
kind: Secret
metadata:
  name: k8sadmin
  namespace: cpaas-system
  annotations:
    kubernetes.io/service-account.name: "k8sadmin"
type: kubernetes.io/service-account-token
EOF

kubectl -n cpaas-system describe secret \
    $(kubectl -n cpaas-system get secret | (grep k8sadmin || echo "$_") | awk '{print
$1}') \
    | grep -F 'token:' | awk '{print $2}'
```

WARNING

本操作创建了一个集群管理员级别的凭证，且该凭证设计为不失效。

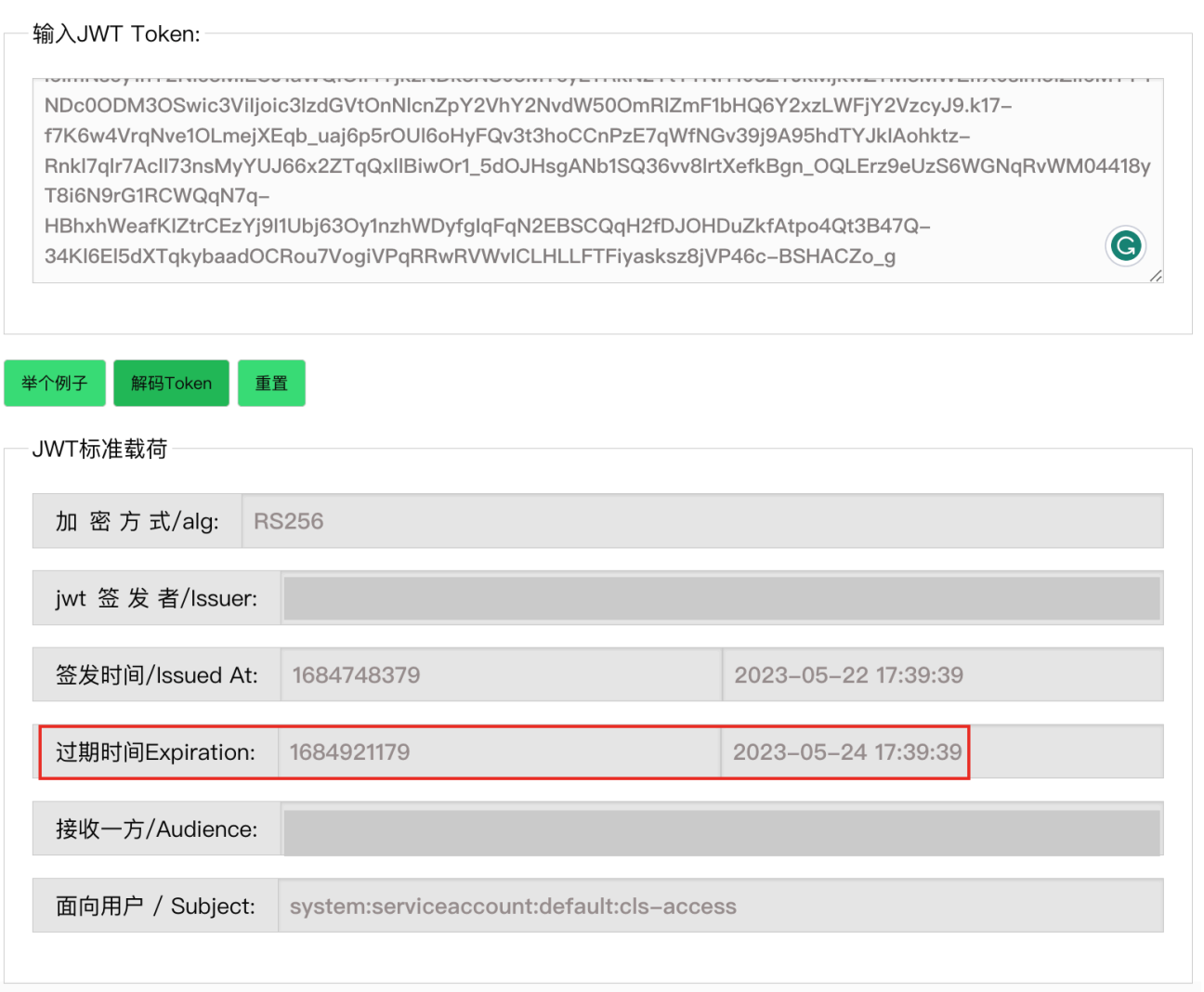
- 如可能，优先使用最小权限的 RBAC，限定到所需资源。
- 请妥善保管该 token，使用完成后应及时轮换或撤销。
- 限制谁可以读取 `cpaas-system` 命名空间中的 `Secret` 对象。

2. 以下为上一步获取的 token 示例。



3. 验证 token 过期时间。

   使用任何支持解析 JWT token 的工具分析该 token，确认其过期时间。 如果解析结果中能找到过期字段（包含 "exp" 的键，如下图所示），则平台在该时间之后将无法管理导入集群。遇此情况，请停止操作并联系技术支持。



TIP

过期时间在原始 JWT 负载中记录为 `"exp": 1684486916,`，该值为 UNIX 时间戳，可转换为 UTC 时间。

完成后清理（撤销访问权限）：

```
kubectl delete clusterrolebinding k8sadmin
kubectl -n cpaas-system delete secret k8sadmin
kubectl -n cpaas-system delete serviceaccount k8sadmin
```

获取导入集群 **API** 服务器地址及 **CA** 证书

TIP

如果已通过平台导入集群页面的 `Parse KubeConfig File` 功能获取了 API 服务器地址和 CA 证书，可跳过此步骤。

1. 执行以下命令：

```
# 查看导入集群的 API 服务器地址，可能有多个地址，请选择适合您环境的地址。
kubectl --kubeconfig "${KUBECONFIG:-${HOME}/.kube/config}" config view --show-managed-fields=false --flatten --raw -ojsonpath='{$.clusters..cluster.server}'
addr_apiserver='<Selected API server address>'

# 获取上述 API 服务器对应的 CA 证书
kubectl --kubeconfig "${KUBECONFIG:-${HOME}/.kube/config}" config view --show-managed-fields=false --flatten --raw \
    -ojsonpath="{$.clusters[?(@.cluster.server == '${addr_apiserver}')].cluster.certificate-authority-data}" \
    | { base64 -d 2>/dev/null || base64 -D; }
```

如何信任不安全的镜像仓库？

目录

问题描述

托管组件镜像的镜像仓库平台可能不提供 HTTPS 服务，或者没有由公共证书颁发机构签发的有效 TLS 证书。如果您信任该仓库，请按照以下步骤配置您的容器运行时。

配置信任不安全的镜像仓库

配置步骤因容器运行时而异。本文档涵盖 Docker 和 Containerd。

Docker 运行时

步骤

1. 在导入集群的每个节点上执行以下操作：

   - 备份 Docker 配置文件。

