容器服务

TKE 边缘集群指南

产品文档

腾讯云

文档目录

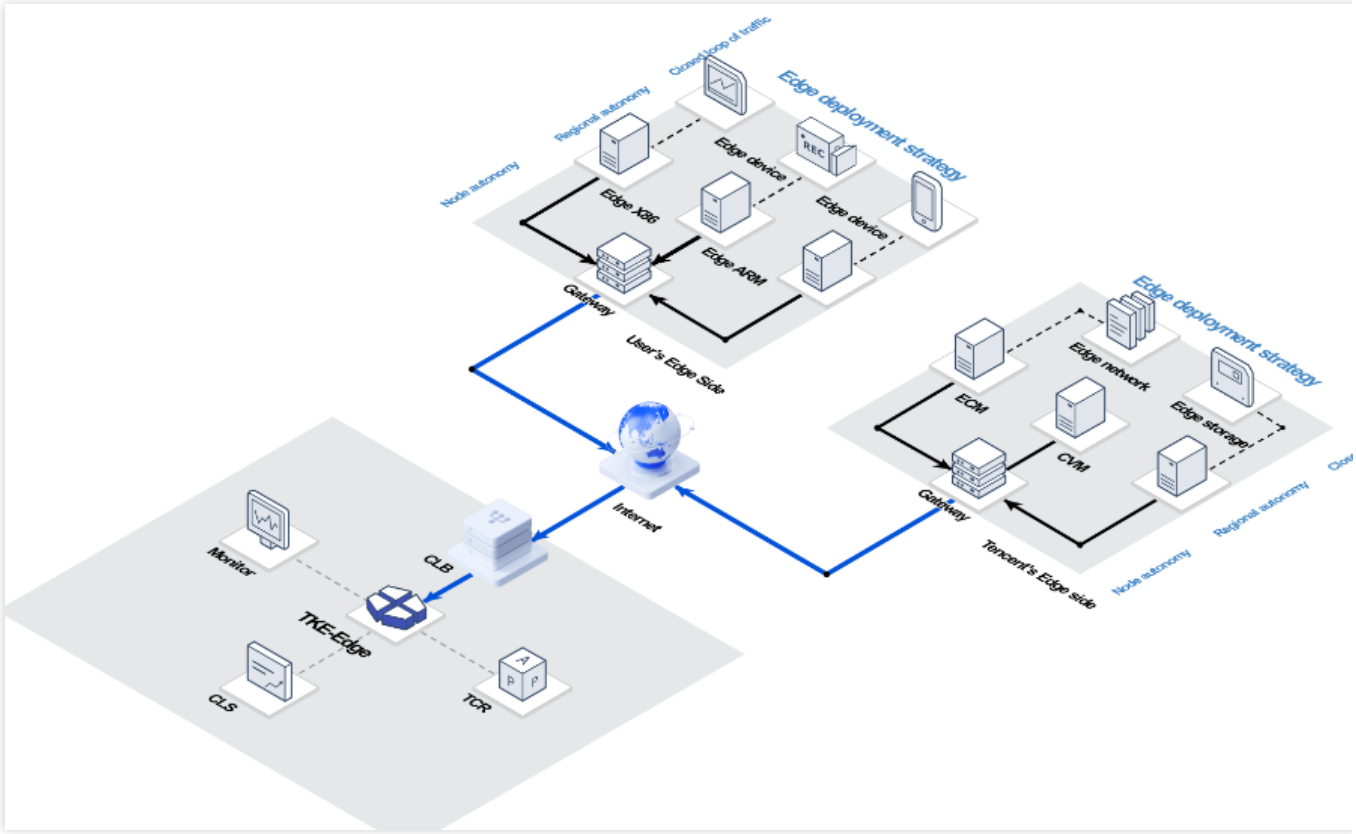
TKE 边缘集群指南
TKE 边缘集群概述

最近更新时间：2023-06-01 11:22:54

简介

腾讯云 TKE 边缘容器服务是腾讯云推出的通过公有云托管控制面纳管边缘资源的容器系统，针对边缘计算场景实现云边一体的云原生能力支持。TKE 边缘容器完全兼容原生 Kubernetes、支持在同一个集群中管理位于多个机房的节点、一键将应用下发到所有边缘节点，并且具备边缘自治和分布式健康检查能力。

边缘容器架构



边缘容器服务功能介绍

云原生支持

腾讯云边缘容器服务基于开源 SuperEdge 项目构建，紧跟社区，支持最新的 Kubernetes 版本及原生的 Kubernetes 集群管理方式，开箱即用，只需要极低的学习成本。

控制面高可用

腾讯云边缘容器服务基于腾讯云 TKE 核心产品托管 Master 控制面，位于任意地点的 worker 节点可以一键接入控制面即可开始提供服务，极大提高集群可用性。

云上能力下沉

腾讯云边缘容器服务可以无缝复用 TKE 产品的大部分支撑能力，例如监控、日志、应用商店等，快速将公有云标准能力下沉到用户侧。

异构资源支持

接入的边缘节点支持各类型异构资源，包括腾讯云提供的 CVM/ECM 机器，用户侧自有的 IDC 机房服务器、PC 机器、IoT 设备等，同时支持 X86、ARM64 以及 ARM 体系架构，支持应用在不同类型资源上混合部署。

多地域部署能力

支持将边缘节点按照所需特性（例如地域等）划分为不同节点池，支持在多个节点池上批量进行应用调度部署，同时提供节点池内流量闭环访问能力。

安全可靠

腾讯云边缘容器服务支持内外网证书分离，节点权限最小化控制来最大限度避免集群访问控制权限泄露问题。云边通信使用 TLS 加密，防止系统管理数据泄露或被篡改。

系统容灾

除了控制面高可用外，边缘容器服务针对云边弱网络场景提供了可靠的边缘自治能力，并且具备分布式集群健康检查能力，可更准确地判定 pod 迁移时机。

云边隧道

借助于腾讯多年积累的隧道技术，即使边缘设备不具备公网地址，边缘容器服务也可以支持管理员从云端直接登录运行在边缘节点上的容器。

TKE 边缘集群定价

TKE-Edge 的国际站点暂时免费。 由于计算节点由用户自行提供，因此不会产生额外费用。

应用场景

边缘计算场景

边缘容器服务针对类似工业互联网、边缘 IoT、边缘 AI 相关客户需求，支持客户快速创建云端控制面，将边缘侧多种类型设备进行批量统一管控，用户可以在云上对应用进行生命周期管理。

混合云场景

边缘容器服务针对大型客户混合云场景需求，可以快速在用户 IDC 侧交付标准 K8s 集群以及云上标准服务，同时和云端 SaaS 服务无缝集成，降低部署和运维实施成本，同时提供公有云级别的高可用性支持。

相关服务

如果您需要通过调用腾讯云 API 来访问腾讯云的产品和服务，请前往 腾讯云 API 文档。

边缘集群管理
创建集群

最近更新时间：2023-06-01 11:22:54

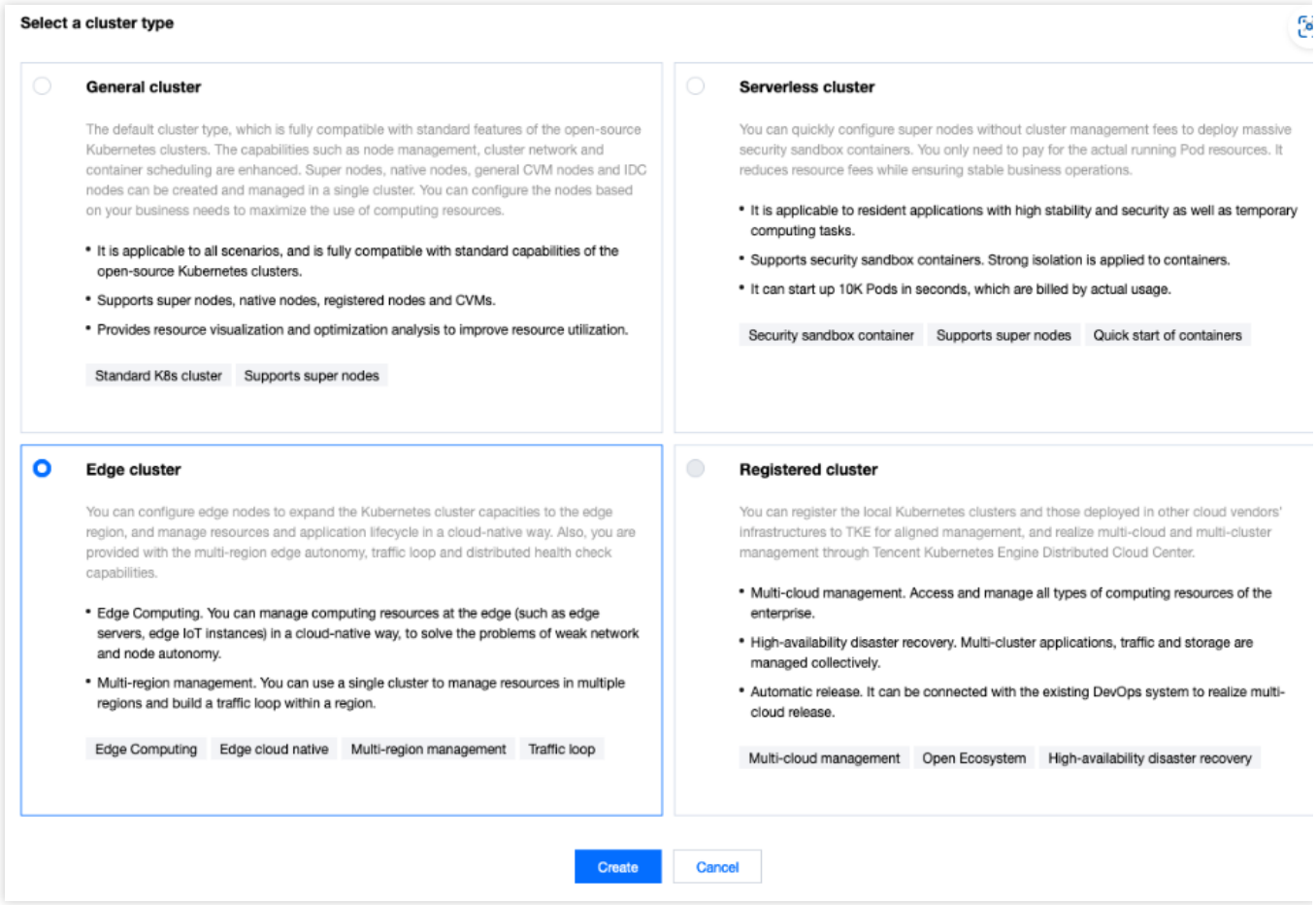
本文介绍如何通过腾讯云容器服务控制台创建 TKE 边缘集群。

容器网络说明

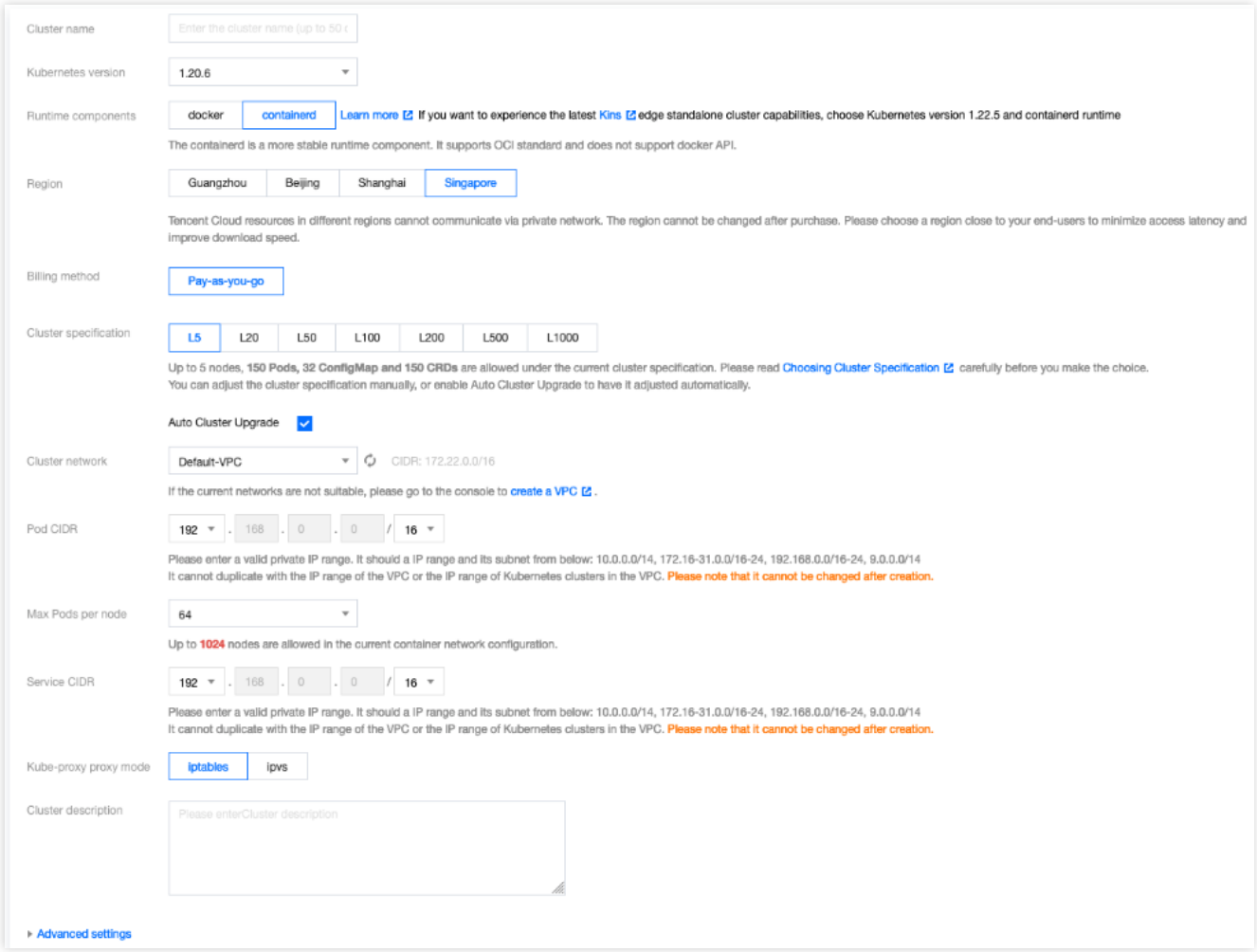
TKE 边缘容器服务将会利用节点侧网络搭建 overlay 网络，请确保集群网络（控制面所在VPC 网络）、容器网络（Pod CIDR/Service CIDR）不会与边缘节点侧的节点内部网络存在冲突。

操作步骤

1. 登录 腾讯云容器服务控制台。
2. 在集群管理页面，单击**新建**。
3. 选择**边缘集群**，单击**创建**。



4. 在**创建边缘集群**页面，根据以下信息创建边缘集群。如下图所示：



集群名称：创建的边缘集群名称，不超过50个字符。

Kubernetes版本：目前支持 Kubernetes 1.18.2, 1.20.6，1.22.5 版本，后续会随着 Kubernetes 社区发布的版本持续更新。

运行时组件：支持 Docker 运行时、Containerd 运行时，更多信息请参考 如何选择 Containerd 和 Docker。

所在地域：当前支持**北京、上海和广州**地域。

计费方式：当前边缘容器支持**按量计费**。

集群规格：您可以根据需求，选择不同集群规格。不同规格对于接入节点数量，pod、configmap 和 CRD 数目均有限制，详情请参考 购买集群配额限制。

自动升配：如果使能该功能，当集群资源超过当前规格最大限制后，会自动将集群规格提升到更高一档并进行计费；后期您可以回收资源，平台不会自动进行降档，您可以手动将规格调整回合适的更低档次。如果未使能该功能，当集群资源超过当前规格最大限制后，新添加的节点将会处于"已封锁"状态，不允许应用调度到该节点，您只有手动提升集群规格后，节点才会自动恢复正常。

集群网络：选择您的腾讯云账号下的 VPC 网络，K8s Master 控制面会在您选择的 VPC 下创建，后续您可以在同样的 VPC 下添加对应的云端计算节点。

Pod CIDR：您需要根据边缘节点内部网络管理情况为集群分配 Pod 网络，请事先规划集群规模，为容器网络分配 IP 数量足够的网段。**不能与 VPC 及 VPC 内已有 Kubernetes 集群使用的网段重复，也不能与边缘节点内网 IP**

