节点

概览

向本地部署集群添加节点

平台管理员可以在平台管理下向自管理集群

管理节点

支持更新节点标

规划节点的原地 OS 升级

规划并验证自建集群节点上的原地操作系统

节点监控

在节点详情页查看节点监控数据。

集群节点规划

概览

节点是集群的基本构建单元。节点可以是虚拟机或物理服务器，每个节点都会运行托管 Pods 所需的组件，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添加、管理、监控和规划节点。

> **NOTE**
>
> 不支持向导入的集群中添加节点，也不支持从导入的集群中删除节点。

目录

节点类型

- 控制平面节点 运行 kube-apiserver、kube-scheduler、kube-controller-manager、etcd、容器网络以及部分平台管理组件等集群组件。
- 计算节点 承载在集群上运行的业务 Pod。请根据工作负载需求规划计算节点数量。

节点操作路径

需求	请继续执行
向集群添加节点。	添加节点
管理标签、污点、容忍度以及节点调度状态。	管理节点
监控节点状态。	监控节点
规划 infra 节点或自定义角色节点。	节点规划
为符合条件的节点规划就地操作系统升级。	就地 OS 升级
为对性能敏感的工作负载调优 CPU、内存或 topology manager 策略。	资源管理器策略

Linux 节点可用性检查

在构建本地集群之前，请完成 节点检查，并确认每个节点都满足要求。如果未满足前置条件，集群部署可能失败。

支持的操作系统和 CPU 型号

有关支持的操作系统和 CPU 型号，请参见 支持的操作系统和 CPU 型号。

向本地部署集群添加节点

当集群需要扩容，或者需要使用新节点替换集群中的异常节点时，你可以通过添加节点的方式，向平台上现有的 本地部署 工作负载集群添加控制平面节点和计算节点。

目录

约束和限制

- 在将节点添加到集群之前，请先准备好节点。使用 Node Availability Check Reference 验证节点。未满足所需条件时，集群部署可能会失败。
- 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的其他节点保持一致。
- 在同一个 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。

- 集群规划指南：一个集群至少必须有 1 个控制平面节点。不支持恰好设置 2 个控制平面节点。拥有 3 个或更多控制平面节点时，集群将成为高可用集群（对于高可用集群，建议使用奇数个节点，最好是 3 或 5 个）。注意：此要求仅适用于添加或变更控制平面容量；你可以安全地添加 worker/计算节点，而不必强制添加控制平面节点。
- 一个节点只能属于一个集群。已属于其他集群的节点不能被添加。

前提条件

- 当 global 集群无法通过 SSH 服务直接访问待添加到集群中的节点，且需要通过代理访问时，请提前准备好代理服务。目前仅支持 SOCKS5 代理。

操作步骤

1. 在左侧导航栏中，单击 **Clusters** > **Clusters**。
2. 单击要添加节点的 本地部署 类型 ***cluster name***。
3. 在 **Nodes** 选项卡下，单击 **Add Node**。
4. 配置相关参数。有关参数详情，请参见 Node Configuration Parameters。
5. 单击 **Add** 对节点执行可用性检查。
   检查通过后，节点添加开始，节点将处于 **Adding** 状态。

后续操作

查看执行进度

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

操作步骤

1. 单击处于 **Adding** 状态节点右侧的 **View Execution Progress**。
2. 在弹出的执行进度对话框中，你可以查看节点执行进度（`status.conditions`）。

提示：当某一类型正在执行，或由于某个原因处于失败状态时，你可以将光标悬停在对应原因上，查看原因的详细信息（`status.conditions.reason`，以蓝色文本显示）。

重新添加失败的节点

添加节点后，如果有部分节点添加失败，节点列表上方会出现提示。单击提示框中的 **Re-add** 按钮，重新添加失败的节点。

管理节点

目录

更新节点标签

Labels↗ 是附加到节点上的键值对，可用于定义节点属性。为节点设置标签后，您可以轻松按标签筛选或选择节点。例如：将 Pods 调度到指定节点。

支持更新正常状态节点的节点标签，以及添加或删除自定义节点标签。

操作步骤

1. 在左侧导航栏中，单击 **Cluster Management** > **Clusters**。
2. 单击包含待更新标签节点的 ***cluster name***。
3. 在 **Nodes** 选项卡下，单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. 单击 **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** 信息栏中的 **Node Labels** 项查看该节点的所有标签信息。