```
mkdir -p '/var/backup-docker-confs/'
if ! [ -f /etc/docker/daemon.json ]; then
    echo 'Docker 配置未找到。请检查 Docker 是否正确安装。如仍无法解决，请联系技术支持。'
    exit 1
else
    cp /etc/docker/daemon.json "/var/backup-docker-confs/daemon.json_$(date -u +%F_%R)"
fi
```

- 编辑 `/etc/docker/daemon.json`。

  确保存在 `insecure-registries` 参数，并添加之前获取的镜像仓库地址。 若有多个仓库，例如：

```
{
    "insecure-registries": [
        "<registry-address>",
        "192.168.134.43"
    ],
    "registry-mirrors": ["https://6telrzl8.mirror.aliyuncs.com"],
    "log-opts": {
      "max-size": "100m",
      "max-file": "2"
    },
    "live-restore": true,
    "metrics-addr": "0.0.0.0:9323",
    "experimental": true,
    "storage-driver": "overlay2"
}
```

2. （可选）使用 jq 验证 Docker 配置语法。

> **TIP**
>
> 确保已安装 `jq`。例如：`yum install jq -y`。

```
jq . < /etc/docker/daemon.json
```

3. 在所有节点上重启 Docker。

```
systemctl daemon-reload
systemctl restart docker
```

Containerd 运行时

注意：

- 所有需要使用镜像的节点（包括新加入的节点）都必须配置并重启 Containerd。
- Containerd v1.4/v1.5 与 v1.6 的配置略有不同，请根据版本选择相应步骤。

1. 在导入集群的每个节点上执行以下操作：

   - 备份配置文件

   ```
   mkdir -p '/var/backup-containerd-confs/'
   if ! [ -f /etc/containerd/config.toml ]; then
       echo 'Containerd 配置未找到。请检查 containerd 是否正确安装。如仍无法解决，请联系技术支持。'
       exit 1
   else
       cp /etc/containerd/config.toml /var/backup-containerd-confs/config.toml_$(date +%F_%T)
   fi
   ```

   - 获取 Containerd 运行时版本

   ```
   # 获取 containerd 版本
   # 与 v1.6 版本比较，选择相应步骤
   ctr --version | grep -Eo 'v[0-9]+\.[0-9]+\.[0-9]+'
   ```

Containerd v1.4 v1.5 不安全仓库配置

2. 在导入集群的每个节点上执行以下操作：

   - 编辑 `/etc/containerd/config.toml`

```
# 配置文件中添加示例内容
# 方括号内为节名称，若文件已有同名节，合并其内容
[plugins."io.containerd.grpc.v1.cri".registry]
    [plugins."io.containerd.grpc.v1.cri".registry.mirrors]
        [plugins."io.containerd.grpc.v1.cri".registry.mirrors."<registry-address>"]
            endpoint = ["https://<registry-address>", "http://<registry-address>"]
        [plugins."io.containerd.grpc.v1.cri".registry.mirrors."192.168.134.43"]
            endpoint = ["https://192.168.134.43", "http://192.168.134.43"]
    [plugins."io.containerd.grpc.v1.cri".registry.configs]
        [plugins."io.containerd.grpc.v1.cri".registry.configs."<registry-
address>".tls]
            insecure_skip_verify = true
        [plugins."io.containerd.grpc.v1.cri".registry.configs."192.168.134.43".tls]
            insecure_skip_verify = true
```

- 重启 Containerd。

```
systemctl daemon-reload && systemctl restart containerd
```

Containerd v1.6 不安全仓库配置

2. 在导入集群的每个节点上执行以下操作：

- 检查配置中是否存在 `config_path`。

```
if ! grep -qF 'config_path' /etc/containerd/config.toml; then
    if grep -qE '\[plugins."io.containerd.grpc.v1.cri".registry.(mirrors|configs)
(\.|\])' /etc/containerd/config.toml; then
        echo '请按照“Containerd v1.4 v1.5 不安全仓库配置”中的步骤操作。'
    else
        cat >> /etc/containerd/config.toml << 'EOF'
[plugins."io.containerd.grpc.v1.cri".registry]
    config_path = "/etc/containerd/certs.d/"
EOF
    fi
fi

config_path_var=$(grep -F '/etc/containerd/certs.d' /etc/containerd/config.toml)
if [ -z "$config_path_var" ]; then
   echo '文件中 config_path 的值异常，请检查！'
   exit 1
fi
```

- 创建 `hosts.toml` 文件。

  如果上一步输出了“请按照‘Containerd v1.4 v1.5 不安全仓库配置’中的步骤操作。”，请参见 *Containerd v1.4 v1.5 不安全仓库配置*。

```
REGISTRY='<registry address obtained in the "Get the registry address" section>'

mkdir -p "/etc/containerd/certs.d/$REGISTRY/"
cat > "/etc/containerd/certs.d/$REGISTRY/hosts.toml" << EOF
server = "$REGISTRY"
[host."http://$REGISTRY"]
  capabilities = ["pull", "resolve", "push"]
  skip_verify = true
[host."https://$REGISTRY"]
  capabilities = ["pull", "resolve", "push"]
  skip_verify = true
EOF
```

- 重启 Containerd。

```
systemctl daemon-reload && systemctl restart containerd
```

从自定义命名的网卡采集网络数据

目录

场景描述

创建业务集群后，平台监控默认只能识别匹配 `eth.*|en.*|wl.*|ww.*` 等模式的网卡名称。对于用户自定义的网卡名称，监控页面无法查看网络流量数据。为此，平台支持修改相关资源参数，手动采集网卡流量数据。

操作步骤

1. 登录 global 集群的控制节点，使用 kubectl 执行以下命令。

2. 首先，在 global 集群中查找对应业务集群的 moduleinfo 资源名称：

   ```
   kubectl get moduleinfo | grep -E 'prometheus|victoriametrics'
   ```

   示例输出：