CIDR 重复，创建成功后不可修改。

Service CIDR：您需要根据边缘节点内部网络管理情况为集群分配 Service 网络，请事先规划集群规模，为服务网络分配 IP 数量足够的网段。**不能与私有网络及私有网络内已有 Kubernetes 集群使用的网段重复，也不能与边缘节点内网 IP CIDR 重复， 创建成功后不可修改**。

Kube-proxy 代理模式：您可以选择 iptables 模式和 ipvs 模式。

集群描述：创建集群的相关信息。该信息将显示在“集群信息”页面。

5. 单击**完成**，即可完成集群 Master 组件创建，可在“边缘集群”列表页面查看集群的创建进度。

节点管理

注意

集群创建完成后，需要开启集群的内网/外网访问后才能正常对外提供服务，详情请参考 集群开启内外网访问。

集群开启内外网访问

最近更新时间：2023-06-01 11:22:54

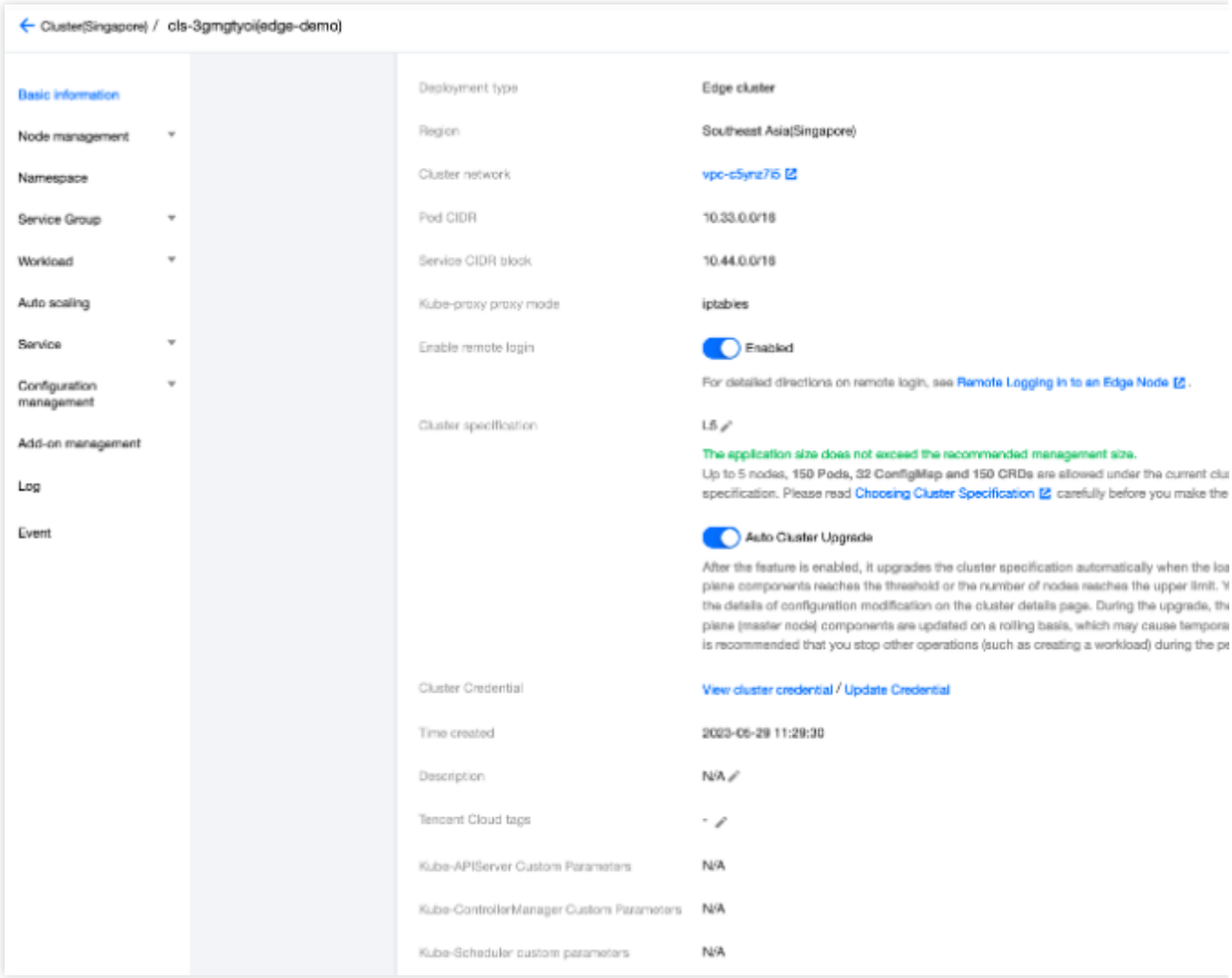
使能添加节点能力

如果您需要使能添加节点能力，首先需要将集群 APIServer 服务对外开放访问，这里包括开放**内网访问**（VPC 内）和开放**外网访问**（对互联网）两种不同类型。

使能外网访问

外网访问是指用户的节点存在于腾讯云公有云体系之外，但是可以通过各种类型互联网访问腾讯云服务，这里的情形适用于大部分边缘节点。

1. 登录 容器服务控制台，选择左侧导航栏中的**集群**。
2. 在集群列表，单击边缘集群 ID，进入该集群详情页。
3. 在集群“基本信息”中，定位到“集群 APIServer 信息”处，使能**外网访问**。

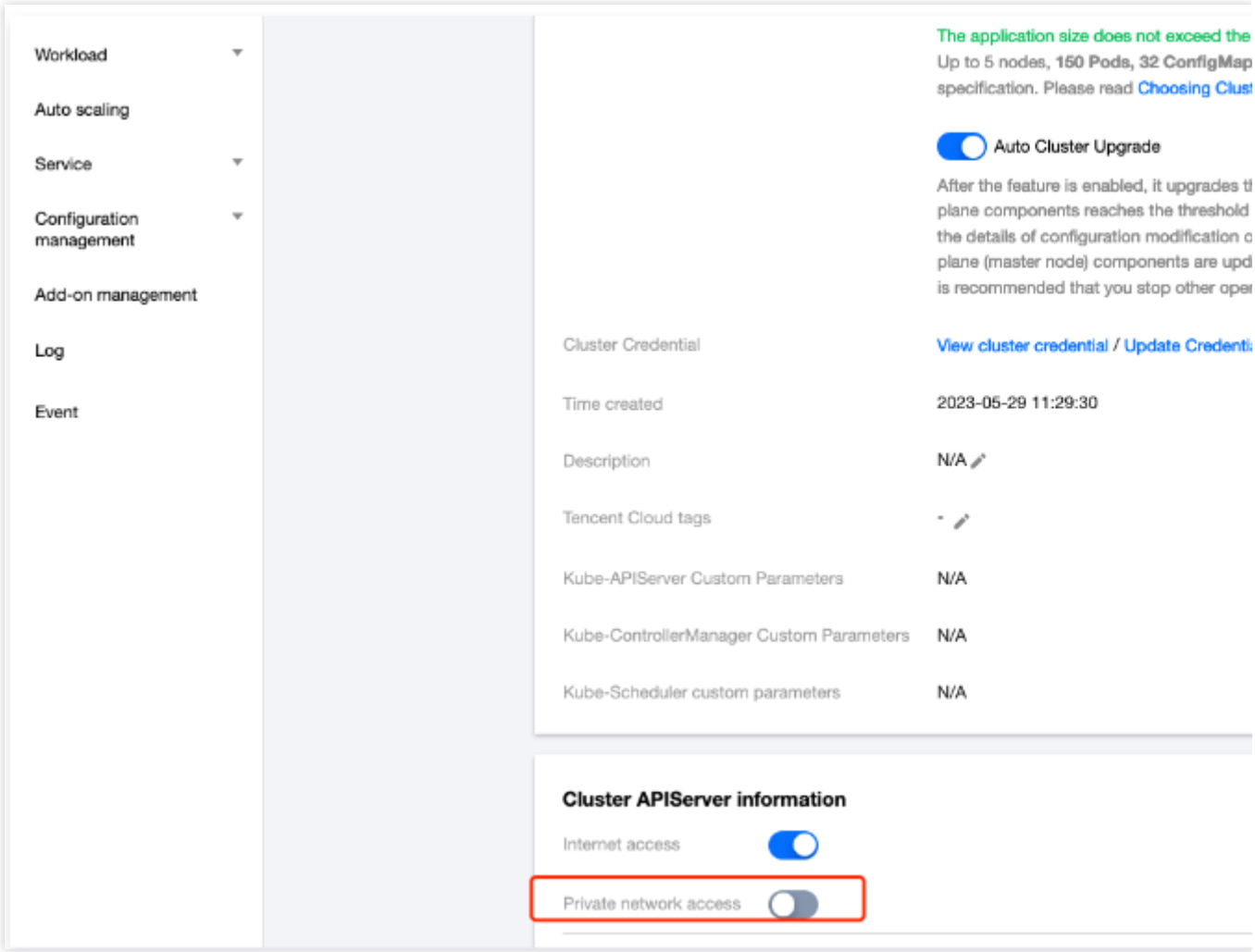


使能成功后，用户即可通过互联网链路添加 ECM、CVM 以及第三方节点（脚本添加）。

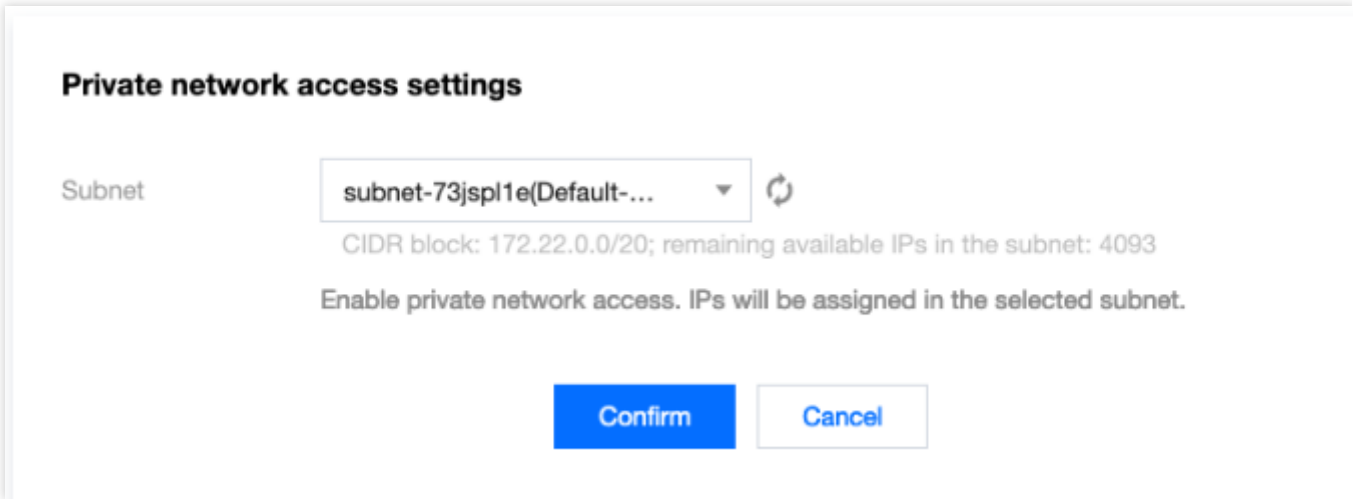
使能内网访问

内网访问为了方便用户添加集群同 VPC 下的 CVM 机器，通过 VPC 内网注册到边缘集群，从而节省公网流量。边缘容器提供了**内网添加节点**的能力，您可参考以下步骤进行使用。

1. 登录 容器服务控制台，选择左侧导航栏中的**集群**。
2. 在集群列表，单击边缘集群 ID，进入该集群详情页。
3. 在集群“基本信息”中，定位到“集群 APIServer 信息”处，使能**内网访问**。



4. 在“内网访问设置”中，选择集群所在 VPC 下的子网，您需要选择特定的子网（这里的子网指后续添加的 CVM 所在的子网）。



5. 单击**确定**后，即可使用脚本通过内网添加本 VPC 下的 CVM 节点。

注意

使能**内网访问**后，在节点页面只能通过**脚本添加节点**方式添加 VPC 内的 CVM 节点，无法使用**创建 ECM 节点**和**创建 CVM 节点**能力。