停止/恢复节点调度

通过设置节点的调度状态，您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，但不影响该节点上已运行的 Pods。
- **Resume Scheduling**：允许新创建的 Pods 调度到该节点。

操作步骤

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

驱逐 **Pods**

将正常状态节点上除由 DaemonSet（daemon set）管理之外的所有 Pods 驱逐到集群中的其他节点，并将该节点设为不可调度状态。

注意：驱逐会移除 Pods 中本地存储的数据。请在驱逐 Pods 之前确认数据影响。

操作步骤

1. 在左侧导航栏中，单击 **Cluster Management** > **Clusters**。
2. 单击包含待驱逐 Pods 节点的 ***cluster name***。
3. 在 **Nodes** 选项卡下，单击待驱逐 Pods 的 ***node name***。
4. 在右上角，单击 **Actions** > **Evict Pods**。
5. 查看待驱逐 Pods 的信息，然后单击 **Evict**。

设置污点

为正常状态节点设置污点信息。

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（tolerations）配合使用，可防止 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 将不会被该节点接受。

例如：如果我们发现某个节点的内存利用率已达到 91%，则不建议继续将新的 Pods 调度到该节点。此时可以为其设置污点。设置污点后，Kubernetes 将不会把 Pods 调度到该节点。

有关 Kubernetes 污点和容忍的行为，请参见 Taints and Tolerations ↗。

操作步骤

1. 在左侧导航栏中，单击 **Cluster Management** > **Clusters**。
2. 单击包含待设置污点节点的 ***cluster name***。
3. 在 **Nodes** 选项卡下，单击待设置污点节点右侧的 **Set Taints**。
4. 设置污点的键、值和效果。一个节点可以添加多个污点。
   污点属性由 `key=value [effect]` 组成。

`key=value` 用于匹配 Pod 容忍。污点表示该节点已被 `key=value` 污染，除非 Pod 能够容忍（Tolerations）`key=value` 污点，否则不允许或应避免将 Pod 调度到该节点。
effect 是污点的影响，提供以下三种选项：

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

5. 单击 **OK**。

标签和污点管理

平台支持批量为节点设置标签和污点。

约束与限制

- 在设置设备标签之前，您需要先在集群中部署设备插件，例如 NVIDIA GPU MPS device plugin、NVIDIA GPU device plugin、GPU Manager device plugin 等。
  提示：设备标签实际上是节点标签。为方便您使用，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. 在左侧导航栏中，单击 **Clusters** > **Clusters**。
2. 单击要管理标签和污点的 ***cluster name***。
3. 在 **Nodes** 选项卡下，多选要管理的节点，然后单击 **Label and Taint Management** 按钮。
   提示：您可以在节点列表页的搜索框中输入您关心的节点标签，快速筛选出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，然后单击 **OK** 将批量操作提交到集群。

   - **Node Labels**：可以为所选节点 添加/更新 指定标签，或 删除 指定标签。选择删除时，平台会筛选出所选节点上的所有标签列表。值设置为 **Any** 时，表示删除包含指定标签键的所有节点上的标签。

- **Taints**：可以为所选节点 添加/更新 指定污点，或 删除 指定污点。选择删除时，平台会筛选出所选节点上的所有污点列表。值设置为 **Any** 时，表示删除包含指定污点键的所有节点上的污点。
- **Device Labels**：可以为所选节点设置要使用的设备，设备列表来自您已在该集群中部署的设备插件。