```
global-6448ef7f7e5e3924c1629fad826372e7     global     prometheus          prometheus
Running   v3.15.0-zz231204040711-9d1fc12474c2   v3.15.0-zz231204040711-9d1fc12474c2
v3.15.0-zz231204040711-9d1fc12474c2
ovn-0954f21f0359720e8c115804376b3e7e        ovn        prometheus          prometheus
Running   v3.15.0-zz231204040711-9d1fc12474c2   v3.15.0-zz231204040711-9d1fc12474c2
v3.15.0-zz231204040711-9d1fc12474c2
```

3. 编辑业务集群对应的 moduleinfo 资源，将 `ovn-0954f21f0359720e8c115804376b3e7e` 替换为上一步查到的业务集群 moduleinfo 资源名称：

```
kubectl edit moduleinfo ovn-0954f21f0359720e8c115804376b3e7e
```

4. 添加 valuesOverride 字段，并根据注释信息修改字段和正则表达式：

```
spec:
  valuesOverride: # 如果该字段不存在，需要在 spec 下新增 valuesOverride 字段及以下参数
    ait/chart-cpaas-monitor:
      ovn: # 替换为业务集群名称
        indicator:
          networkDevice: eth.*|em.*|en.*|wl.*|ww.*|[A-Z].*i|custom_interface # 将
custom_interface 替换为自定义正则表达式，确保正确匹配网卡名称
```

5. 等待 10 分钟后，检查节点监控页面的网络相关图表，确认修改生效。

创建本地集群

目录

前提条件

节点要求

1. 如果您从下载安装包下载了单架构安装包，请确保节点机器的架构与安装包一致，否则节点因缺少对应架构的镜像无法启动。
2. 验证节点操作系统和内核是否受支持。详情请参见支持的操作系统和内核。
3. 对节点机器进行可用性检查。具体检查项请参考节点预处理 > 节点检查。
4. 如果节点机器的 IP 无法通过 SSH 直接访问，请为节点提供 SOCKS5 代理。`global` 集群将通过该代理服务访问节点。

负载均衡

生产环境中，集群控制平面节点需要负载均衡器以保证高可用性。 您可以提供自有硬件负载均衡器，或启用 `自建 VIP`，该方式通过 haproxy + keepalived 提供软件负载均衡。 建议使用硬件负载均衡器，原因如下：

- 性能更优：硬件负载均衡性能优于软件负载均衡。
- 复杂度更低：如果不熟悉 keepalived，配置错误可能导致集群不可用，排查耗时且严重影响集群可靠性。

使用自有硬件负载均衡器时，可将负载均衡器的 VIP 作为 `IP 地址 / 域名` 参数；如果有解析到负载均衡器 VIP 的域名，也可使用该域名作为 `IP 地址 / 域名` 参数。 注意：

- 负载均衡器必须正确转发流量至集群所有控制平面节点的端口 `6443`、`11780` 和 `11781`。
- 如果集群仅有一个控制平面节点，且使用该节点 IP 作为 `IP 地址 / 域名` 参数，则无法后续将单节点集群扩展为高可用多节点集群。因此，建议即使是单节点集群也提供负载均衡器。

启用 `自建 VIP` 时，需准备：

1. 可用的 VRID
2. 支持 VRRP 协议的主机网络
3. 所有控制平面节点和 VIP 必须在同一子网，且 VIP 不能与任何节点 IP 重复。

连接 `global` 集群与业务集群

平台要求 `global` 集群与业务集群之间互通访问。若不在同一网络，需要：

1. 为业务集群提供 `外部访问`，确保 `global` 集群可访问业务集群。网络要求确保 `global` 可访问所有控制平面节点的端口 `6443`、`11780` 和 `11781`。

2. 为 `global` 添加业务集群可访问的额外地址。创建业务集群时，将该地址以注解形式添加到集群，键为 `cpaas.io/platform-url`，值为 `global` 的公网访问地址。

镜像仓库

集群镜像支持平台内置、私有仓库和公共仓库三种选项。

- 平台内置：使用 `global` 集群提供的镜像仓库。若集群无法访问 `global`，请参见为内置仓库添加外部地址。
- 私有仓库：使用您自建的镜像仓库。推送所需镜像至私有仓库的具体操作，请联系技术支持。
- 公共仓库：使用平台公共镜像仓库。使用前请完成更新公共仓库凭据。

容器网络

若计划使用 Kube-OVN 的 Underlay 方案，请参考准备 Kube-OVN Underlay 物理网络。

创建操作步骤

1. 进入管理员视图，点击左侧导航栏的集群**/Clusters**。
2. 点击创建集群。
3. 按照以下说明配置：基础信息、容器网络、节点设置和扩展参数。

基础信息

参数	说明

Kubernetes 版本	所有可选版本均经过严格测试，保证稳定性和兼容性。 推荐：选择最新版本以获得最佳功能和支持。
容器运行时	默认提供 containerd 作为容器运行时。 若偏好使用 Docker 作为容器运行时，请参见选择容器运行时。
集群网络协议	支持三种模式：IPv4 单栈、IPv6 单栈、IPv4/IPv6 双栈。 注意：选择双栈模式时，确保所有节点均正确配置 IPv6 地址；网络协议设置后不可更改。
集群访问端点	`IP 地址 / 域名`：填写预先准备的域名，若无域名则填写 VIP。 `自建 VIP`：默认关闭。仅当未提供 LoadBalancer 时启用，启用后安装程序会自动部署 `keepalived` 实现软件负载均衡支持。 `外部访问`：当集群与 `global` 集群不在同一网络环境时，填写为集群准备的外部可访问地址。

容器网络

Kube-OVN

Alauda 开发的企业级云原生 Kubernetes 容器网络编排系统。它将 OpenStack 领域成熟的网络能力引入 Kubernetes，支持跨云网络管理、传统网络架构与基础设施互联及边缘集群部署场景，同时大幅提升 Kubernetes 容器网络的安全性、管理效率和性能。

参数	说明
子网	又称 Cluster CIDR，表示默认子网段。集群创建后可添加额外子网。
传输模式	**Overlay**：基于基础设施抽象的虚拟网络，不占用物理网络资源。创建 Overlay 默认子网时，集群内所有 Overlay 子网使用相同的集群 NIC 和节点 NIC 配置。 **Underlay**：依赖物理网络设备的传输方式，可直接为 Pod 分配物理网络地址，性能和物理网络连通性更佳。Underlay 子网的节点必须有多块网卡，桥接网络使用的网卡必须专用于 Underlay，且不承载其他流量（如 SSH）。创建 Underlay 默认子网时，集群 NIC 实为桥接网络默认 NIC，节点 NIC 为桥接网络中的节点 NIC 配置。 • 默认网关：物理网络网关地址，即 Cluster CIDR 段的网关地址（必须在 Cluster CIDR 地址范围内）。 • **VLAN ID**：虚拟局域网标识（VLAN 编号），例如 `0`。 • 保留 **IP**：设置不自动分配的保留 IP，如子网内已被其他设备使用的 IP。
服务 **CIDR**	Kubernetes 类型为 ClusterIP 的 Service 使用的 IP 地址范围。不可与默认子网范围重叠。
Join CIDR	Overlay 传输模式下，节点与 Pod 通信使用的 IP 地址范围。不可与默认子网或服务 CIDR 重叠。

Calico

Calico 是一种三层网络方案，为容器提供安全的网络连接。

参数	说明
默认子网	又称 Cluster CIDR，表示默认子网段。集群创建后可添加额外子网。
服务 **CIDR**	Kubernetes 类型为 ClusterIP 的 Service 使用的 IP 地址范围。不可与默认子网范围重叠。

Flannel

Flannel 为集群内所有容器提供扁平网络环境，使不同节点主机上创建的容器拥有跨集群唯一的虚拟 IP 地址。Pod 子网根据掩码均匀划分给集群节点，每个节点上的 Pod 从分配给该节点的段中获取 IP 地址。提升容器间通信效率，无需考虑 IP 转换问题。

参数	说明
集群 **CIDR**	集群启动时创建 Pod 使用的 IP 地址范围。支持设置当前容器网络下每个节点可分配给 Pod 的最大 IP 数量。 注意：平台会根据上述配置自动计算集群可容纳的最大节点数，并在输入框下方提示显示。 重要：集群创建后，集群网络不可更改，请谨慎规划网络。
服务 **CIDR**	Kubernetes 类型为 ClusterIP 的 Service 使用的 IP 地址范围。不可与容器子网范围重叠。

自定义

若需安装其他网络插件，选择自定义模式。集群创建成功后，可手动安装网络插件。

参数	说明
集群 **CIDR**	集群启动时创建 Pod 使用的 IP 地址范围。
服务 **CIDR**	Kubernetes 类型为 ClusterIP 的 Service 使用的 IP 地址范围。不可与容器子网范围重叠。

节点设置

参数	说明