后续操作

内网/外网访问能力开启后，您就可以通过以下方式为集群添加节点了：

创建 ECM 节点

创建 CVM 节点

脚本添加第三方节点

连接集群

最近更新时间：2023-06-01 11:22:54

操作场景

您可以通过 Kubernetes 命令行工具 Kubectl 从本地客户端机器连接到 TKE 边缘集群。本文档指导您如何连接集群。

前提条件

cURL 软件程序已安装

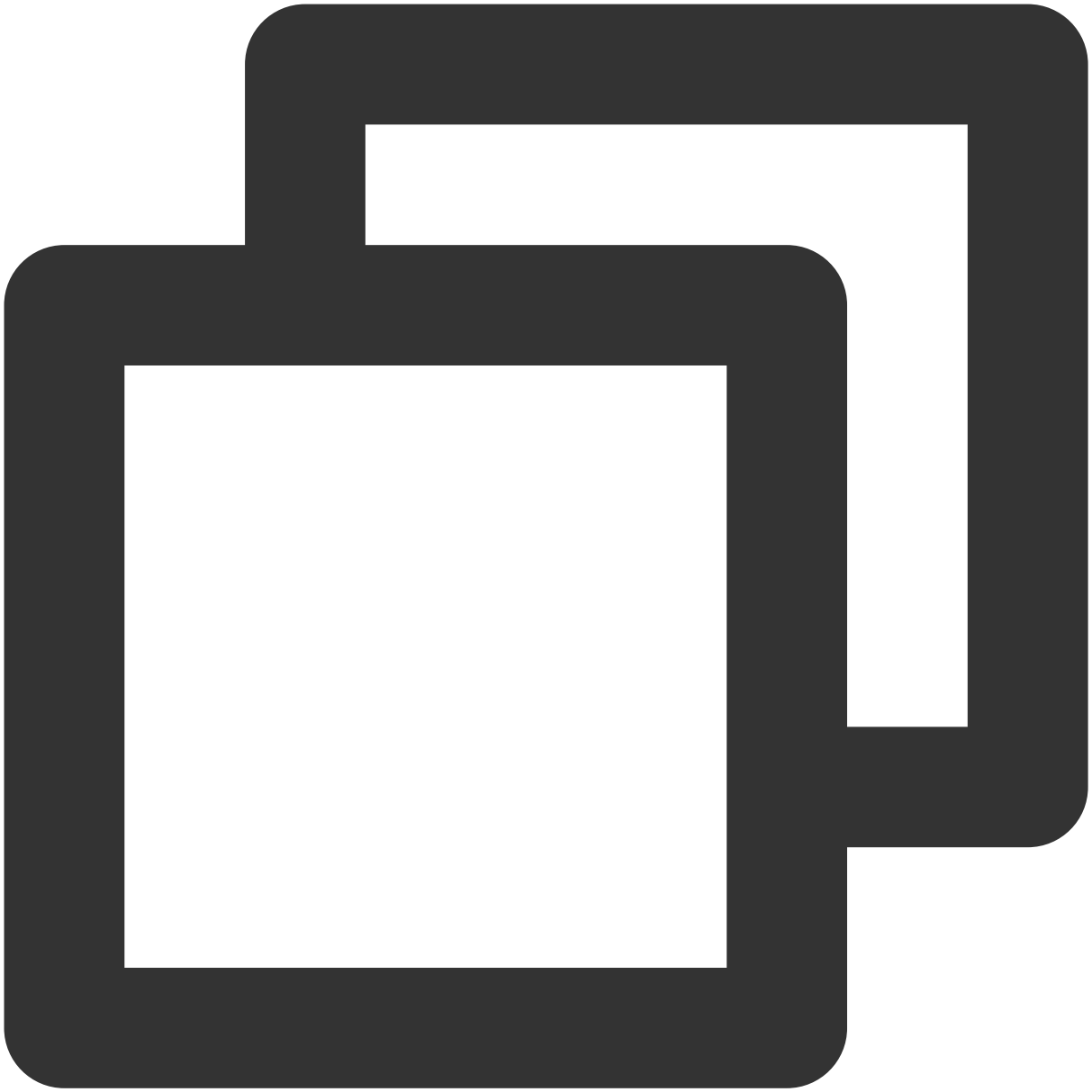
根据操作系统选择适当的方式获取 kubectl

注意：

将命令中的 v1.18.4 替换为您的业务所需的 kubectl 版本。

MacOS X

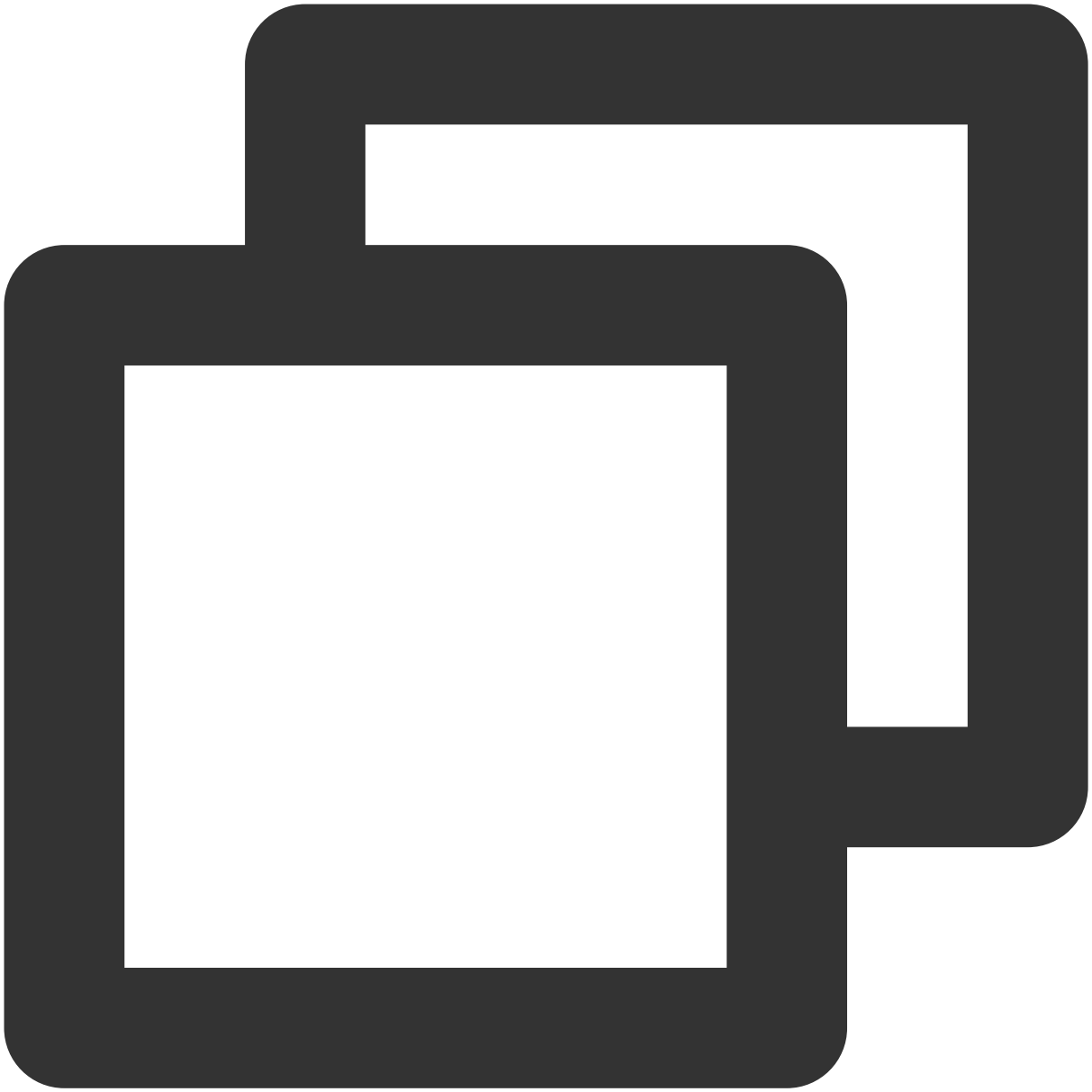
运行下面的命令获取 kubectl



```
curl -LO https://storage.googleapis.com/kubernetes-release/release/v1.18.4/bin/da
```

Linux

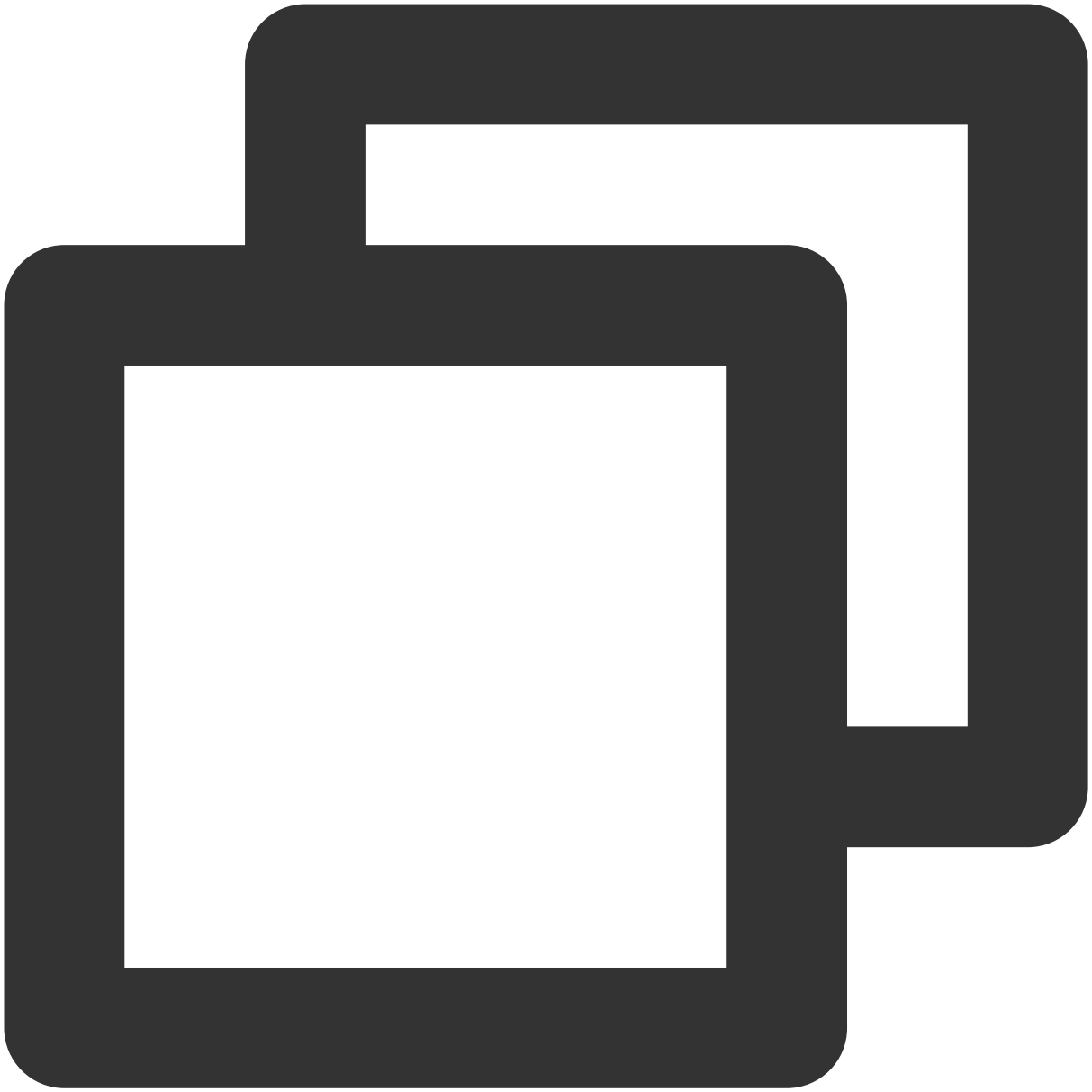
运行下面的命令获取 kubectl



```
curl -LO https://storage.googleapis.com/kubernetes-release/release/v1.18.4/bin/li
```

Windows

运行下面的命令获取 kubectl

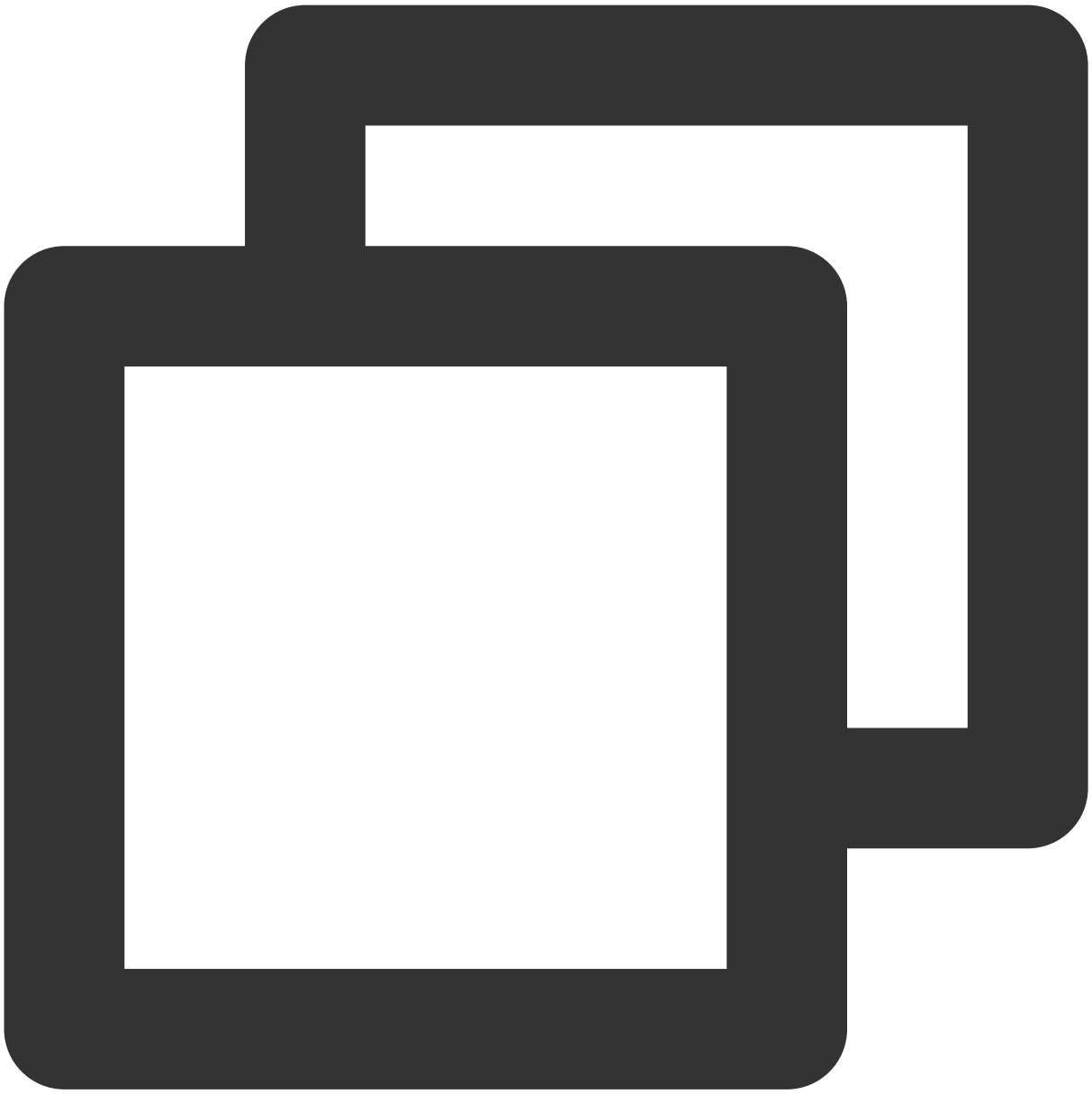


```
curl -LO https://storage.googleapis.com/kubernetes-release/release/v1.18.4/bin/wi
```