启用/禁用虚拟化开关

当裸金属集群中的节点为物理机时，您可以通过启用/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。

当开关启用时，允许将新创建的虚拟机调度到该物理机节点；当开关禁用时，禁止将新创建的虚拟机调度到该物理机节点，但这不会影响已经在该节点上运行的虚拟机。

提示：相关操作和注意事项，请参见 Prepare Virtualization Environment。

删除裸金属集群节点

支持删除裸金属类型集群中的节点。例如：删除裸金属集群中的故障节点。

约束与限制

- 不支持删除导入集群中的节点。
- 当集群中只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. 在左侧导航栏中，单击 **Cluster Management** > **Clusters**。
2. 单击包含待删除节点的 **On-Premises** 类型 ***cluster name***。
3. 在 **Nodes** 选项卡下，单击待删除节点右侧的 **Delete**。
   提示：如果在删除 Linux 节点后需要清理该节点下的资源，可单击对话框底部的 **Download Cleanup Script**，将清理脚本下载到本地。节点成功删除后，登录该节点并执行清理脚本。
4. 输入节点名称，然后单击 **Delete**。

规划节点的原地 OS 升级

集群管理员可以使用此检查清单，为 ACP 自建集群中的节点规划原地操作系统升级。在操作系统变更前后，完成 ACP 检查。

> **范围和支持边界**
>
> 在维护窗口期间，将此检查清单与操作系统提供商支持的升级操作步骤结合使用。在升级任何节点之前，请先根据 ACP 支持矩阵确认目标操作系统和内核。该检查清单不能保证每一条操作系统升级路径都受支持。

目录

范围和限制

仅当满足以下所有条件时，才使用此操作步骤：

- 集群是由 ACP 管理的本地部署集群或自建集群。
- 你可以登录到目标节点并管理节点操作系统。
- 你具有平台管理员权限、`kubectl` 管理员权限，以及对每个目标节点的 SSH 或控制台访问权限。
- 在操作系统升级之前，你可以从目标节点驱逐工作负载 Pod。

以下类型的集群不要使用此操作步骤：

- 使用 Immutable Infrastructure 的集群。请通过使用新镜像替换节点来应用操作系统变更。
- 由云提供商管理节点或控制平面操作系统的托管云 Kubernetes 集群。
- 你无法登录到节点或控制平面节点的导入集群。

升级前

在对任何节点更改操作系统之前，先完成以下检查。

1 步骤 1：确认目标操作系统和内核

确认目标操作系统版本、内核版本、内核来源以及 CPU 架构均在受支持范围内。有关当前支持矩阵和已知限制，请参阅 受支持的 OS 和内核版本。

在维护窗口之前应用以下规则：

- 操作系统主版本和次版本必须与支持矩阵一致。
- 核心内核版本必须与支持矩阵一致。仅允许构建后缀不同。
- 内核必须是操作系统厂商提供的官方内核。
- 不支持 Ubuntu HWE 内核以及第三方或自行编译的内核。
- 如果目标版本不在支持矩阵中，请在升级前联系技术支持确认兼容性。

2 步骤 2：检查冲突软件包

在原地操作系统升级期间，操作系统包管理器可能会安装或覆盖容器运行时、Kubernetes 或容器网络二进制文件。升级前，请检查并解决与 ACP 组件冲突的软件包。有关软件包列表和命令，请参阅 移除冲突软件包。

如果发现冲突软件包，请在卸载之前制定应用迁移方案并备份受影响的数据。

3 步骤 3：记录当前运行时和节点组件版本

在操作系统升级前记录版本信息。升级节点后，使用这些记录进行对比。

```
containerd --version
runc --version
crictl --version
kubelet --version
```

同时记录操作系统升级流程可能会更新的关键节点配置文件，例如 `/etc/resolv.conf`、`/etc/fstab`、systemd 配置文件以及容器运行时配置文件。

4 步骤 4：验证集群容量并驱逐节点

确认其余节点有足够容量来运行从目标节点驱逐的 Pod。然后在操作系统升级前驱逐该节点。

你可以使用 ACP 控制台从节点驱逐 Pod。有关控制台操作，请参阅 管理节点。

如果你使用 `kubectl`，请在运行前与运维团队确认命令选项。例如：

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

由 DaemonSet 管理的 Pod 不会被驱逐操作驱逐。使用本地存储的工作负载在驱逐后可能会丢失本地数据。继续之前，请先确认工作负载影响。

控制平面节点的额外检查

控制平面节点运行 `kube-apiserver`、`etcd`、`kube-scheduler` 和 `kube-controller-manager` 等组件。请一次升级一个控制平面节点，并在每个节点升级后验证集群健康状态。