网卡名称	集群网络插件使用的主机网络接口设备名称。 注意： • 选择 Kube-OVN 默认子网的 Underlay 传输模式时，必须指定网卡名称，该网卡将作为桥接网络的默认 NIC。 - 平台默认识别名称类似 `eth.|en.|wl./ww.` 的网卡流量进行监控。若使用其他命名规则的网卡，请参考从自定义命名网卡采集网络数据修改相关资源，确保平台能正确监控网卡流量。
节点名称	可选择使用节点 IP 或主机名作为平台上的节点名称。 注意：选择主机名作为节点名称时，确保集群中所有节点主机名唯一。

节点列表	添加节点至集群，或从草稿恢复暂存的节点信息。详见下方添加节点参数说明。
监控类型	支持 **Prometheus** 和 **VictoriaMetrics**。 选择 **VictoriaMetrics** 时，需配置部署类型： **-** 部署 **VictoriaMetrics**：部署所有相关组件，包括 **VMStorage**、**VMAlert**、**VMAgent** 等。 **-** 部署 **VictoriaMetrics Agent**：仅部署日志采集组件 **VMAgent**。此方式需关联平台中已部署的其他集群上的 VictoriaMetrics 实例，为集群提供监控服务。
监控节点	选择部署集群监控组件的节点。支持选择允许部署应用的计算节点和控制平面节点。 为避免影响集群性能，建议优先选择计算节点。集群创建成功后，存储类型为本地卷 的监控组件将在所选节点部署。

添加节点参数

参数	说明
类型	控制平面节点：负责运行 kube-apiserver、kube-scheduler、kube-controller-manager、etcd、容器网络及部分平台管理组件。启用允许部署应用后，控制平面节点也可作为计算节点使用。 工作节点：负责承载集群中运行的业务 Pod。

IPv4 地址	节点的 IPv4 地址。内网模式创建的集群填写节点的私有 **IP**。
IPv6 地址	集群启用 IPv4/IPv6 双栈时有效。节点的 IPv6 地址。
允许部署应用	集群中 节点类型为 控制平面节点 时有效。是否允许在该控制平面节点部署业务应用，调度业务相关 Pod 到该节点。
显示名称	节点的显示名称。
SSH 连接 **IP**	访问节点 SSH 服务时可连接的 IP 地址。 若可通过 `ssh <用户名>@<节点 IPv4 地址>` 登录节点，则此参数可不填；否则填写节点的公网 IP 或 NAT 外网 IP，确保 `global` 集群及代理可通过该 IP 连接节点。
网卡名称	输入节点使用的网卡名称。网卡配置生效优先级如下（从左到右，依次降低）： **Kube-OVN Underlay**：节点 NIC > 集群 NIC **Kube-OVN Overlay**：节点 NIC > 集群 NIC > 节点默认路由对应的 NIC **Calico**：集群 NIC > 节点默认路由对应的 NIC

	Flannel：集群 NIC > 节点默认路由对应的 NIC
关联桥接网络	注意：创建集群时不支持桥接网络配置，仅在为已创建 Underlay 子网的集群添加节点时可用。 选择已有的添加桥接网络。若不想使用桥接网络默认 NIC，可单独配置节点 NIC。
SSH 端口	SSH 服务端口号，例如 `22`。
SSH 用户名	SSH 用户名，需为具有 root 权限的用户，例如 `root`。
代理	是否通过代理访问节点的 SSH 端口。当 `global` 集群无法直接通过 SSH 访问待添加节点（如 `global` 集群与业务集群不在同一子网；节点 IP 为 `global` 集群无法直接访问的内网 IP）时，需开启此开关并配置代理相关参数。配置代理后，可通过代理访问和部署节点。 注意：当前仅支持 SOCKS5 代理。 访问地址：代理服务器地址，例如 `192.168.1.1:1080`。 用户名：访问代理服务器的用户名。 密码：访问代理服务器的密码。
SSH 认证	登录添加节点的认证方式及对应认证信息。选项包括：

	密码：需提供具有 root 权限的用户名及对应的 **SSH** 密码。 密钥：需提供具有 root 权限的 私钥及 私钥密码。
保存草稿	将当前对话框配置的数据保存为草稿，并关闭添加节点对话框。 在不离开创建集群页面的情况下，可选择从草稿恢复打开添加节点对话框，恢复保存的草稿配置数据。 注意：恢复的数据为最近一次保存的草稿数据。

扩展参数

注意：

- 除必填配置外，不建议设置扩展参数，错误设置可能导致集群不可用，且集群创建后无法修改。
- 若输入的**Key**与默认参数**Key**重复，则会覆盖默认配置。

操作步骤

1. 点击扩展参数展开扩展参数配置区域。可选为集群设置以下扩展参数：

参数	说明
Docker 参数	`dockerExtraArgs`，Docker 的额外配置参数，将写入 `/etc/sysconfig/docker`。不建议修改。若通过 `daemon.json` 配置 Docker，需以键值对形式配置。
Kubelet 参数	`kubeletExtraArgs`，Kubelet 的额外配置参数。

	注意：当输入 容器网络的 节点 **IP** 数量 参数时，系统会自动生成默认的 **Kubelet** 参数，键为 `max-pods`，值为 节点 **IP** 数量，用于设置集群中任一节点可运行的最大 Pod 数量。该配置不在界面显示。 在 **Kubelet** 参数区域新增 `max-pods: 最大可运行 Pod 数` 键值对会覆盖默认值。允许任意正整数，但建议使用默认值（节点 IP 数量）或不超过 `256`。
Controller Manager 参数	`controllerManagerExtraArgs`，Controller Manager 的额外配置参数。
Scheduler 参数	`schedulerExtraArgs`，Scheduler 的额外配置参数。
APIServer 参数	`apiServerExtraArgs`，APIServer 的额外配置参数。
APIServer URL	`publicAlternativeNames`，证书中 APIServer 访问地址。仅可填写 IP 或域名，最长 253 字符。
集群注解	集群注解信息，以键值对形式标记集群特性，供平台组件或业务组件获取相关信息。

4. 点击创建。页面将返回集群列表，集群状态为创建中。

创建后操作

查看创建进度

在集群列表页面，可查看已创建集群列表。对于状态为创建中的集群，可查看执行进度。

操作步骤

1. 点击集群状态右侧的小图标查看执行进度。

2. 弹出的执行进度对话框中，可查看集群执行进度（status.conditions）。

   提示：当某类型处于进行中或失败状态且有原因时，将鼠标悬停在对应原因（蓝色文字）上，可查看原因详情（status.conditions.reason）。

关联项目

集群创建完成后，可在项目管理视图中将其添加到项目中。

关于托管控制平面

Hosted Control Plane (HCP) 是一种轻量级的多集群管理模型，将控制平面与工作节点分离。每个集群的控制平面都被容器化并托管在管理集群中，从而减少资源消耗，加快集群创建和升级速度，并提升多集群操作的可扩展性。

> **Note**
>
> 因为 Hosted Control Plane 的发版周期与灵雀云容器平台不同，所以 Hosted Control Plane 的文档现在作为独立的文档站点托管在 Hosted Control Plane ↗。

集群节点规划

集群利用 Kubernetes 节点角色标签 `node-role.kubernetes.io/<role>` 来为节点分配不同的角色。为了描述方便，我们将此类标签称为角色标签。

默认情况下，集群包含两种类型的节点：控制平面节点和工作节点，分别用于承载控制平面工作负载和应用工作负载。

在集群中：

- 控制平面节点被标记为角色标签 `node-role.kubernetes.io/control-plane`。

  > 注意：
  > 在 Kubernetes v1.24 之前，社区也使用标签 `node-role.kubernetes.io/master` 来标记控制平面节点。为兼容历史版本，这两个标签均被视为识别控制平面节点的有效标签。

- 工作节点默认没有角色标签，但你也可以显式地为工作节点分配角色标签 `node-role.kubernetes.io/worker`。

除了这些默认的角色标签外，你还可以在工作节点上定义自定义角色标签，以进一步将它们划分为不同的功能类型。例如：

- 你可以添加角色标签 `node-role.kubernetes.io/infra`，将节点指定为 infra 节点，用于承载基础设施组件。

- 你可以添加角色标签 `node-role.kubernetes.io/log`，将节点指定为 log 节点，专门用于承载日志组件。

本文档将指导你如何创建 infra 节点和自定义角色节点，并将工作负载迁移到这些节点上。

目录

在非不可变集群上创建 **Infra** 节点

默认情况下，集群只包含控制平面节点和工作节点。如果你想将某些工作节点指定为专门承载基础设施组件的 infra 节点，需要手动为这些节点添加相应的角色标签和污点。

> 注意：
> 本节操作仅适用于非不可变集群。即以下操作不支持在云集群（例如通过 `Alauda Container Platform EKS Provider` 集群插件部署的 EKS 托管集群）、第三方集群或节点使用不可变操作系统的集群上执行。

添加 **Infra** 节点

步骤 **1**：为节点资源添加 **Infra** 角色标签