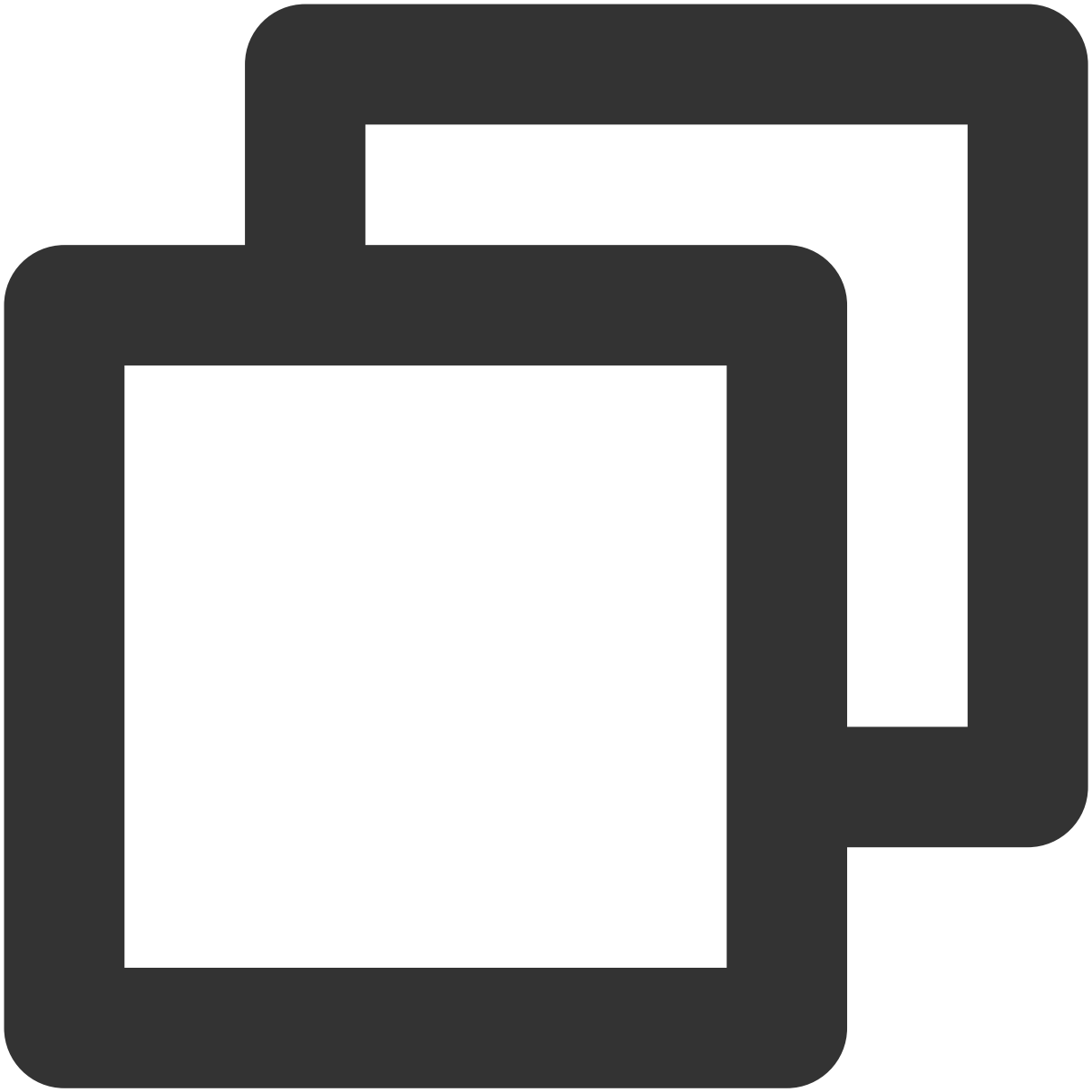
操作步骤

安装 Kubectl

1. 安装 kubectl 命令行工具。
2. 此步骤以 Linux 系统为例。执行以下命令，添加执行权限。

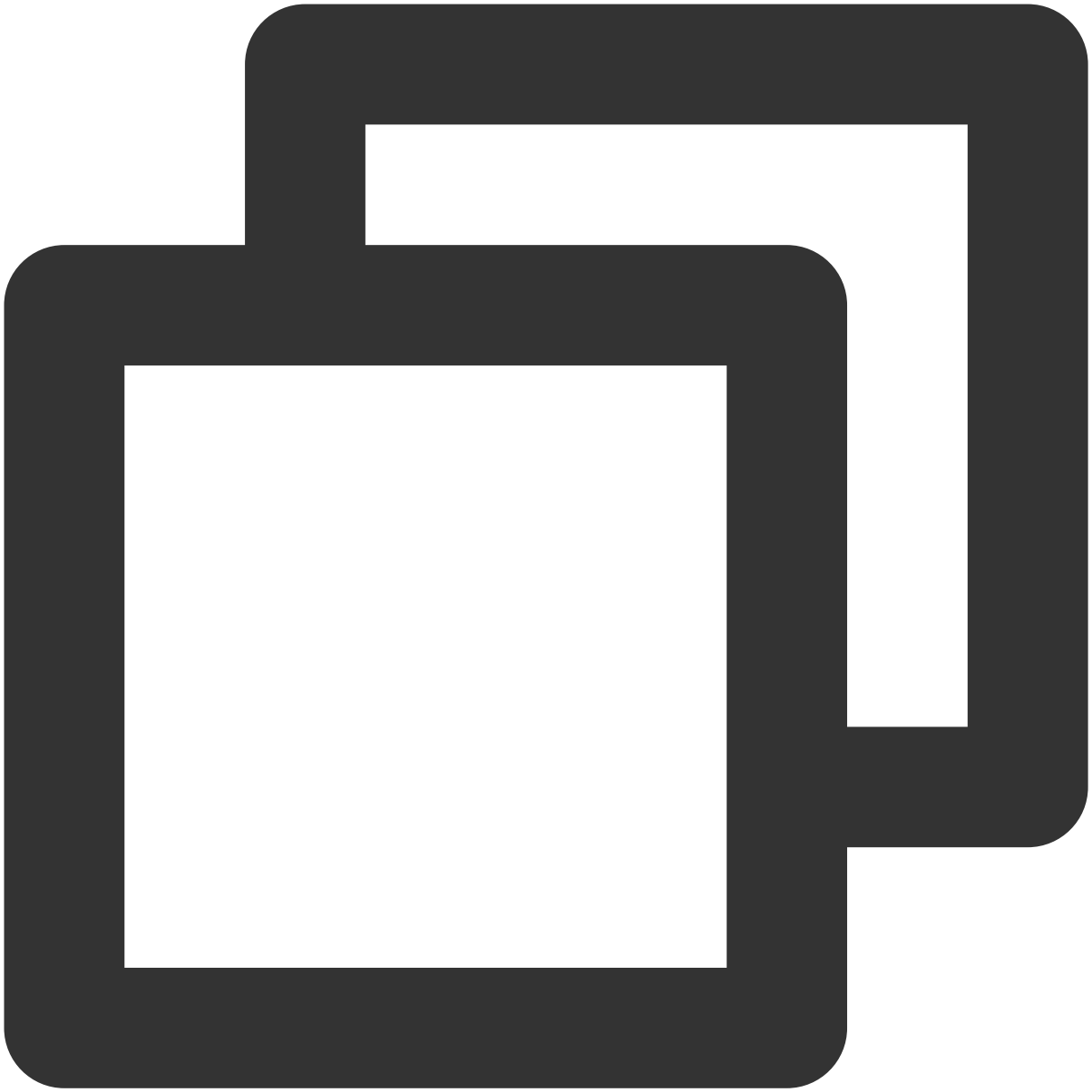


```
chmod +x ./kubectl
```



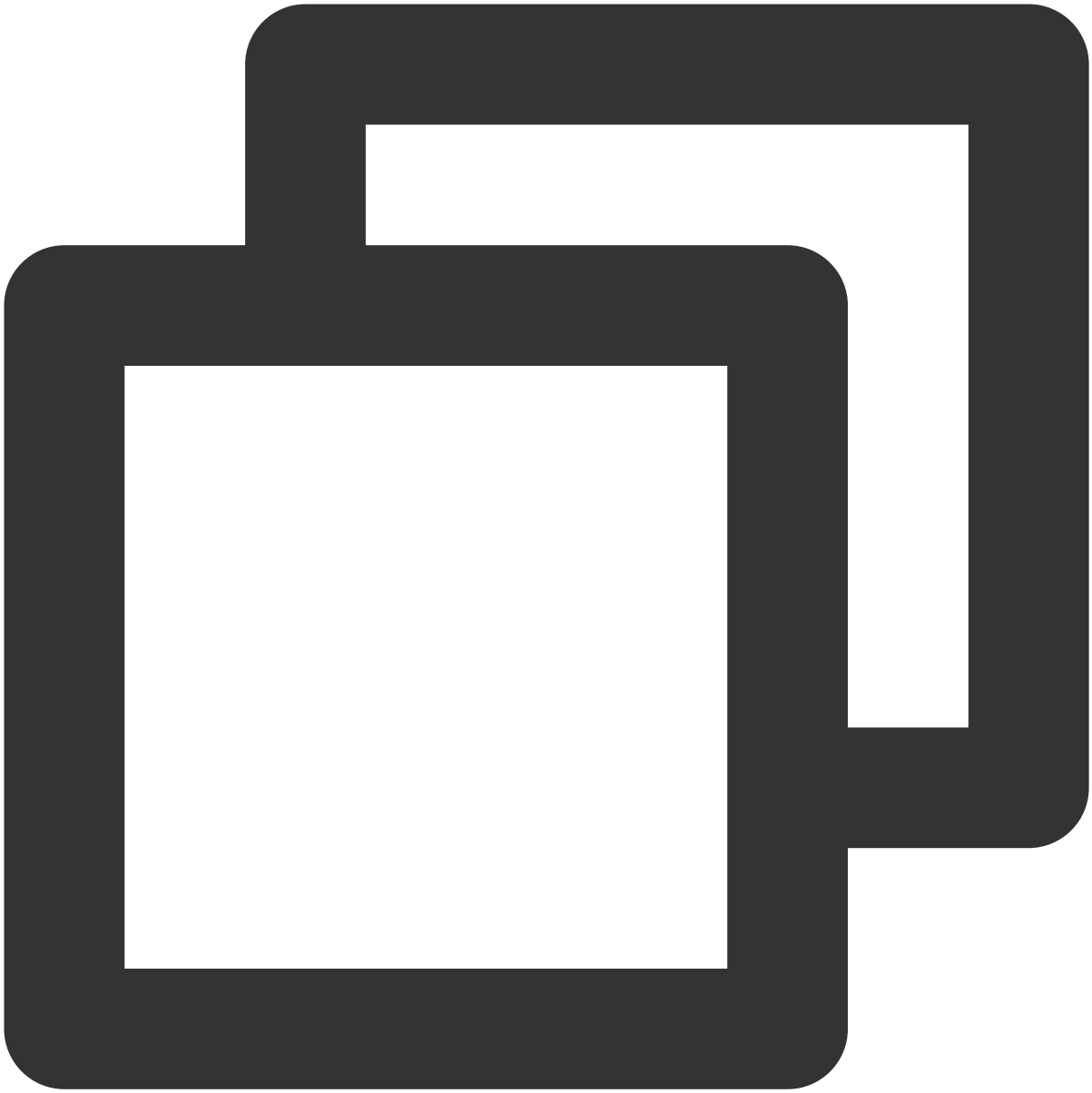
```
sudo mv ./kubectl /usr/local/bin/kubectl
```

3. 执行以下命令，测试安装结果。



```
kubectl version
```

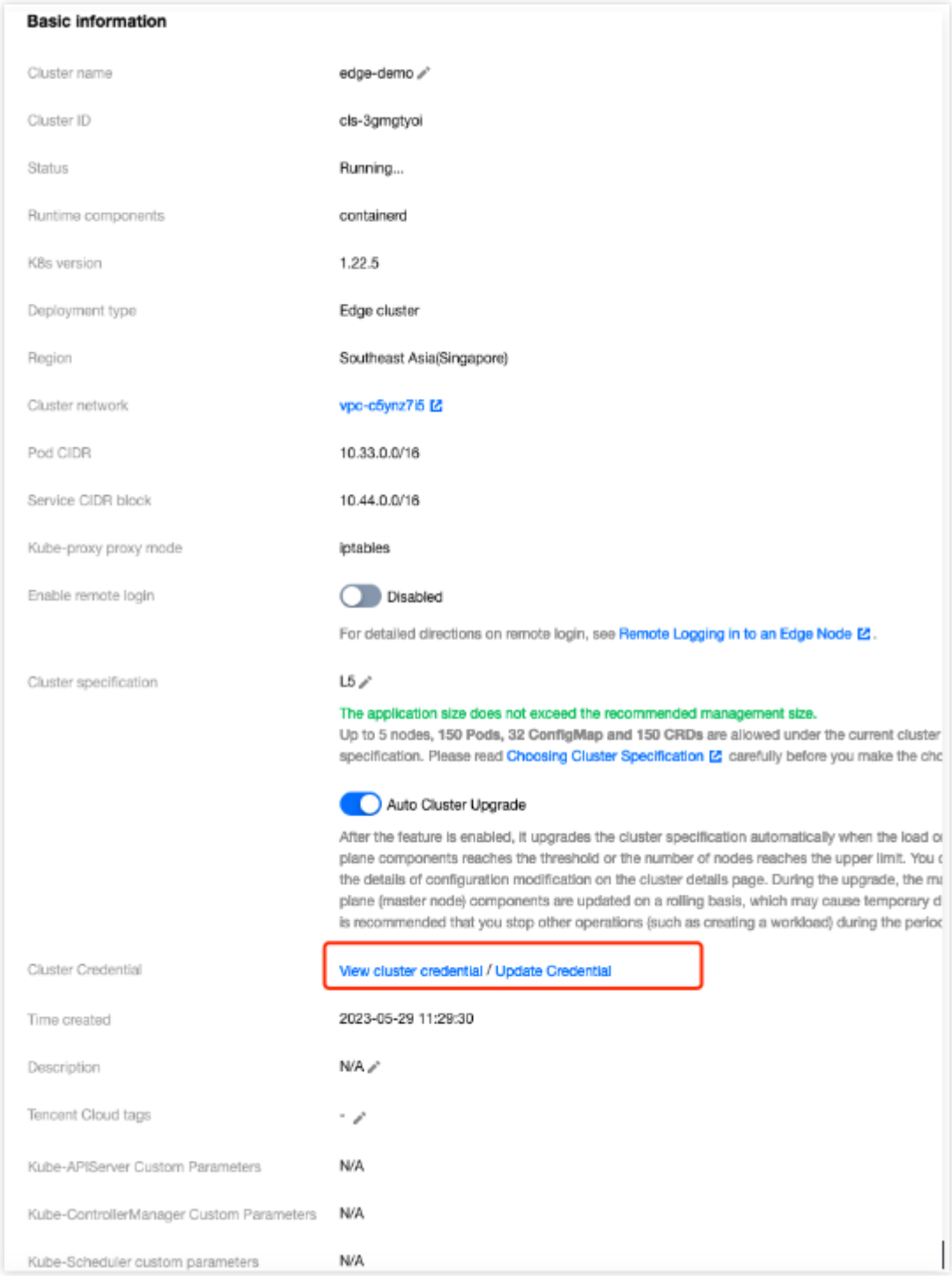
如若输出类似以下版本信息，即表示安装成功。



```
Client Version: version.Info{Major:"1", Minor:"5", GitVersion:"v1.5.2", GitCommit:"
```

获取集群证书信息

1. 登录容器服务控制台，选择左侧导航栏中的 集群。
2. 在“集群管理”页，单击需要连接的**集群 ID/名称**，即可在“基本信息”页面中**查看集群凭证**。如下图所示：



3. 在弹出的“集群凭证”窗口中，可以查看、复制及下载凭证内容。

说明

您可以根据实际需求，单击**复制**或**下载**将集群访问凭证保存到本地。

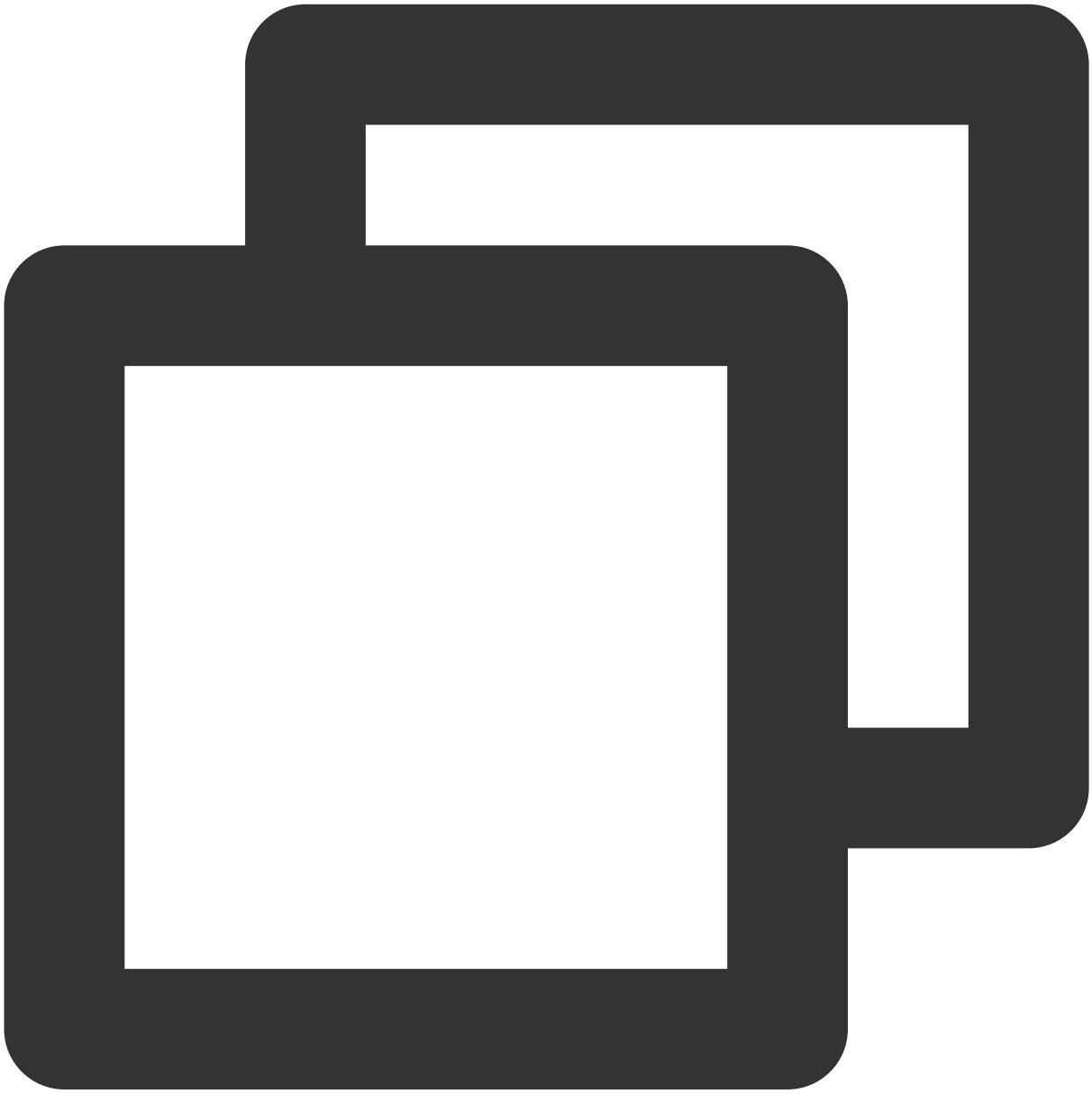
4. 在“外网访问”中，单击

将集群外网访问状态设置为开启，并可参考 设置 Kubectl 命令自动补全 直接使用集群访问凭证进行访问。

通过证书信息使用 Kubectl 操作集群

请求方法

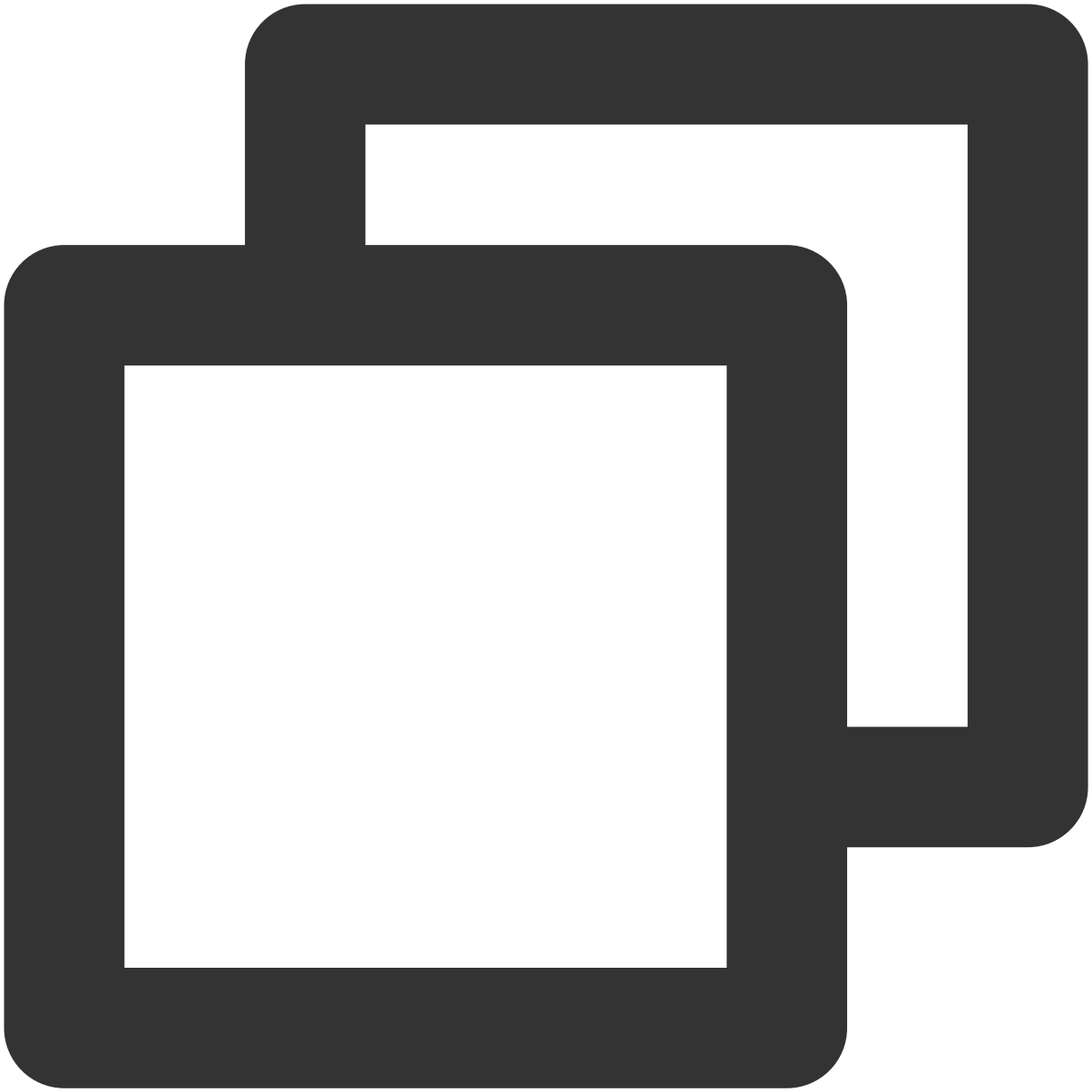
Kubectl 命令格式如下所示：



```
--kubeconfig=本地集群访问凭证
```