在升级控制平面节点之前：

- 备份 etcd 数据。有关受支持的备份机制和恢复注意事项，请参阅 etcd 备份与恢复。
- 确认 etcd 集群处于健康状态，并且在一个控制平面节点不可用时仍可维持法定人数（quorum）。
- 在对控制平面节点执行滚动维护时，确认集群至少有三个控制平面节点。
- 如有可能，先在计算节点上验证该操作步骤，然后再继续处理控制平面节点。

在检查控制平面健康状态时，你可以使用以下命令作为参考：

```
kubectl get nodes
kubectl get pods -n kube-system | grep -E "etcd|apiserver|scheduler|contr
oller"
```

如果你在控制平面节点上使用 `etcdctl`，请使用你环境中的证书路径：

```
ETCDCTL_API=3 etcdctl member list \
  --endpoints=https://127.0.0.1:2379 \
  --cacert=/etc/kubernetes/pki/etcd/ca.crt \
  --cert=/etc/kubernetes/pki/etcd/server.crt \
  --key=/etc/kubernetes/pki/etcd/server.key \
  --write-out=table
```

执行操作系统升级

请按照操作系统厂商针对目标版本提供的支持升级操作步骤进行操作。具体的软件包命令、仓库配置和重启要求由操作系统厂商以及你所在组织的操作系统维护策略决定。

在操作系统升级期间：

- 不要有意安装与 ACP 组件冲突的容器运行时、Kubernetes 或容器网络软件包。
- 除非厂商操作步骤要求更改，否则请保留节点网络配置、DNS 配置、时间同步配置以及 systemd 服务配置。
- 当厂商操作步骤要求时，重启节点。
- 在所有升级后检查通过之前，保持节点不可调度。

升级后

在每个节点完成操作系统升级和重启后，运行以下检查。

1 步骤 1：确认操作系统和内核

验证节点正在运行预期的操作系统和内核版本。

```
cat /etc/os-release
uname -r
```

将输出结果与支持矩阵及你批准的维护计划进行比较。

2 步骤 2：验证基础节点配置

验证操作系统升级没有还原所需的节点设置。

```
echo "=== Base node configuration check ==="

# SELinux must be disabled on systems that use SELinux.
getenforce 2>/dev/null || echo "SELinux command not available"

# AppArmor must be disabled on systems that use AppArmor.
systemctl status apparmor 2>/dev/null || echo "AppArmor service not i
nstalled"

# Swap must be disabled.
swapon --show
free -h | grep Swap

# Firewall services must be disabled according to your cluster networ
k plan.
systemctl status firewalld 2>/dev/null || echo "firewalld not install
ed"
systemctl status ufw 2>/dev/null || echo "ufw not installed"

# /tmp must not be mounted with noexec.
mount | grep " /tmp "

# DefaultTasksMax must be infinity.
systemctl show --property=DefaultTasksMax
```

如果操作系统升级后启用了 `swap`，请根据你的操作系统策略将其禁用，并从 `/etc/fstab` 中移除 swap 条目。

如果操作系统升级后启用了 SELinux 或 AppArmor，请根据你的操作系统策略以及 ACP 的节点要求将其禁用。

3 步骤 3：验证运行时和节点组件版本

将运行时和节点组件版本与升级前记录进行比较。

```
containerd --version
runc --version
crictl --version
kubelet --version
```

然后验证 `containerd` 正在运行：

```
systemctl status containerd
```

如果有任何二进制文件被操作系统软件包覆盖，请停止维护，并在继续处理其他节点之前联系技术支持。

4 步骤 4：验证时间同步和 DNS

验证操作系统升级未更改时间同步和 DNS 配置。

```
date
timedatectl
cat /etc/resolv.conf
```

节点之间的时间偏差必须不超过 10 秒。如果偏差大于 10 秒，请先同步时间，再恢复节点上的工作负载调度。

5 步骤 5：重启 kubelet 并验证节点恢复

在操作系统升级完成且基础节点检查通过后，重启 `kubelet`。

```
systemctl daemon-reload
systemctl restart kubelet
systemctl status kubelet
```