```
kubectl label nodes 192.168.143.133 node-role.kubernetes.io/infra="" --overwrite
```

该命令为节点 192.168.143.133 添加 infra 角色标签：`node-role.kubernetes.io/infra: ""`，表示该节点为 infra 节点。

步骤 2：为节点资源添加污点

为防止其他工作负载调度到 infra 节点，添加污点。

```
kubectl taint nodes 192.168.143.133 node-role.kubernetes.io/infra=reserved:NoSchedule
```

该命令为节点 192.168.143.133 添加污点 `node-role.kubernetes.io/infra=reserved:NoSchedule`，表示只有容忍该污点的应用才能调度到此节点。

步骤 3：验证标签和污点

检查节点是否已分配 infra 角色标签和污点：

```
# kubectl describe node 192.168.143.133
Name:               192.168.143.133
Roles:              infra
Labels:             node-role.kubernetes.io/infra=reserved
                    ...
Taints:             node-role.kubernetes.io/infra=reserved:NoSchedule
```

输出表明节点 192.168.143.133 已被配置为 infra 节点，并带有污点 `node-role.kubernetes.io/infra=reserved:NoSchedule`。

将 Pod 迁移到 Infra 节点

如果你想将特定 Pod 调度到 infra 节点，需要进行以下配置：

- 使用 nodeSelector 指定 infra 角色标签。
- 配置对应的容忍污点（tolerations）以容忍 infra 节点的污点。

下面是一个配置为运行在 infra 节点上的 Deployment 示例清单。