示例

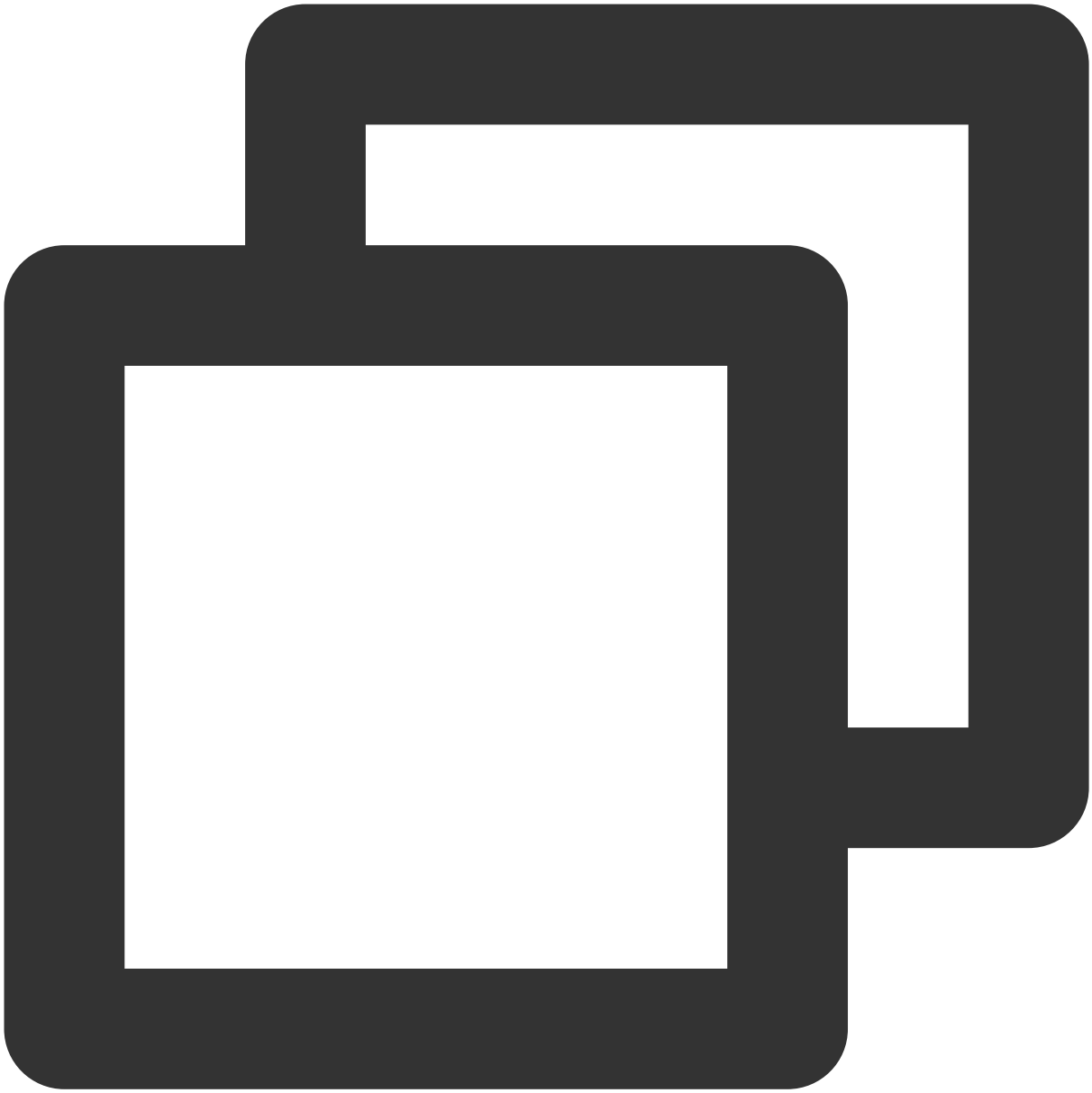
执行以下命令，查看集群下现存命名空间。



```
kubectl get namespace --kubeconfig=cls-8ipgf8u4.kubeconfig
```

本文使用集群凭证以 `cls-8ipgf8u4.kubeconfig` 为例，请替换为实际使用凭证。

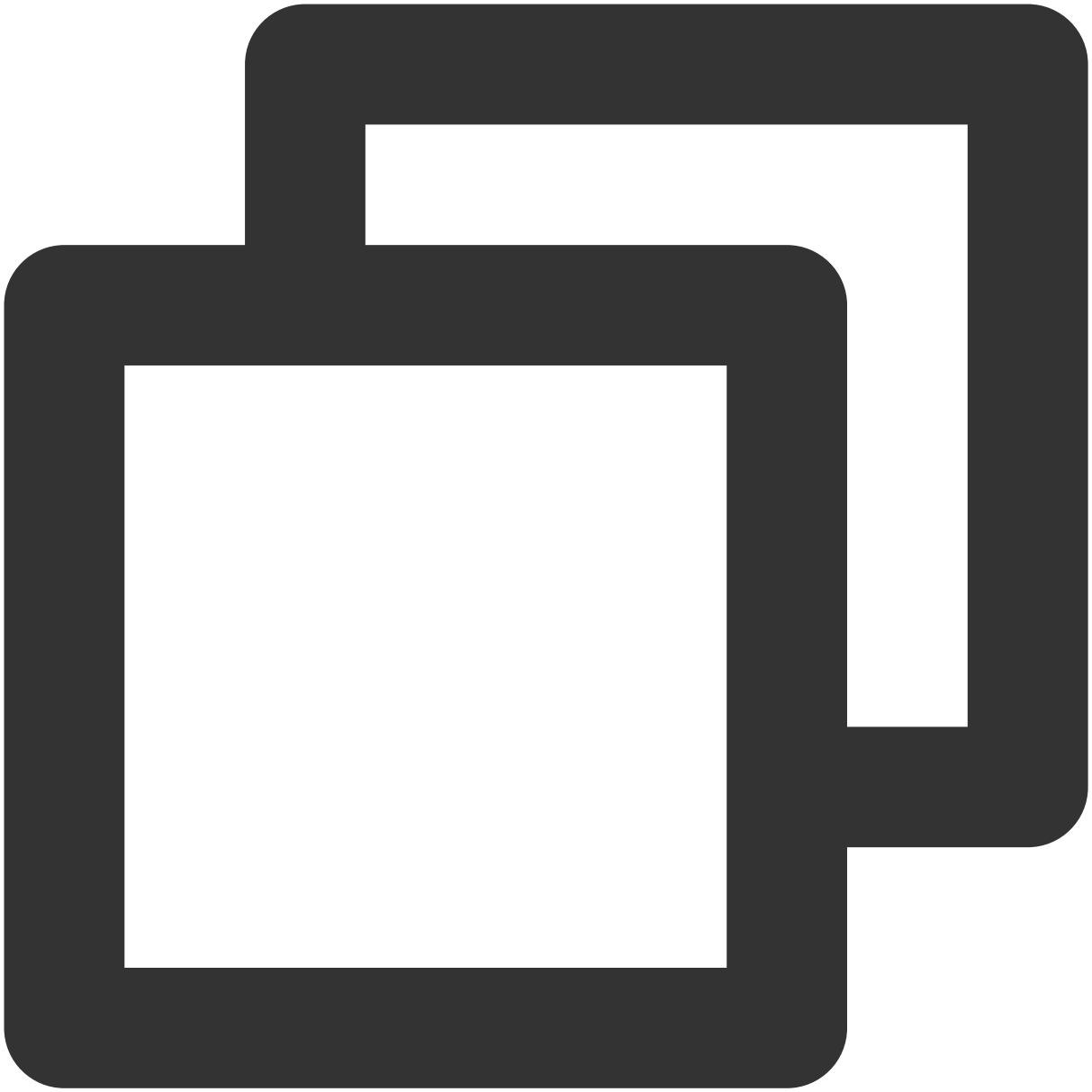
返回类似以下信息，则表示请求成功。



```
NAME          STATUS    AGE
default       Active    11d
kube-system   Active    11d
```

设置 Kubectl 命令自动补全

您可以通过执行以下命令，配置 Kubectl 自动补全，提高可使用性。



```
source <(kubectl completion bash)
```

边缘节点管理
添加节点
添加 ECM

最近更新时间：2023-06-01 11:22:54

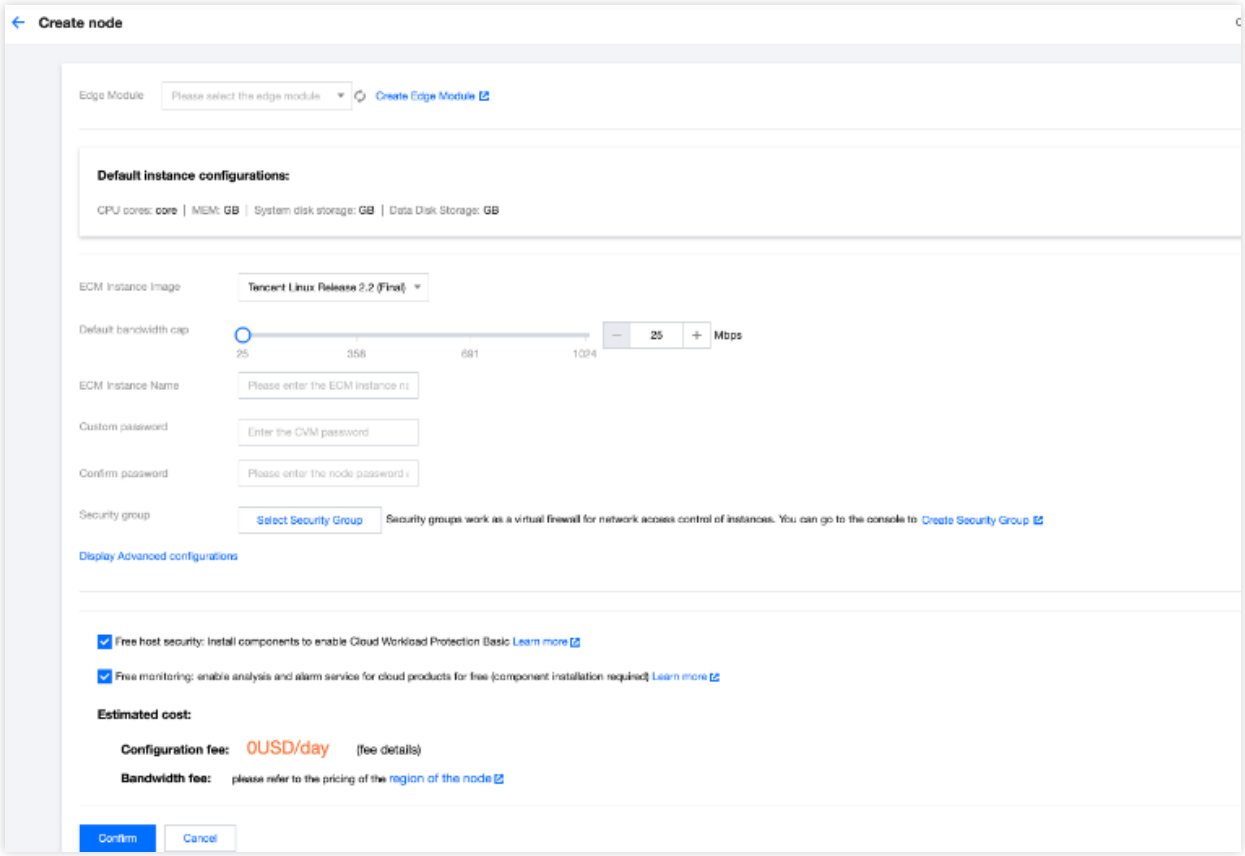
本文介绍如何向边缘容器集群添加 ECM 节点。ECM 是腾讯云边缘计算机器的简称，基于腾讯云分布在各地区的边缘节点提供计算和网络云服务。边缘节点分布范围广、距离用户更近，具有低网络时延、高可用、低成本的特点，可以和边缘容器服务无缝集成使用。

前置条件

请参考 集群开启内外网访问 使能**外网访问**能力。

创建 ECM 节点

1. 登录 容器服务控制台，选择左侧导航栏中的**集群**。
2. 在集群列表，单击边缘集群 ID，进入该集群详情页。
3. 选择页面左侧**节点管理** > **节点**，进入节点列表页面，单击**创建ECM节点**。
4. 在“新建节点”页面，根据实际需求配置相关参数，如下图所示：