等待节点恢复到 `Ready` 状态。

```
kubectl get node <node-name> --watch
```

当节点恢复健康后，恢复调度。

```
kubectl uncordon <node-name>
```

你也可以从 ACP 控制台恢复调度。有关控制台操作，请参阅 管理节点。

6 步骤 6：验证工作负载恢复

在恢复调度后，验证节点状态和工作负载放置情况。

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

然后验证受节点驱逐影响的业务服务。只有在当前节点及相关服务都处于健康状态后，才能继续处理下一个节点。

已知高风险场景

症状	可能原因	检查方法	建议操作
`containerd` 启动失败	操作系统升级覆盖了 ACP 运行时二进制文件	将 `containerd --version` 与升级前记录进行比较	停止维护并联系技术支持
节点保持 `NotReady`	`kubelet`、容器运行时、CNI、DNS 或节点网络配置发生了更改	检查 `systemctl status kubelet`、`systemctl status containerd` 和节点事件	恢复已更改的配置，或联系技术支持
集群组件报告 TLS 错误	节点在升级期间或升级后发生了时间漂移	检查 `timedatectl` 并在节点之间比较 `date`	在继续之前先同步时间
DNS 解析失败	`/etc/resolv.conf` 被覆盖	检查 `cat /etc/resolv.conf`	恢复已批准的 DNS 配置
`kubelet` 因安全策略错误而失败	SELinux 或 AppArmor 被重新启用	检查 `getenforce` 或 `systemctl status apparmor`	根据节点要求禁用该服务
执行 uncordon 后工作负载无法调度	节点仍然不健康、带有污点，或资源受限	检查 `kubectl describe node <node-name>`	在继续之前先解决节点状态问题

操作检查清单

在维护窗口期间使用此检查清单，并在升级后保留已完成记录。

阶段	项目	负责人	状态
升级前	确认目标操作系统和内核在支持矩阵中		
升级前	确认内核是厂商官方内核		
升级前	检查并解决冲突软件包		
升级前	记录 `containerd`、`runc`、`crictl` 和 `kubelet` 版本		
升级前	记录 DNS、时间同步、`/etc/fstab` 和运行时配置		
升级前	确认集群有足够容量承载已驱逐的工作负载		
升级前	驱逐目标节点并确认工作负载影响		
仅控制平面节点	备份 etcd 数据		
仅控制平面节点	确认 etcd 健康状态和法定人数		
升级	执行操作系统厂商支持的升级操作步骤		
升级	如厂商操作步骤要求，重启节点		
升级后	确认操作系统和内核版本		
升级后	确认 SELinux 或 AppArmor 按要求已禁用		
升级后	确认 swap 已禁用		
升级后	确认防火墙服务与集群网络规划一致		

阶段	项目	负责人	状态
升级后	确认 `/tmp` 未以 `noexec` 方式挂载		
升级后	确认 `DefaultTasksMax` 为 `infinity`		
升级后	确认运行时和节点组件版本未被覆盖		
升级后	确认 `containerd` 和 `kubelet` 运行正常		
升级后	确认节点之间的时间偏差不超过 10 秒		
升级后	确认 DNS 配置正确		
升级后	确认节点处于 `Ready` 状态		
升级后	恢复节点上的调度		
升级后	确认工作负载和业务服务健康		

节点监控

在节点详情页查看节点监控数据。

TIP

- 当集群包含多个节点时，您可以在节点详情页的资源路径区域单击 *当前节点名称*，展开节点下拉列表，然后单击选择一个节点，以便快速切换到其他节点详情页。
- 当为集群配置了监控组件时，您可以查看节点监控数据，包括资源运行状态、资源使用情况和资源趋势统计。

目录

操作步骤

1. 在左侧导航栏中，单击 **Clusters** > **Clusters**。
2. 单击目标节点所在的 *集群名称*。
3. 在 **Nodes** 选项卡下，单击目标 *节点名称*。
4. 单击 **Monitoring** 选项卡，进入节点监控数据展示页面并查看相关节点监控数据。