```
apiVersion: apps/v1
kind: Pod
metadata:
  name: infra-pod-demo
  namespace: default
spec:
  ...
  nodeSelector:
    node-role.kubernetes.io/infra: ""
  tolerations:
  - effect: NoSchedule
    key: node-role.kubernetes.io/infra
    value: reserved
    operator: Equal
  ...
```

nodeSelector 确保 Pod 只调度到带有标签 `node-role.kubernetes.io/infra: ""` 的节点， 容忍配置允许 Pod 容忍污点 `node-role.kubernetes.io/infra=reserved:NoSchedule`。

通过这些配置，Pod 将被调度到 infra 节点。

> 注意：
> 通过 OLM Operators 或集群插件安装的 Pod 不一定能迁移到 infra 节点，是否支持迁移取决于各 Operator 或集群插件的配置。

自定义节点规划

除了 infra 节点，你可能还想将工作节点指定为其他专用用途——例如承载日志组件、存储服务或监控代理。

你可以通过为工作节点分配更多自定义角色标签及对应污点，将其转变为自定义角色节点。

定义自定义角色节点的一般步骤

流程与创建 infra 节点类似。

步骤 1：添加自定义角色标签

```
kubectl label nodes <node> node-role.kubernetes.io/<role>="" --overwrite
```

将 <role> 替换为你期望的角色名称，如 monitoring、storage 或 log。

步骤 2：添加对应污点

```
kubectl taint nodes <node> node-role.kubernetes.io/<role>=<value>:NoSchedule
```

将 <role> 替换为你的自定义角色名称，<value> 替换为有意义的描述，如 reserved 或 dedicated。该值为可选，但有助于文档说明和清晰度。

步骤 3：验证配置

```
kubectl describe node <node>
```

确保 Labels 和 Taints 字段反映你的自定义角色配置。

示例：创建专用于日志组件的节点

如果你想创建专门用于安装日志组件的节点，可以添加 log 角色。具体操作如下。

步骤 1：添加 **Log** 角色标签

```
kubectl label nodes 192.168.143.133 node-role.kubernetes.io/log="" --overwrite
```

该标签表示该节点被指定用于日志相关工作负载。

步骤 2：为节点添加污点

```
kubectl taint nodes 192.168.143.133 node-role.kubernetes.io/log=reserved:NoSchedule
```

该污点防止未容忍的工作负载调度到该节点。

步骤 3：验证标签和污点

```
Name:               192.168.143.133
Roles:              log
Labels:             node-role.kubernetes.io/log=reserved
                    ...
Taints:             node-role.kubernetes.io/log=reserved:NoSchedule
```

确认节点已成功配置为 log 节点，并带有相应的标签和污点。

通过以上实践，你可以根据节点的预期用途有效地划分 Kubernetes 节点，提升工作负载隔离性，确保特定组件部署到配置合适的节点上。

etcd 加密

本指南帮助您安装、理解和操作 ACP 中的 etcd Encryption Manager，以实现集群内 etcd 数据加密密钥的自动轮换。

它确保存储在 etcd 中的敏感数据（如 secrets 和 configmaps）使用安全算法加密，从而增强集群的安全性。

目录

安装

请参见 Cluster Plugin 获取安装说明。

> 注意：
>
> - 当前支持：
>
> - On-Premises 集群

> - DCS 集群
> - 不支持：
>   - `global cluster`

工作原理

安装后，会在 `kube-system` 命名空间部署一个 `etcd-encryption-manager` 控制器，该控制器：

- 定期轮换 etcd 数据加密密钥。
- 保留最近 8 个密钥以支持回滚兼容。
- 更新所有控制节点上的加密配置。
- 触发 `kube-apiserver` 热加载新密钥。
- 自动迁移资源，使用新密钥重新加密数据。

在整个操作过程中，集群保持稳定。

默认配置

参数	值
加密资源	secrets, configmaps
加密算法	256-bit AES-GCM
轮换间隔	168 小时（7 天）

操作指南

配置文件

路径	内容
`/etc/kubernetes/encryption-provider.conf`	当前加密配置
`/etc/kubernetes/encryption-provider-history.bak`	历史密钥记录（用于恢复）
`/etc/kubernetes/encryption-provider-bak/`	过期的加密配置版本

状态检查

运行以下命令检查当前轮换状态：

```
kubectl get EtcdEncryptionConfig default -o yaml
```

示例输出：

```yaml
apiVersion: cluster.alauda.io/v1alpha1
kind: EtcdEncryptionConfig
metadata:
  name: default
spec:
  resources:
    - secrets
    - configmaps
  rotationInterval: 168h0m0s
  type: aesgcm
status:
  deployStatus:
    192.168.100.1:
      revision: 3
      state: Success
    192.168.100.2:
      revision: 3
      state: Success
    192.168.100.3:
      revision: 3
      state: Success
  migration:
    completeTimestamp: "2025-05-27T05:47:01Z"
    resources:
      - secrets
      - configmaps
    revision: 3
    state: Success
  revision: 3
```

Menu

如何操作

为内置镜像仓库添加外部访问地址

选择容器运行时

使用 **Manager** 策略优化 **Pod** 性能

更新公共仓库凭证

为内置镜像仓库添加外部访问地址

目录

Overview

当 `global` 集群使用 `Platform Built-in` 镜像仓库时，业务集群通常也会使用该仓库拉取镜像。该镜像仓库不仅为 `global` 集群内的组件提供服务，还必须对业务集群节点可访问。

在某些场景下，业务集群节点无法直接访问 `global` 集群的镜像仓库地址——例如 `global` 集群位于私有数据中心，而业务集群位于公有云或边缘环境时。

本指南介绍如何为平台默认镜像仓库配置一个外部可访问的地址，以便业务集群能够拉取镜像。

Prerequisites

开始之前，请准备以下内容：

- 业务集群节点可访问的域名

- 域名指向的 IP 地址
- 域名对应的有效 SSL 证书

WARNING

- 域名必须与平台访问地址不同
- 确保该域名的 IP 地址能够将流量转发至 `global` 集群所有控制平面节点

Procedure

配置平台镜像仓库的证书和路由规则

1. 将域名的有效证书复制到 `global` 集群任一控制平面节点

2. 创建包含域名证书的 TLS secret：

   ```
   kubectl create secret tls registry-address.tls --cert=<certificate-filename> --key=
   <key-filename> -n kube-system
   ```

   示例：

   ```
   kubectl create secret tls registry-address.tls --cert=custom.crt --key=custom.key -n
   kube-system
   ```

   注意：创建证书后，请监控 `global` 集群 `kube-system` 命名空间中 **registry-address.tls** secret 的过期时间，证书到期前及时替换。

3. 在 `global` 集群任一控制平面节点创建 ingress 规则：