主要参数信息如下：

边缘模块：选择边缘模块，确定 ECM 边缘节点规格。

ECM实例镜像：腾讯云提供公共镜像和自定义镜像。默认为与所属模块相同的镜像，请根据实际需求进行选择。

默认网络带宽上限：对带宽上限进行限制，若超出此上限，则默认丢包。默认为25Mbps，上限为1024Mbps。

ECM实例名称：表示需要创建的实例名称，用户自定义。

设置密码和确认密码：定义设置登录实例的密码。

安全组：安全组是一种虚拟防火墙，用于实例的网络访问控制。默认已选安全组为所属模块的安全组，用户可自行更改安全组设置。

高级设置：可修改默认设置，请根据实际需求进行选择：

Pod最大数量：决定分配给每个 Node 的 Pod 数量上限。

容器目录：设置容器和镜像存储目录，建议存储到数据盘。例如 `/var/lib/docker` 。

启动配置：shell 脚本格式。

节点省份：建议选择与您的客户最近的省份，可降低访问时延、提高访问速度。

节点地区：请根据实际需求进行选择。

网络类型：请根据实际需求选择公网运营商。

免费开通主机安全加固：默认勾选，帮助用户构建服务器安全防护体系，防止数据泄露。

免费开通云监控：默认勾选，免费开通云产品监控，安装组件获取主机监控指标并以监控图标形式展示，且支持设置自定义告警阈值等。

5. 单击**确定**后，等待相应的 ECM 实例创建成功并且状态变为 Ready 即可。

添加 CVM

最近更新时间：2023-06-01 11:22:54

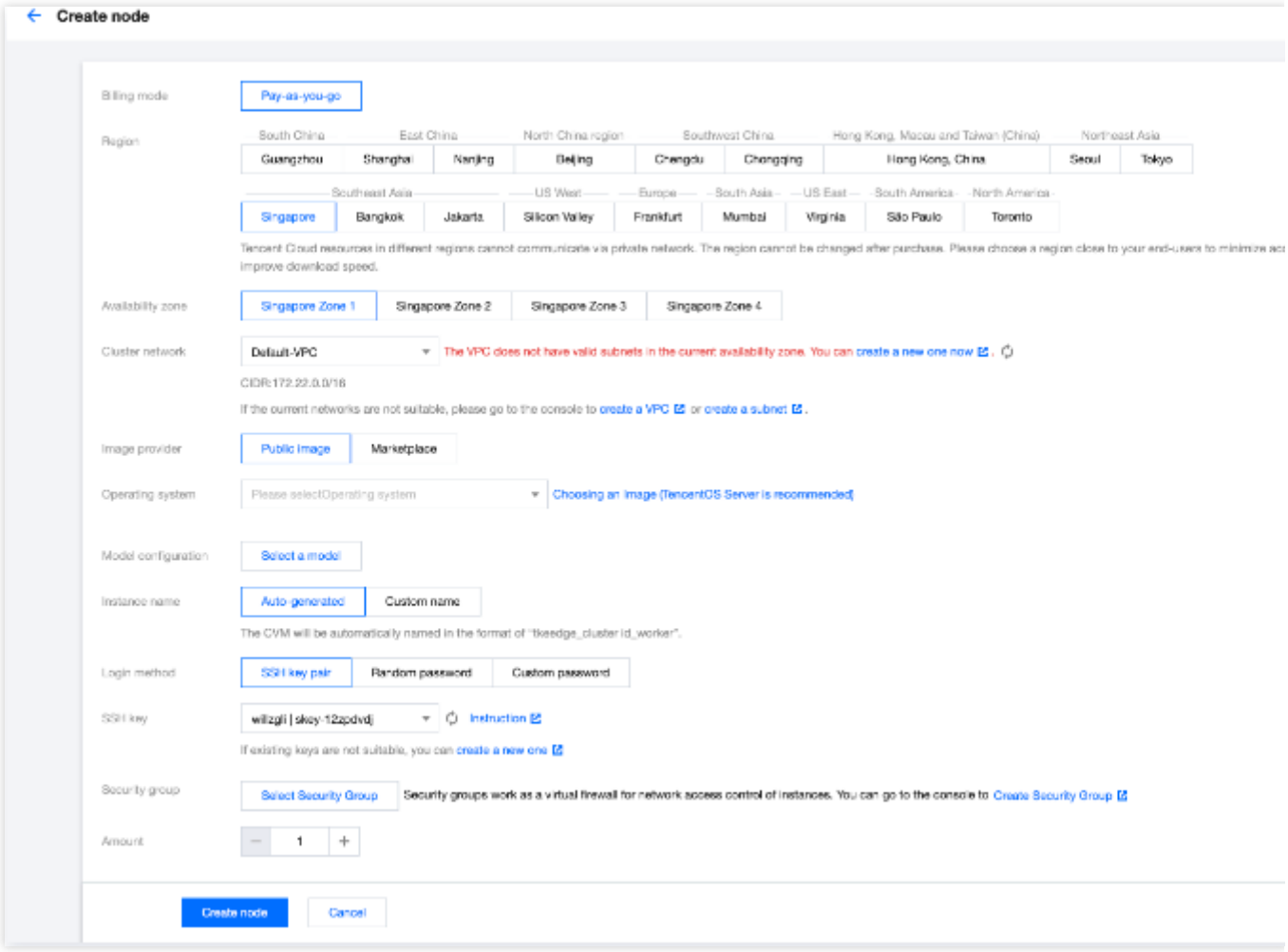
本文介绍如何向已创建的边缘集群中添加腾讯云标准的 CVM 节点。您可以直接购买所需地域的腾讯云 CVM，将其直接加入边缘集群提供服务。这里和标准 TKE 托管集群的区别是，您可以购买不同地域不同 VPC 下的 CVM 节点加入同一个边缘集群，而不需要限制在边缘集群创建时所选择的 VPC 下。

前置条件

请参考 集群开启内外网访问 使能**外网访问**能力。

创建 CVM 节点

1. 登录 容器服务控制台，选择左侧导航栏中的**集群**。
2. 在集群管理页，单击需要创建 CVM 节点的边缘集群 ID，进入该集群详情页。
3. 选择页面左侧**节点管理 > 节点**，进入节点列表页面，单击**创建cvm节点**。如下图



4. 在“新建节点”页面，根据实际需求配置相关参数。这里的参数和 TKE 添加节点类似，详情请参见 新增节点。

脚本添加第三方节点

最近更新时间：2023-06-01 11:22:54

本文描述如何使用 edgectl 脚本添加用户的第三方节点，和 CVM 以及 ECM 不同，此节点属于用户所有，可能位于用户 IDC 机房内、办公室或者各种类型边缘地域，这些机器可以通过互联网访问腾讯云服务。这种场景用户可以从云端下载 edgectl 脚本本地执行后，将节点加入到边缘集群中。

前置条件

请参考 集群开启内外网访问 使能**外网访问**或者**内网访问**能力。

注意

内网访问使能后，如您需要添加内网 CVM 机器，暂时只能使用脚本添加节点。

前提准备

按照以下条件准备好边缘节点：

节点来源：可使用 云服务器控制台 或 边缘计算机器控制台 中已有的服务器、其他平台或自建机房的服务器。

节点处理器：支持 x86_64、ARM64、ARM。

支持的操作系统如下（经过测试可以适配）：

Ubuntu 20.04/18.04/16.04

Centos 8.0/7.8/7.6/7.2

TencentOS Server 3.2/3.1/2.6/2.4

Tencent Linux Release 2.2 (Final)

Debian 10.2/9.0

SUSE Linux Enterprise Server 12 SP3

请确保需添加节点已安装 `wget` 、 `systemctl` 及 `iptables` 。

节点网络需具备主动访问公网能力。

说明

由于不同用户部署的操作系统环境各不一致，不排除某些配置影响无法成功添加节点，这时请详细查看节点上 edgectl 的部署日志排查问题：**/tmp/tkeedge-install.log**

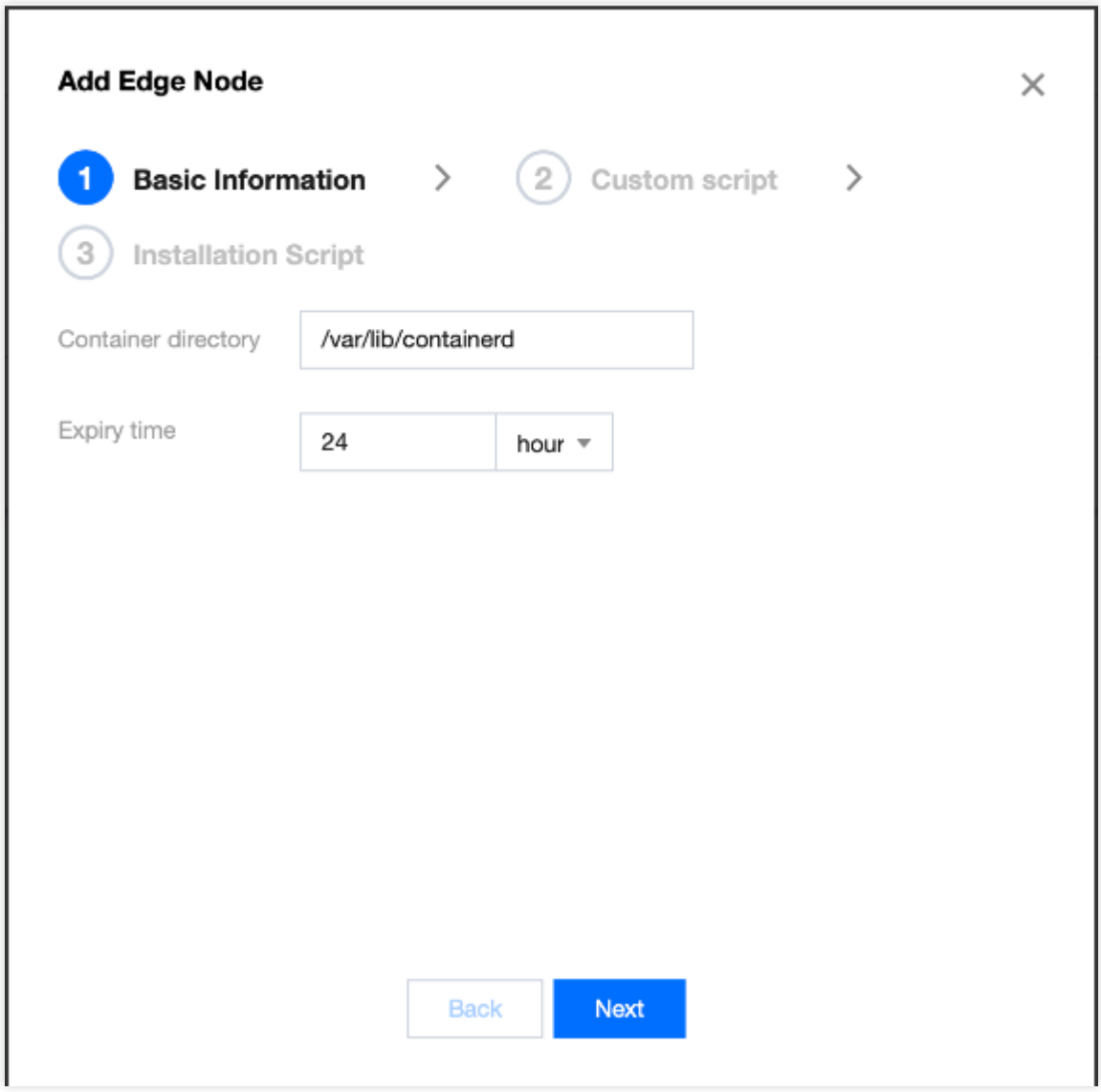
操作步骤

1. 登录 容器服务控制台，选择左侧导航栏中的**集群**。
2. 在集群管理页，单击需要添加边缘节点的边缘集群 ID，进入该集群详情页。

3. 选择页面左侧**节点管理** > **节点**，进入节点列表页面，单击**脚本添加节点**。

4. 在弹出的“添加边缘节点”窗口中，按照以下步骤进行配置，得到节点侧初始化脚本。

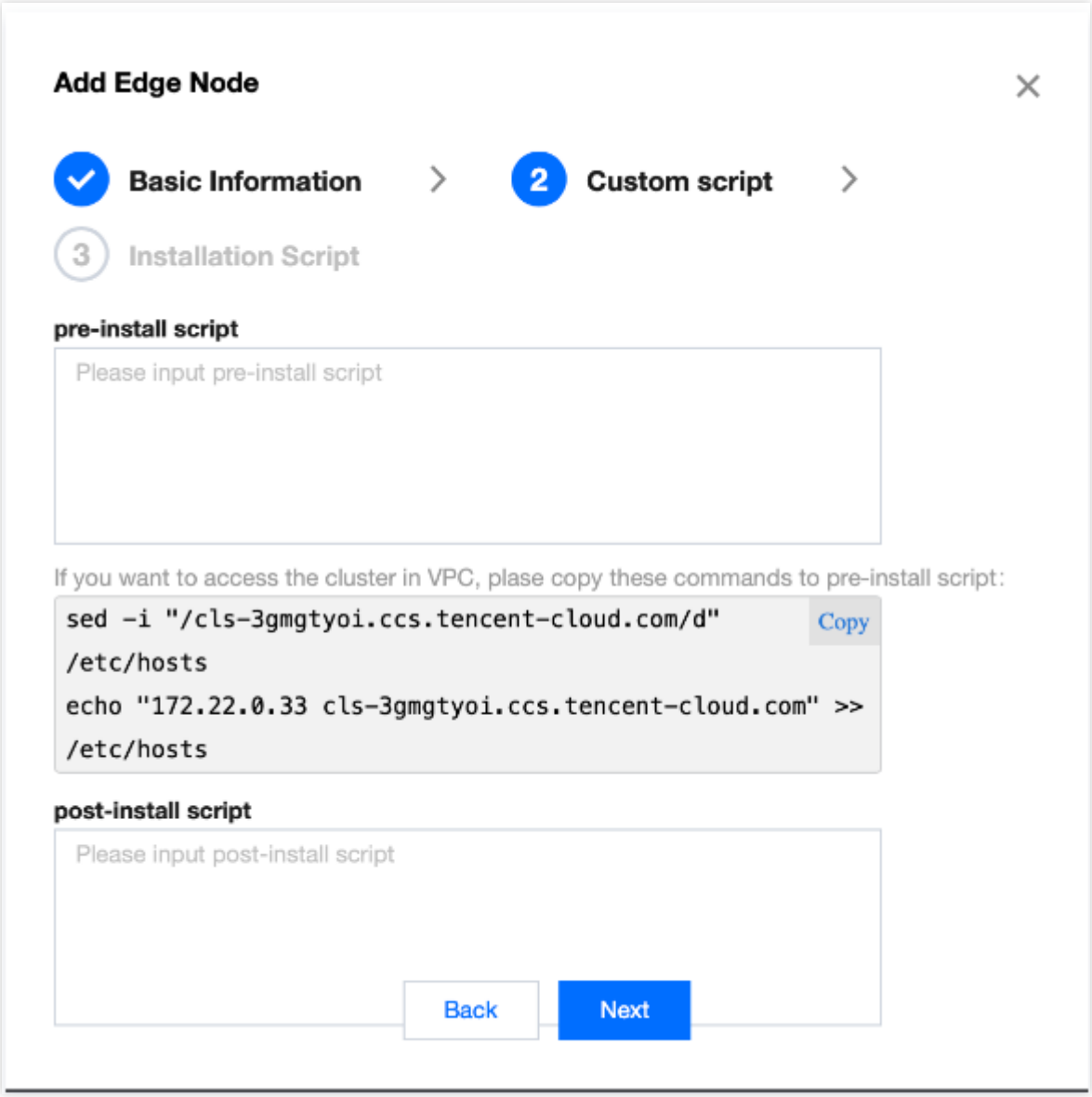
4.1 在“配置”步骤中，获取初始配置，并可修改以下参数值：



容器目录：设置容器和镜像存储目录，建议存储到数据盘。例如 /var/lib/docker。

过期时间：设置"安装脚本"命令的过期时间，可配置范围为 1分钟 ~ 24小时，默认配置为 24 小时。

4.2 单击**下一步**，添加自定义脚本：



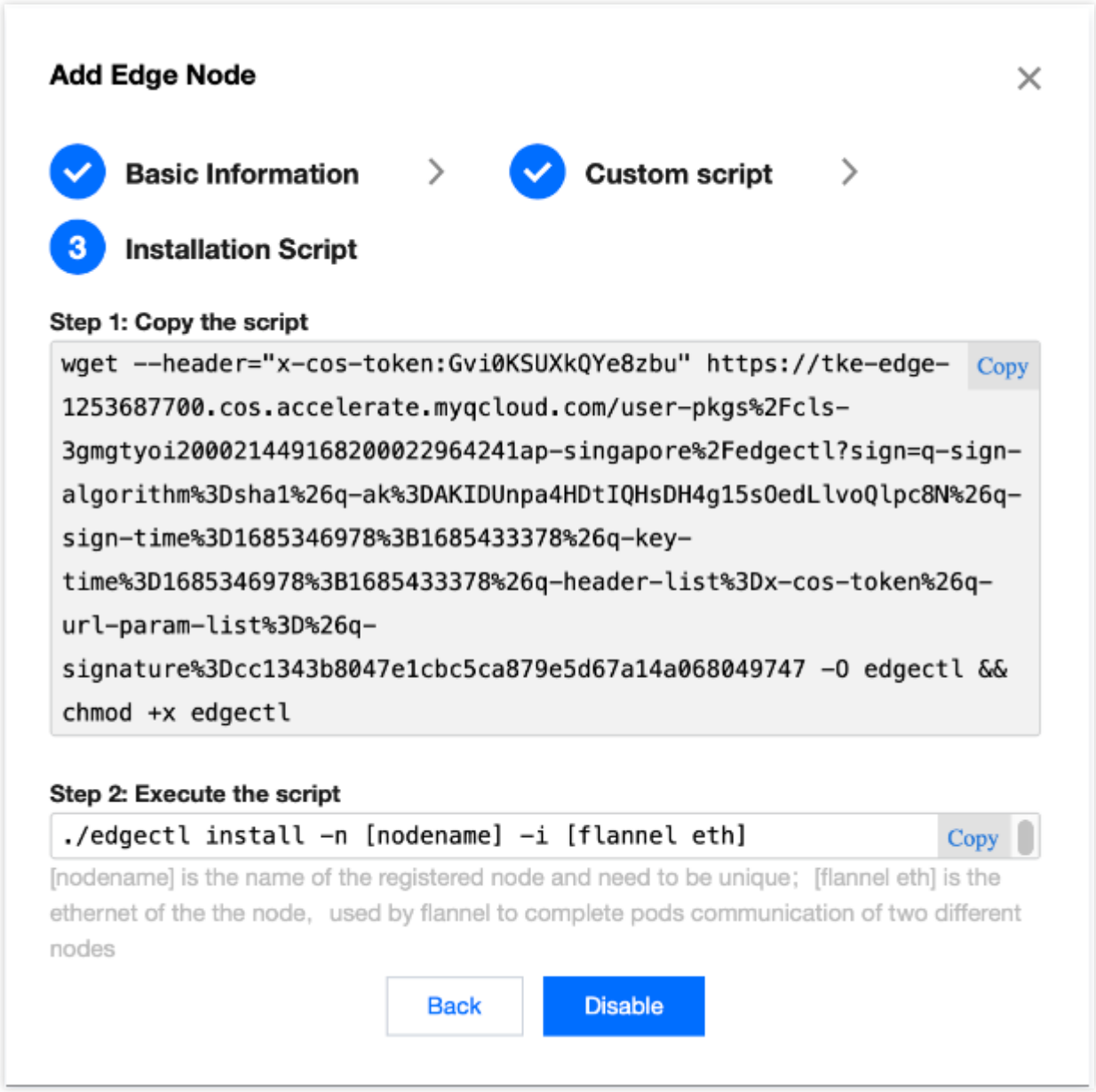
pre-install脚本：这里可以设置节点接入的前置脚本，用于在特殊条件下执行做一些清理或者预配置相关工作，增强不同场景适配性。

post-install脚本：这里可以设置节点接入的后置脚本，用于在节点成功接入后，执行一些后续清理或者配置相关工作，增强不同场景适配性。

注意

在上图中，因为使能了“内网访问”，会提示需要执行命令，如果用户想要通过内网添加 CVM 节点，请复制黏贴这些指令到 pre-install 脚本；如果通过外网添加节点，则忽略这些指令。