TIP

- 将鼠标悬停在卡片上并单击 详情 图标，以查看 PromQL 表达式；单击 导出 图标可导出当前页面所有图表的 PromQL 表达式。
- 当集群包含多个节点时，您可以在节点详情页的资源路径区域单击 *当前节点名称*，展开节点下拉列表，然后单击选择一个节点，以便快速切换到其他节点详情页。

TIP

在存储空间统计展示区域中，当节点存在超过 4 个存储分区时：

- 在分区总使用量饼图中，使用量最高的前 3 个分区会单独显示，其余分区会显示为 其他，并在悬停到该区域时显示其总使用量数据；
- 在分区使用量条形图中，使用量最高的前 3 个分区会单独显示，其余分区会显示为 其他，并在悬停到条形图上时显示其总使用量和各自的使用率。

监控趋势统计如下表所述。

参数	描述
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 / 节点的 CPU 总量。 说明：如果某一时间段内节点的 CPU 使用率出现突增，您需要先找出占用 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都可能导致 CPU 使用率过高。 请求率 = 节点上所有 Pod 的 CPU requests / 节点的 CPU 总量。 说明：如果某一时间段内节点的 CPU 请求率出现突增，可能是由于集群超分配比设置不合理，或者节点上运行的 Pod 的 request 值过高，从而导致资源浪费。 限制率 = 节点上所有 Pod 的 CPU limits / 节点的 CPU 总量。 说明：如果某一时间段内节点的 CPU 限制率出现突增，表明节点上运行的 Pod 的 limit 值设置过高，可能会导致 CPU 资源浪费。
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 / 节点的总内存。 内存是服务器上的重要组件之一，也是 CPU 进行通信的桥梁。因此，内存性能对机器有显著影响。程序运行时，数据加载、线程并发以及 I/O 缓冲都依赖内存。可用内存大小决定了程序能否正常运行以及运行方式。 请求率 = 节点上所有 Pod 的内存 requests / 节点的总内存。 说明：如果某一时间段内节点的内存请求率出现突增，可能是由于集群超分配比设置不合理，或者节点上运行的 Pod 的 request 值过高，从而导致资源浪费。 限制率 = 节点上所有 Pod 的内存 limits / 节点的总内存。 说明：如果某一时间段内节点的内存限制率出现突增，表明节点上运行的 Pod 的 limit 值设置过高，可能会导致内存资源浪费。
Storage	指定时间范围内的空间使用率和 **inode** 使用率。

参数	描述
	空间使用率 = 已使用存储空间 / 存储空间总量。 通过监控历史磁盘空间数据，您可以评估某一时间段内的磁盘使用情况。当磁盘使用率较高时，您可以通过清理不必要的镜像或容器来释放磁盘空间。 **inode** 使用率 = 已使用 inode 存储 / inode 存储总量。 说明：每个文件都必须拥有一个 inode 来存储文件元数据，例如文件创建者和创建日期。inode 也会占用磁盘空间，且大量小型缓存文件容易导致 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。该值是当前由 CPU 正在执行和等待执行的进程总数与 CPU 可执行的最大进程数之比，是衡量系统繁忙/空闲状态的重要指标。 说明：如果在某一时间段内 1 分钟/5 分钟/15 分钟曲线相近，说明集群的 CPU 负载相对稳定。 如果在某一时间段或某一时间点 1 分钟值远大于 15 分钟值，说明最近 1 分钟内负载在上升，需要持续观察。一旦 1 分钟值超过 CPU 数量，可能表示系统过载，您需要进一步分析问题根因。 如果在某一时间段或某一时间点 1 分钟值远小于 15 分钟值，说明最近 1 分钟内系统负载在下降，而前 15 分钟内曾产生较高负载。
Disk Throughput	指定时间范围内的磁盘吞吐量，指磁盘传输数据流的速度，其中传输数据为读写数据之和。
Disk IOPS	指定时间范围内的磁盘 IOPS 为每秒连续读写次数之和，表示磁盘每秒读写操作数量的性能指标。
Network Traffic Rate	指定时间范围内的网络流入和流出速率，由节点的物理网卡统计。
Network Packet Rate (packets/sec)	指定时间范围内的网络收发包速率，由节点的物理网卡统计。