```
REGISTRY_DOMAIN_NAME=<www.registry.com> # 替换为你的可访问域名
cat << EOF | kubectl create -f -
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  annotations:
    nginx.ingress.kubernetes.io/backend-protocol: HTTPS
  name: registry-address
  namespace: kube-system
  labels:
    service_name: registry
spec:
  rules:
    - host: $REGISTRY_DOMAIN_NAME
      http:
        paths:
          - backend:
              service:
                name: registry
                port:
                  number: 443
            path: /v2/
            pathType: ImplementationSpecific
          - backend:
              service:
                name: registry
                port:
                  number: 443
            path: /v2/_catalog
            pathType: ImplementationSpecific
          - backend:
              service:
                name: registry
                port:
                  number: 443
            path: /v2/.+/tags/list
            pathType: ImplementationSpecific
          - backend:
              service:
                name: registry
                port:
                  number: 443
            path: /v2/.+/manifests/[A-Za-z0-9_+.-:]+
```

```
            pathType: ImplementationSpecific
          - backend:
              service:
                name: registry
                port:
                  number: 443
            path: /v2/.+/blobls/[A-Za-z0-9-:]+
            pathType: ImplementationSpecific
          - backend:
              service:
                name: registry
                port:
                  number: 443
            path: /v2/.+/blobls/uploads/[A-Za-z0-9-:]+
            pathType: ImplementationSpecific
          - backend:
              service:
                name: registry
                port:
                  number: 443
            path: /auth/token
            pathType: ImplementationSpecific
  tls:
    - secretName: registry-address.tls
      hosts:
        - $REGISTRY_DOMAIN_NAME
EOF
```

返回类似 `... created` 表示 ingress 创建成功。

4. 检查是否存在 Registry Service 资源：

```
kubectl -n kube-system get svc | grep registry
```

若 Service 不存在，使用以下命令创建：

```
cat << EOF | kubectl create -f -
apiVersion: v1
kind: Service
metadata:
  labels:
    name: registry
    service_name: registry
  name: registry
  namespace: kube-system
spec:
  ports:
    - protocol: TCP
      port: 443
      targetPort: 60080
  selector:
    component: registry
  type: ClusterIP
EOF
```

5. 通过域名拉取镜像测试配置：

```
crictl pull <registry-domain-name>/automation/qaimages:helloworld
```

或

```
docker pull <registry-domain-name>/automation/qaimages:helloworld
```

选择容器运行时

目录

Overview

Container Runtime 是 Kubernetes 的核心组件，负责管理镜像和容器的生命周期。

通过平台创建集群时，可以选择 Containerd 或 Docker 作为运行时组件。

注意：Kubernetes 1.24 及以上版本不再官方支持 Docker 运行时。官方推荐的运行时是 Containerd。如果仍需使用 Docker 运行时，必须先在 feature gate 中启用 `cri-docker`，才能在创建集群时选择 Docker 作为运行时组件。有关 feature gate 的使用详情，请参见 Feature Gate Configuration。

快速选择指南

选择 **Containerd**	选择 **Docker**
• 调用链更短 • 组件更少 • 更加稳定 • 消耗更少的节点资源	• 支持 docker-in-docker • 允许在节点上使用 `docker build/push/save/load` 命令 • 可以调用 Docker API • 支持 docker compose 或 docker swarm

Docker 与 Containerd 的区别

常用命令

Containerd	**Docker**	说明
crictl ps	docker ps	查看运行中的容器
crictl inspect	docker inspect	查看容器详情
crictl logs	docker logs	查看容器日志
crictl exec	docker exec	在容器内执行命令
crictl attach	docker attach	附加到容器
crictl stats	docker stats	显示容器资源使用情况
crictl create	docker create	创建容器
crictl start	docker start	启动容器
crictl stop	docker stop	停止容器
crictl rm	docker rm	删除容器
crictl images	docker images	查看镜像列表
crictl pull	docker pull	拉取镜像

Containerd	Docker	说明
None	docker push	推送镜像
crictl rmi	docker rmi	删除镜像
crictl pods	None	查看 Pod 列表
crictl inspectp	None	查看 Pod 详情
crictl runp	None	启动 Pod
crictl stopp	docker images	查看镜像
ctr images ls	None	停止 Pod
crictl stopp	docker load/save	导入/导出镜像
ctr images import/export	None	停止 Pod
ctr images pull/push	docker pull/push	拉取/推送镜像
ctr images tag	docker tag	镜像打标签

调用链差异

- Docker 作为 Kubernetes 容器运行时的调用关系为：

  kubelet > cri-dockerd > dockerd > containerd > runC

- Containerd 作为 Kubernetes 容器运行时的调用关系为：

  kubelet > cri 插件（在 containerd 进程中）> containerd > runC

总结：虽然 dockerd 增加了 swarm 集群、docker build 和 Docker API 等功能，但也可能引入 bug，并增加调用链层级。Containerd 调用链更短，组件更少，稳定性更高，且消耗更少的节点资源。

日志与参数对比

对比项	Docker	Containerd
存储路径	当 Docker 作为 Kubernetes 容器运行时时，容器日志由 Docker 存储在 `/var/lib/docker/containers/$CONTAINERID` 等目录下。Kubelet 会在 `/var/log/pods` 和 `/var/log/containers` 创建指向该目录下容器日志文件的符号链接。	当 Containerd 作为 Kubernetes 容器运行时时，容器日志由 Kubelet 存储在 `/var/log/pods/$CONTAINER_NAME` 目录下，并在 `/var/log/containers` 目录创建指向日志文件的符号链接。

配置参数	在 Docker 配置文件中指定：
	`"log-driver": "json-file",`
	`"log-opts": {"max-size": "100m","max-file": "5"}`
		方式一：在 kubelet 参数中指定：
		`--container-log-max-files=5`
		`--container-log-max-size="100Mi"`
		方式二：在 KubeletConfiguration 中指定：
		`"containerLogMaxSize": "100Mi",`
		`"containerLogMaxFiles": 5,`

将容器日志保存到数据盘	将数据盘挂载到 "data-root"（默认是 `/var/lib/docker`）。	创建符号链接 `/var/log/pods` 指向数据盘挂载点下的目录。

CNI 网络对比

对比项	**Docker**	**Containerd**
谁调用 **CNI**	cri-dockerd	集成在 Containerd 中的 cri-plugin（containerd 1.1 之后）

如何配置 **CNI**	cri-dockerd 参数 `--cni-conf-dir` `--cni-bin-dir` `--cni-cache-dir`	Containerd 配置文件（toml）：
		`[plugins.cri.cni]`
		`bin_dir = "/opt/cni/bin"`
		`conf_dir = "/etc/cni/net.d"`

使用 **Manager** 策略优化 **Pod** 性能

本指南为 Kubernetes 集群管理员提供了一份实用且可直接应用的手册，介绍如何启用和验证 **CPU ManagerPolicy**、**Memory ManagerPolicy** 和 **Topology ManagerPolicy**。通过协调 CPU 绑定、NUMA 亲和性和拓扑对齐，您可以为关键工作负载提供一致的延迟和更优的性能。

目录

范围和前提条件

角色和权限

- 需要维护窗口访问权限、`kubectl` 管理权限以及节点的 SSH 访问权限。

工作负载要求

- 为实现专用 CPU 和 NUMA 亲和性，Pod 必须运行在 **Guaranteed QoS** 类别：requests = limits 且 CPU 以完整核数指定（例如 2、4）。

不涵盖内容

- HugePages 不在本指南范围内。如需 HugePages 支持，请联系您的支持团队。

快速开始：示例 **Kubelet** 配置

将以下片段添加到 `/var/lib/kubelet/config.yaml`，并根据您的环境调整数值：