4.3 在“安装脚本”步骤中，复制脚本命令用来获取到对应的节点初始化脚本。



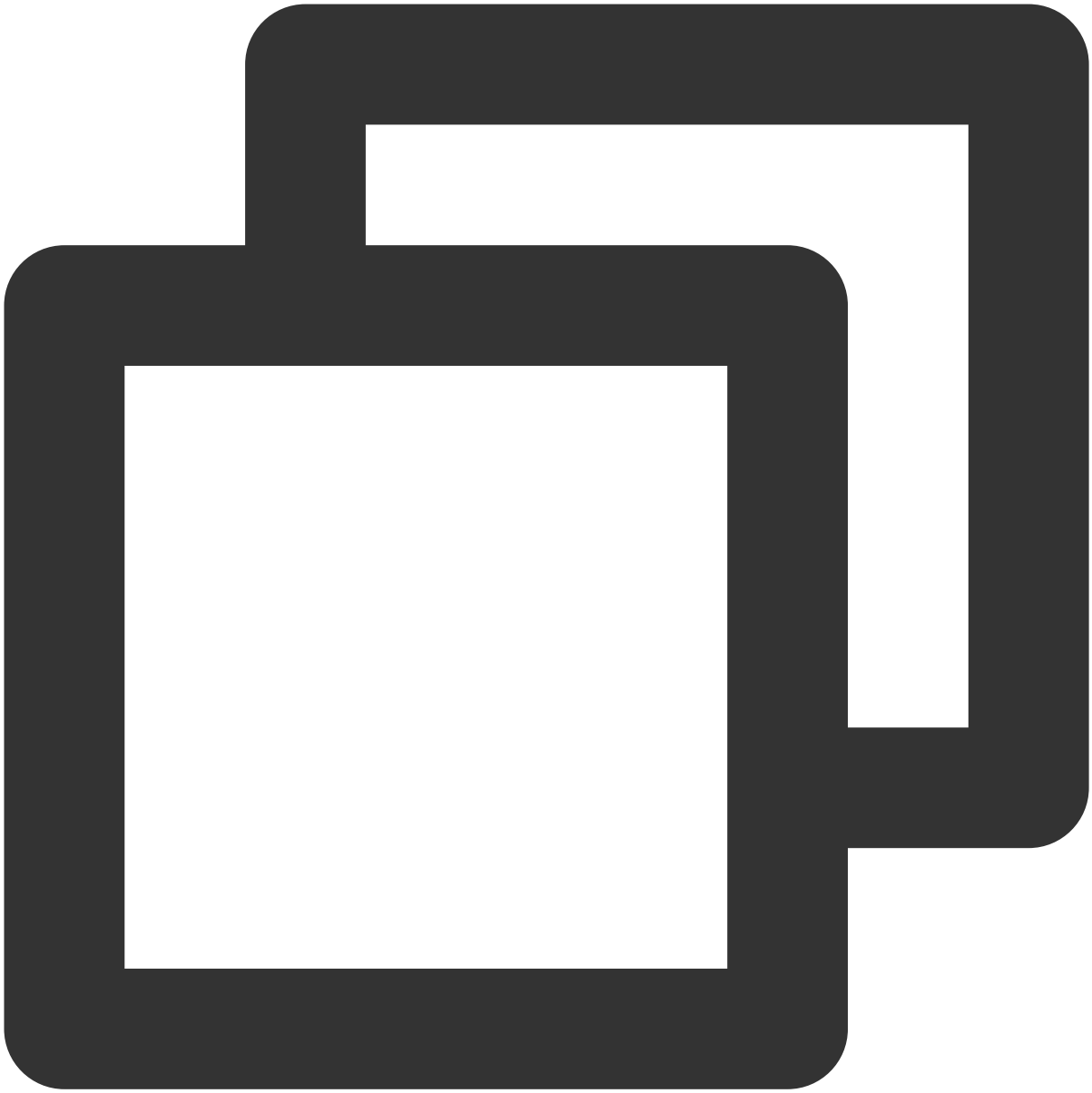
5. 登录已准备好的服务器，并切换至 `root` 帐户执行已复制的命令。

说明：

执行该命令会自动下载 edgectl 边缘节点管理工具。

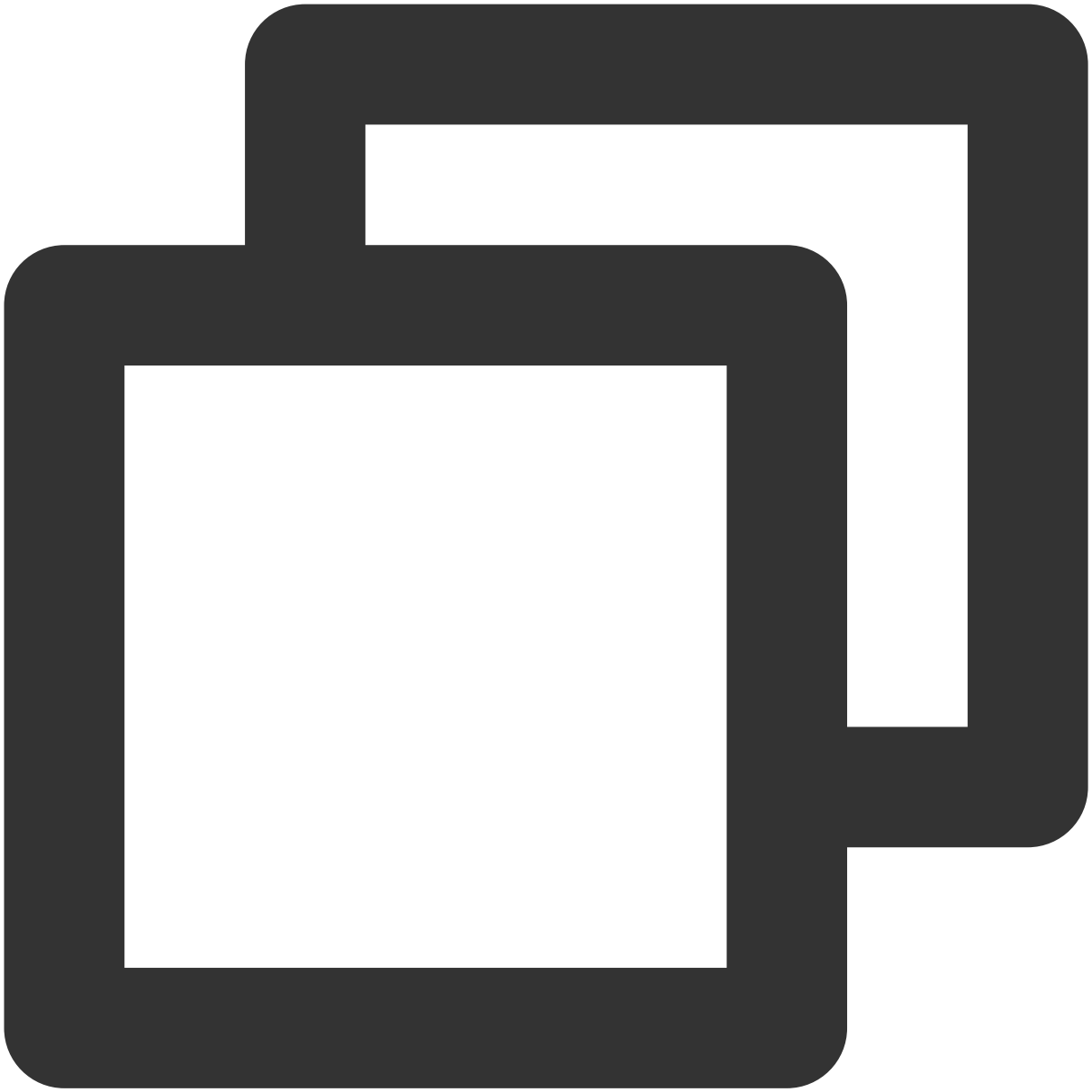
此步骤获取的 edgectl 有效期为 “过期时间” 所设置的时间，请在有效期内完成添加节点操作。过期后请重新执行脚本添加节点操作，重新获取 edgectl。

6. 查看 edgectl 命令，运行初始化节点操作。更多 edgectl 操作请参见 edgectl 边缘节点管理工具说明。



```
./edgectl -h

Usage:
   edgectl command [flags]

Available Commands:
   check        Check the edge node if to be add to clusters
   install      Install components to edge node
   clear        Clear edge node and recovery as usual

Flags:
```

```
    -h, --help   Help for edgectl
```



```
./edgectl install -n Your-Node-Name -i Your-Interface-Name
```

说明

执行该命令会检查主机环境并安装边缘节点所需的组件，自动将节点注册至边缘集群。

受网络和安装速度影响，添加节点操作可能等待数分钟。

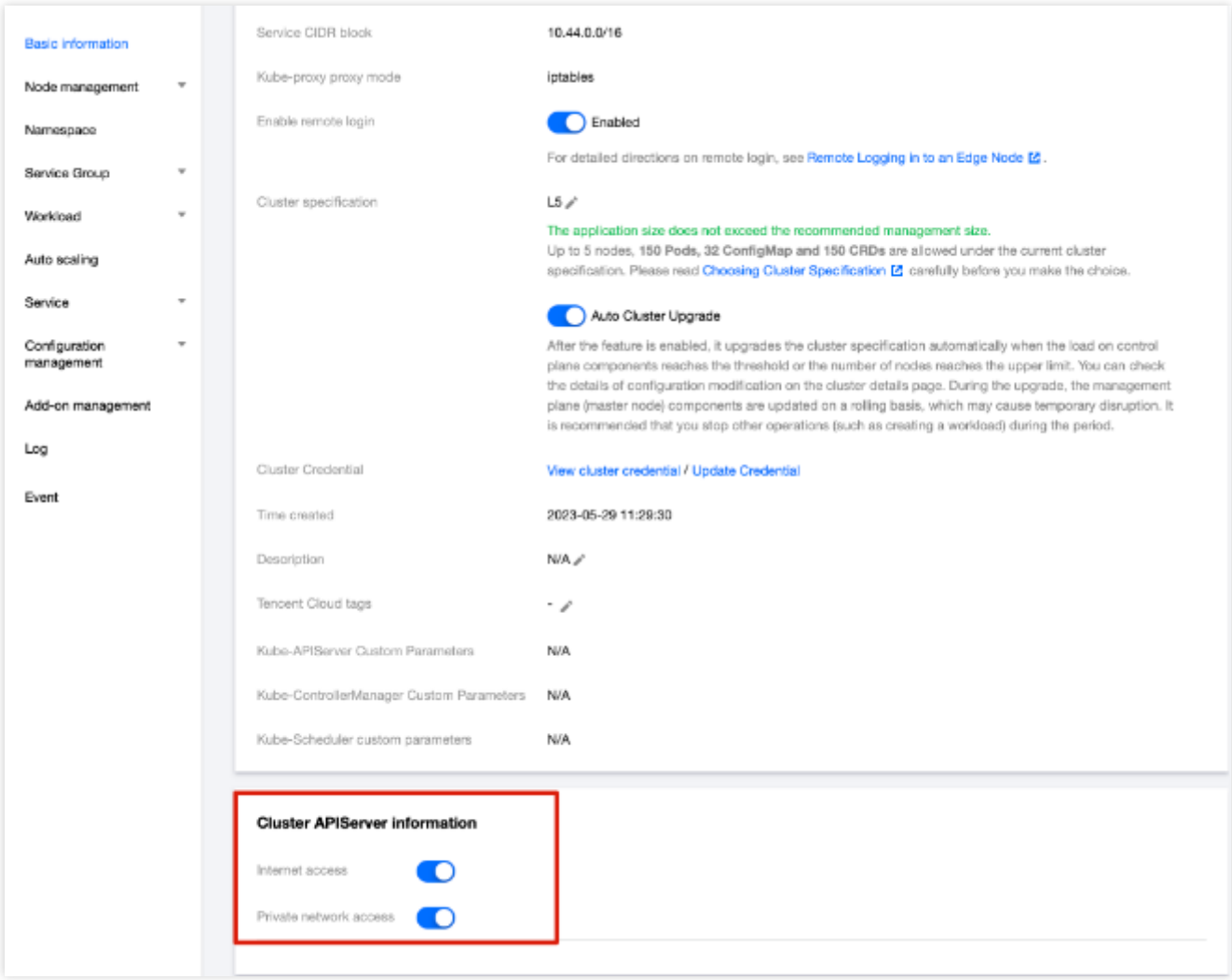
查看日志文件确认安装过程信息：`/tmp/tke-edge-install.log`。

7. 确认命令执行成功后，前往“节点列表”页面并刷新，即可查看已新增的节点。您还可进行节点的其他操作，例如驱逐、移出、封锁、编辑标签或取消封锁等。

相关操作

关闭集群内外网访问

在集群的“基本信息”页面，关闭“外网访问”和“内网访问”即可。

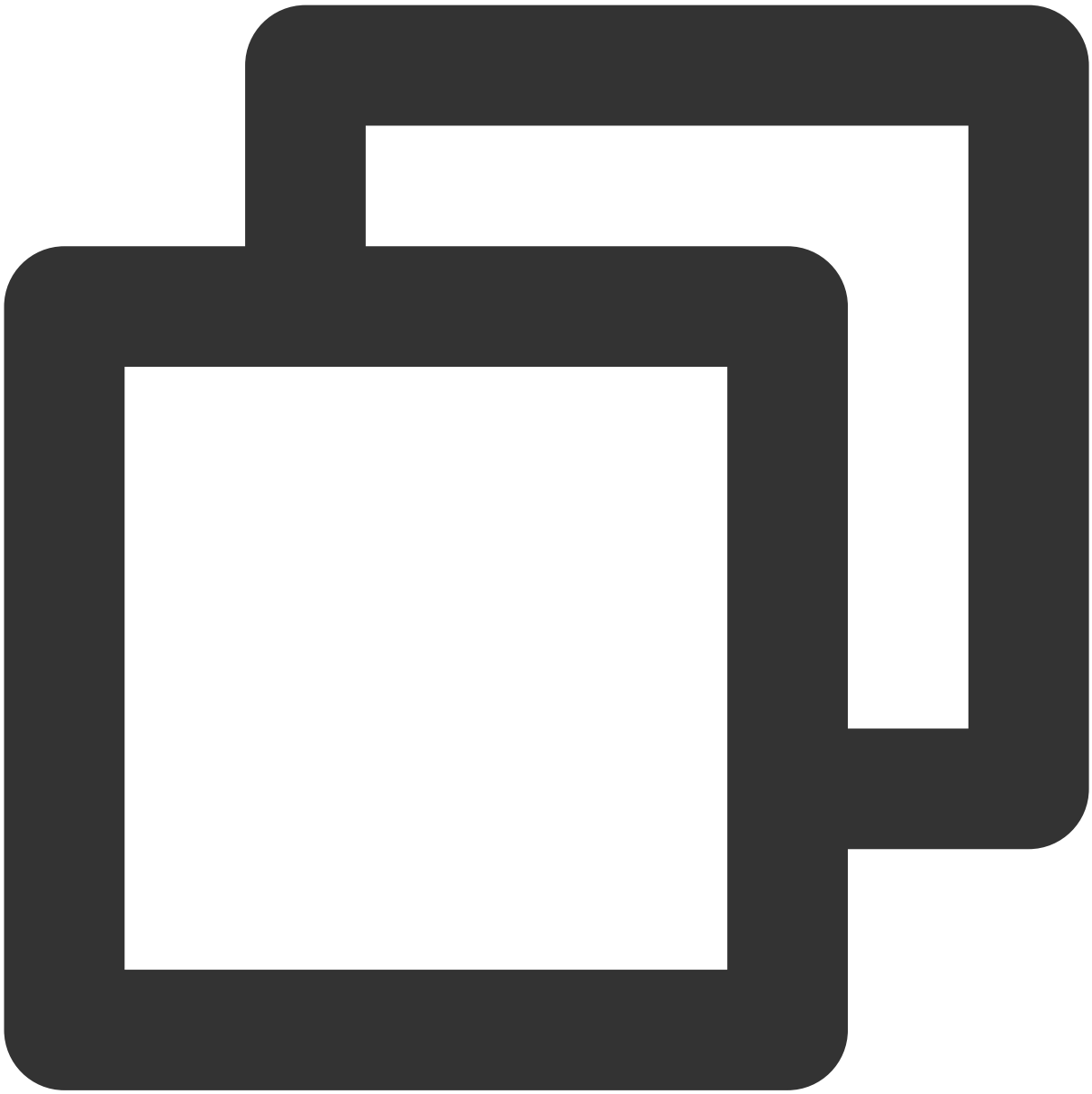


边缘节点权限问题

1. 通过边缘集群 K8S 所管理的边缘节点，当前节点上权限默认设置为 system:node，关于 system:node 权限的说明请参考 Using Node Authorization。
2. 在边缘节点上的 kubeconfig 文件将默认使用 system:node 权限，如需获取集群凭证用以操作集群，请参考 连接集群。