集群节点规划

集群使用格式为 `node-role.kubernetes.io/<role>` 的 Kubernetes 节点角色标签为节点分配不同角色。在本文中，`role label` 指的是这类标签。

默认情况下，集群包含两类节点：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：

- 控制平面节点会被标记为角色标签 `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。为了向后兼容，这两个标签都被视为用于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。不过，如果需要，你也可以显式为工作节点分配角色标签 `node-role.kubernetes.io/worker`。

除了这些默认角色标签之外，你还可以在工作节点上定义自定义角色标签，以便将其进一步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，用于专门承载日志组件。

结合角色标签和 taint 创建 infra 节点或自定义角色节点，然后在工作负载隔离需要时，将选定的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。如果你希望将某些工作节点指定为专用于承载基础设施组件的 infra 节点，则需要手动为这些节点添加相应的角色标签和 taint。

> 注意：
> 本节中的操作仅适用于非不可变集群。不要将这些操作用于由云提供商管理的集群、第三方集群，或节点使用不可变 OS 的集群。

添加 **Infra** 节点

步骤 **1**：为 **Node** 资源添加 **Infra** 角色标签

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

该命令会向 Node 192.168.143.133 添加 infra 角色标签：`node-role.kubernetes.io/infra: ""`，表示该节点是 infra 节点。

步骤 2：为 Node 资源添加 Taint

添加 taint 以防止其他工作负载被调度到 infra 节点上。

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

该命令会向 Node 192.168.143.133 添加 taint `node-role.kubernetes.io/infra=reserved:NoSchedule`，表示只有容忍此 taint 的应用才能被调度到该节点上。

步骤 3：验证标签和 Taint

检查节点是否已被分配 infra 角色标签和 taint：

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

输出表明 Node `192.168.143.133` 已配置为 infra 节点，并带有 `node-role.kubernetes.io/infra=reserved:NoSchedule` taint。

将 Pod 迁移到 Infra 节点

如果你希望将特定 Pod 调度到 infra 节点上，需要进行以下配置：

- 指向 infra 角色标签的 nodeSelector。
- 与 infra 节点 taint 对应的 toleration。

下面是一个配置为在 infra 节点上运行的 Pod 清单示例。

```
apiVersion: 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 确保该 Pod 只会被调度到带有 `node-role.kubernetes.io/infra: ""` 标签的节点上，而 toleration 允许该 Pod 容忍 taint `node-role.kubernetes.io/infra=reserved:NoSchedule`。

通过这些配置，Pod 将被调度到 infra 节点上。

> 注意：
> 通过 OLM Operators 或 Cluster Plugins 安装的 pods 并不总是可以迁移到 infra 节点。能否移动这些 pods 取决于各个 Operator 或 Cluster Plugin 的配置。

自定义节点规划

除了 infra 节点之外，你可能还希望为其他专用用途指定工作节点，例如承载日志组件、存储服务或监控 agent。

你可以通过为工作节点分配更多自定义角色标签及其对应的 taint，来实现这一点，从而有效地将它们转变为自定义角色节点。

定义自定义角色节点的一般步骤

该过程与创建 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，例如 monitoring、storage 或 log。

步骤 2：添加对应的 Taint

```
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，并将 <value> 替换为有意义的描述，例如 reserved 或 dedicated。该值是可选的，但可以帮助运维人员理解调度意图。

步骤 3：验证配置

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

确保 Labels 和 Taints 字段反映了你的自定义角色配置。

示例：创建一个专用于日志组件的节点

如果你想创建一个专门用于安装日志组件的节点，可以添加 log 角色。在这种情况下，按如下方式创建 log 节点。

步骤 1：添加 Log 角色标签

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

该标签表示此节点被指定用于与日志相关的工作负载。

步骤 2：为节点添加 Taint

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

该 taint 可防止未被调度的工作负载部署到该节点上。

步骤 3：验证标签和 Taint

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

这将确认该节点已成功配置为带有相应标签和 taint 的 log 节点。

使用角色标签和 taint 按用途对 Kubernetes 节点进行分区，提升工作负载隔离能力，并将选定组件调度到配置适当的节点上。