```
# -- CPU ManagerPolicy --
cpuManagerPolicy: "static"              # 选项：none | static
cpuManagerPolicyOptions:
  full-pcpus-only: "true"               # 推荐：仅分配完整物理核
cpuManagerReconcilePeriod: "5s"
reservedSystemCPUs: ""                  # 例如 "0-1"，如果为系统保留特定 CPU

# -- Memory ManagerPolicy --
memoryManagerPolicy: "Static"           # 选项：none | Static
reservedMemory:
  - numaNode: 0
    limits:
      memory: "2048Mi"
  - numaNode: 1
    limits:
      memory: "2048Mi"

# -- Topology ManagerPolicy --
topologyManagerPolicy: "single-numa-node"     # 选项：none | best-effort | restricted |
single-numa-node
topologyManagerScope: "pod"                   # 选项：container | pod
```

说明：

- `full-pcpus-only: "true"` 有助于提升延迟一致性。
- `topologyManagerScope: pod` 确保同一 Pod 内的容器对齐到相同的 NUMA 拓扑。
- `reservedMemory` 必须基于 kubelet 配置和驱逐阈值计算（详见下一节）。

如何计算 `reservedMemory`

计算公式：

```
R_total = kubeReserved(memory) + systemReserved(memory) + evictionHard(memory.available)
```

所有 NUMA 节点的 `reservedMemory` 之和必须等于 `R_total`。

步骤（针对 **N** 个 **NUMA** 节点）：

1. 计算 `R_total`（单位 Mi）。
2. 计算商和余数：
   - base = floor(R_total / N)
   - rem = R_total − base × N
3. 分配数值：
   - NUMA 节点 0 = base + rem
   - 其余 NUMA 节点 = base

示例（**2** 个 **NUMA** 节点）：

- kubeReserved=512Mi，systemReserved=512Mi，evictionHard=100Mi → R_total = 1124Mi
- base = 562，rem = 0

```yaml
reservedMemory:
- numaNode: 0
  limits:
    memory: "562Mi"
- numaNode: 1
  limits:
    memory: "562Mi"
```

应用配置

针对每个节点：

1. 标记为不可调度并驱逐 **Pod**

```
kubectl cordon <node>
kubectl drain <node> --ignore-daemonsets --delete-emptydir-data
```

2. 停止 **Kubelet** 并清理状态

```
sudo systemctl stop kubelet
sudo rm -f /var/lib/kubelet/cpu_manager_state
sudo rm -f /var/lib/kubelet/memory_manager_state
```

3. 重载守护进程并启动 **Kubelet**

```
sudo systemctl daemon-reload
sudo systemctl start kubelet
```

4. 恢复节点调度

```
kubectl uncordon <node>
```

- 对于 DaemonSet 和系统 Pod，请显式重启或删除 Pod。

5. 验证恢复状态

```
kubectl get nodes
kubectl get pods -A -o wide | grep <node>
```

验证

CPU ManagerPolicy 状态

```
sudo cat /var/lib/kubelet/cpu_manager_state | jq .
```

检查：

- `.policyName` = `"static"`
- `.defaultCpuSet` 列出非专用 CPU
- `.entries` 显示容器与 CPU 的分配关系

Memory ManagerPolicy 状态

```
sudo cat /var/lib/kubelet/memory_manager_state | jq .
```

检查：

- `.policyName` = `"Static"`
- 保留内存总和匹配 `R_total`
- Guaranteed Pod 根据 `single-numa-node` 策略分配到对应 NUMA 节点

关键策略和行为

CPU ManagerPolicy

- 目的：为 Guaranteed Pod 分配独占物理 CPU
- 配置：`cpuManagerPolicy: static` ，`full-pcpus-only: "true"`
- 行为：仅对 Guaranteed Pod 生效；Burstable/BestEffort 不受影响

Memory ManagerPolicy

- 目的：在 NUMA 节点级别预留并对齐内存
- 配置：`memoryManagerPolicy: "Static"` ，`reservedMemory`
- 行为：与 Topology ManagerPolicy 配合效果最佳

Topology ManagerPolicy

- 目的：在单个 NUMA 节点上对齐 CPU、内存和设备分配
- 配置：`topologyManagerPolicy: single-numa-node`，`topologyManagerScope: pod`
- 模式：best-effort、restricted、single-numa-node（严格）

术语

- **NUMA node**：非统一内存访问域
- **CPU pinning**：将容器绑定到专用 CPU
- **NUMA affinity**：优先使用与 CPU 同一 NUMA 节点的内存
- **Topology alignment**：将 CPU、内存和设备共置于同一 NUMA 节点
- **Guaranteed Pod**：requests = limits；CPU 以完整核数指定

Updating Public Repository Credentials

目录

Overview

`Public Repository` 是一个由平台提供的公开互联网镜像仓库服务。当您希望集群使用 `Public Repository` 作为镜像仓库时，需要更新内置的 `public-registry-credential` Cloud Credentials，以确保平台有权限从公共仓库拉取镜像。

Procedure

1. 登录 灵雀云 **Customer Portal**，并从右上角 用户信息 下拉菜单中的 企业管理 部分下载贵组织的认证文件。
2. 在 管理员 控制台左侧导航栏中，进入 集群 > **Cloud Credential**。
3. 找到名为 `public-registry-credential` 的云凭证，点击右侧下拉菜单中的 更新。
4. 在 上传公共仓库地址 部分，上传您从 灵雀云 **Customer Portal** 下载的认证文件。
5. 点击 更新 以应用更改。