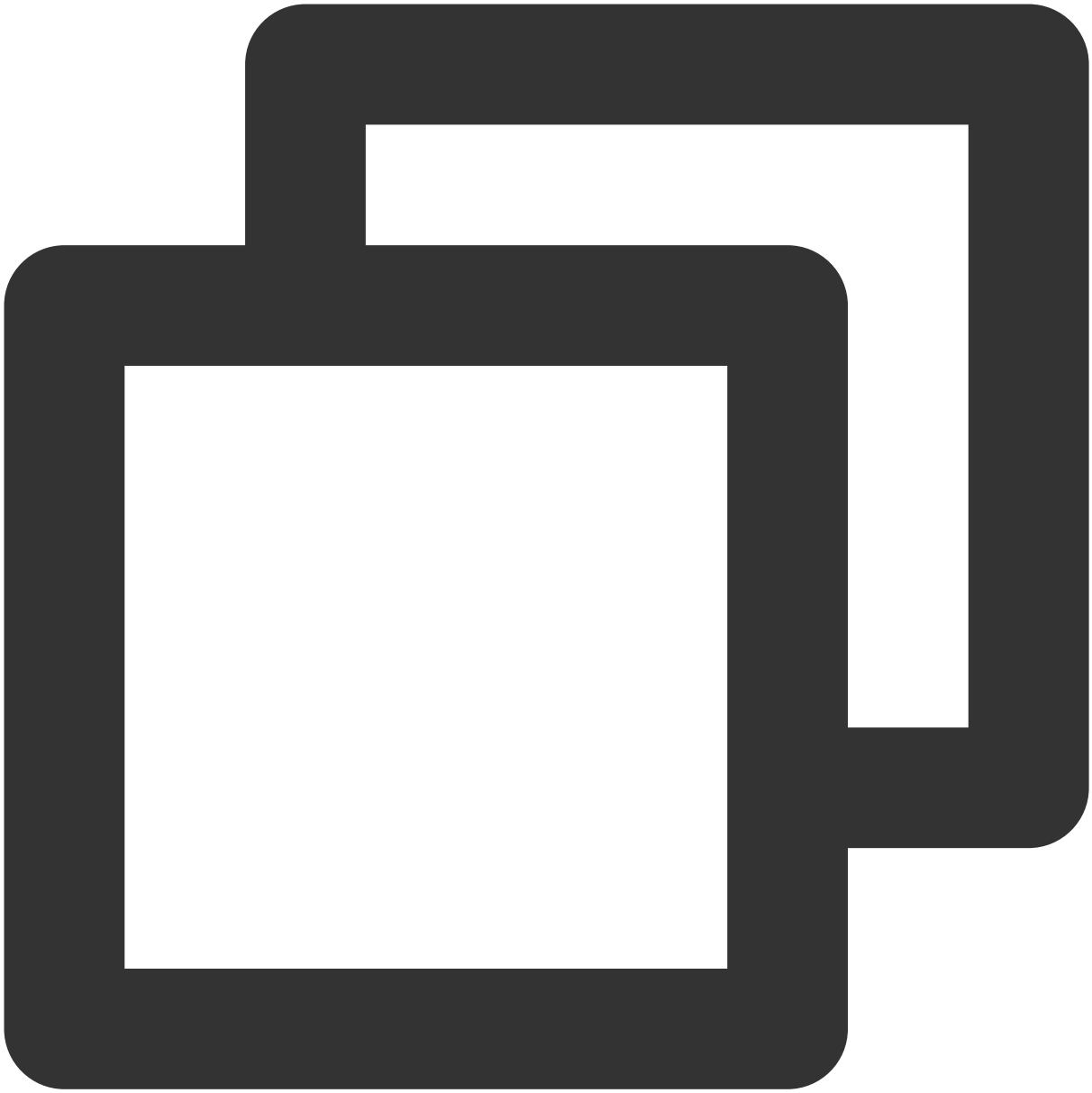
边缘节点带 GPU 相关

边缘集群中已经提交了一个 **nvidia-device-plugin** 的 DaemonSet，镜像版本 `nvidia-device-plugin:v0.9.0` ，若边缘节点需要部署 DaemonSet，执行以下命令便可自动部署到相应节点：



```
kubectl label nodes ${NODE_NAME} nvidia-device-enable=enable
```

取消可执行如下命令：



```
kubectl label nodes ${NODE_NAME} nvidia-device-enable-
```

其中 `${NODE_NAME}` 为节点名称。

腾讯云可支持带 GPU 的边缘节点操作系统

支持的 NVIDIA 系列 GPU 版本：

GN6

GN7

GN8

GN10X、GN10Xp

GI3X

支持的 Linux 系统：

CentOS 7.2 64位

CentOS 7.6 64位

CentOS 7.7 64位

CentOS 7.8 64位

CentOS 7.9 64位

Ubuntu Server 18.04.1 LTS 64位

TencentOS Server 2.4

Tencent Linux Release 2.4

以上相应系统带 NVIDIA 系列 GPU 版本边缘集群会默认安装 GPU 驱动，其他系统用户可自行安装相应 GPU 驱动。

脚本添加带 GPU 的 CVM 节点

用脚本添加的边缘节点边缘集群不负责安装 GPU 驱动，GPU 驱动用户可自行安装。

注意

添加带 GPU 的 CVM 系统版本是 边缘节点系统 的子集，若边缘节点的系统不支持，添加节点可能会失败，请遵守 边缘节点的系统 要求。

节点管理

最近更新时间：2023-06-01 11:22:54

本文介绍如何管理集群中已经添加的节点。

查看节点信息

1. 登录 容器服务控制台，选择左侧导航栏中的**集群**。
2. 在集群管理页，单击边缘集群 ID，进入该集群详情页。
3. 选择页面左侧**节点管理** > **节点**，进入节点列表页面。
4. 单击需要查看的节点名称，进入节点详情页面，在详情页面可以显示节点的具体信息：

pod 管理：可以查看该节点下 Pod 列表以及 pod 运行状态。
事件：展示该节点在 Kubernetes 集群下的 Event 信息。
详情：展示该节点在 Kubernetes 集群下的相关主机信息和 Kuberentes 信息。
YAML：展示该节点的 Node 资源的 Yaml 信息。

驱逐节点

如果需要对某个节点进行封锁，后续不允许应用调度到该节点，为节点添加 `Label：SchedulingDisabled` 后，可以驱逐该节点。

1. 登录 容器服务控制台，选择左侧导航栏中的**集群**。
2. 在集群管理页，单击边缘集群 ID，进入该集群详情页。
3. 选择页面左侧**节点管理** > **节点**，进入节点列表页面。
4. 选择需要驱逐的节点，选择**操作** > **驱逐**，节点即会进入“已封锁”状态，后续应用不会调度到该节点。
5. 如果需要解除封锁，选择**操作** > **取消封锁**，节点即可进入正常状态。

删除节点

如果集群内某个节点不再需要，可以从集群中删除节点。

1. 登录 容器服务控制台，选择左侧导航栏中的**集群**。
2. 在集群管理页，单击边缘集群 ID，进入该集群详情页。
3. 选择页面左侧**节点管理** > **节点**，进入节点列表页面。
4. 选择需要删除的节点，选择**操作** > **移出**，节点即会从 Kubernetes 集群中删除 Node 资源。

注意

脚本添加的节点：此移出操作只会从 Kubernetes 集群中删除 Node，等同于 Kubectl delete node，并不会清理节点数据，需要用户自行清理。
通过 ECM/CVM 添加的节点：此移出操作不仅会从 Kubernetes 集群中删除 Node，同时会释放申请的 ECM/CVM资源，用户无需再手动释放该资源。

节点远程登录

最近更新时间：2023-06-01 11:22:54

操作场景

本文介绍如何在云端通过 SSH 远程登录边缘节点，边缘节点可以是公有云上的节点或者本地的内网节点。

开启节点远程登录

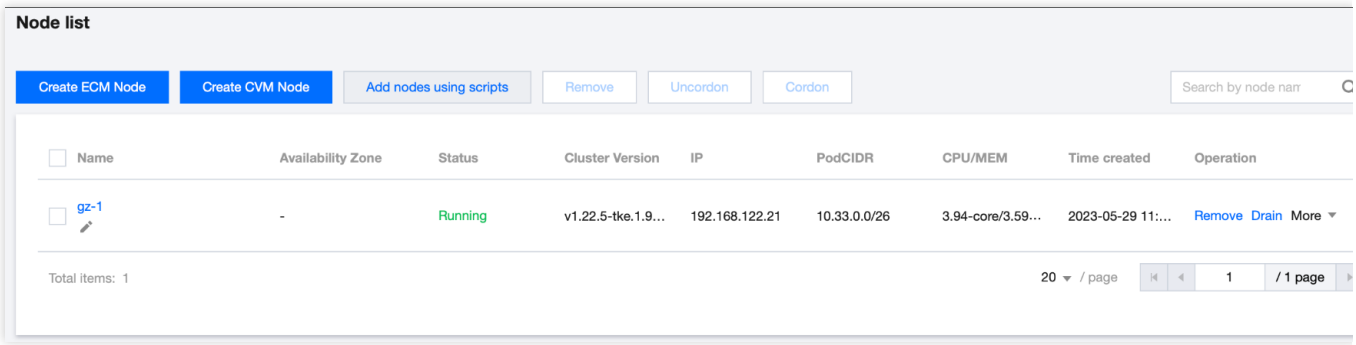
关闭节点远程登录

远程登录边缘节点

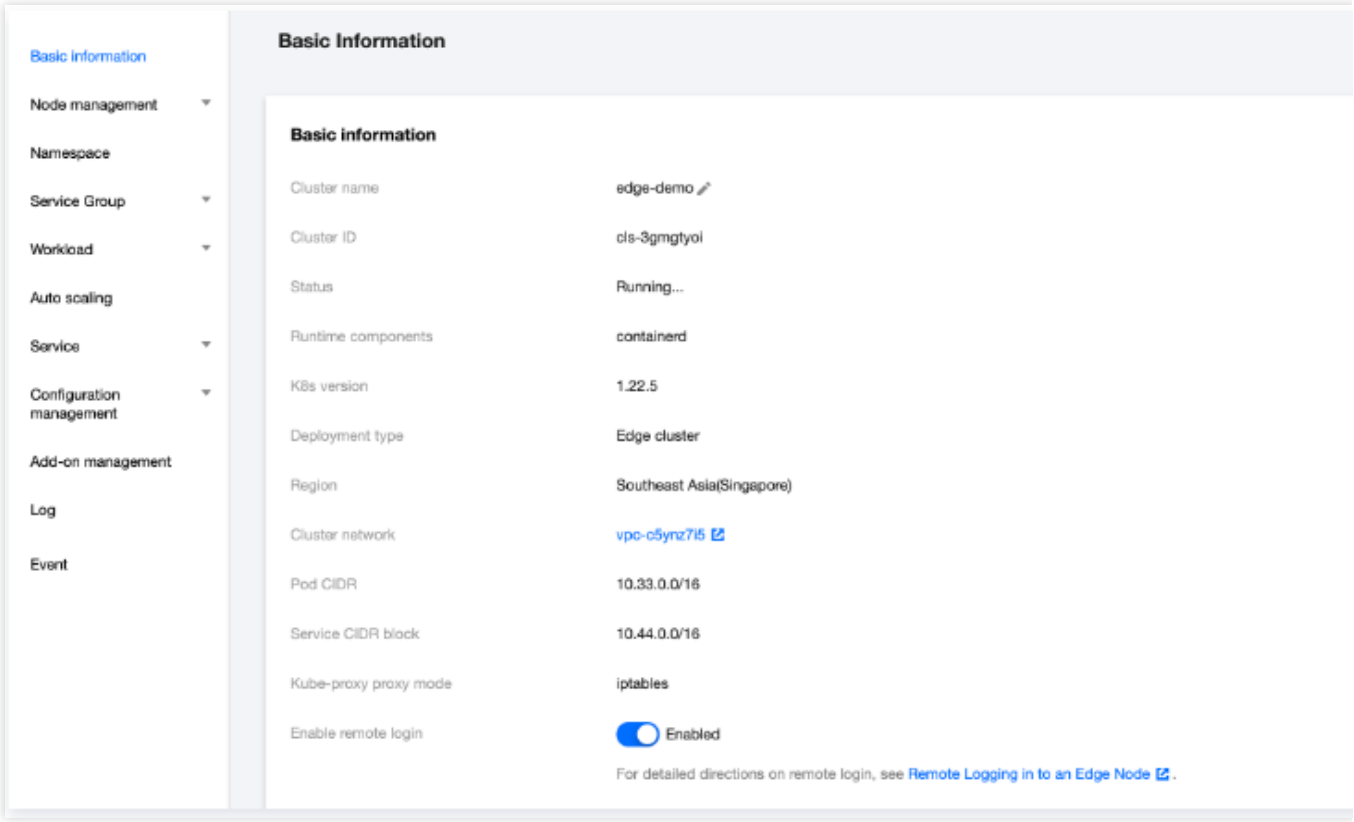
操作步骤

开启节点远程登录

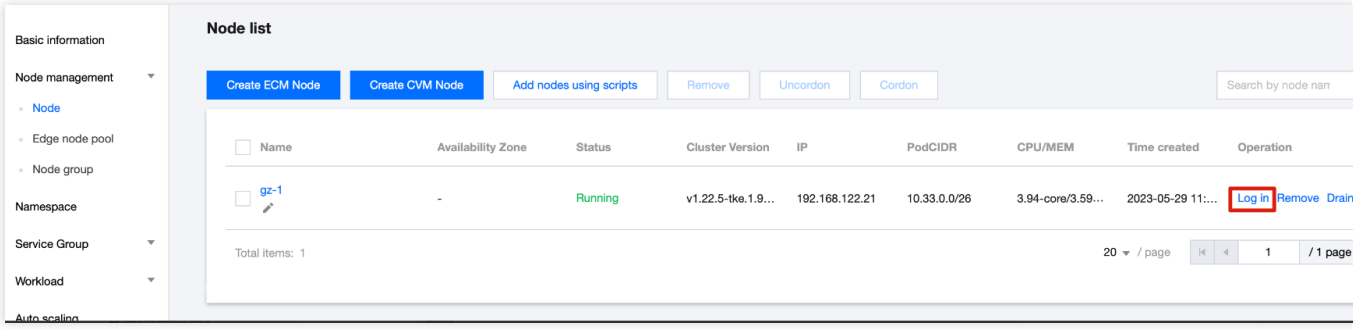
1. 登录 腾讯云容器服务控制台。
2. 在集群管理页面，单击需要远程登录节点的集群 ID，进入该集群详情页。
3. 选择页面左侧**节点管理** > **节点**，进入节点列表页面，确认**操作**列是否显示**登录**超链接，默认均不显示。如下图所示：



4. 选择页面左侧**基本信息**，进入集群基本信息页面，单击**开启远程登录**开关，开启远程登录功能。如下图所示：

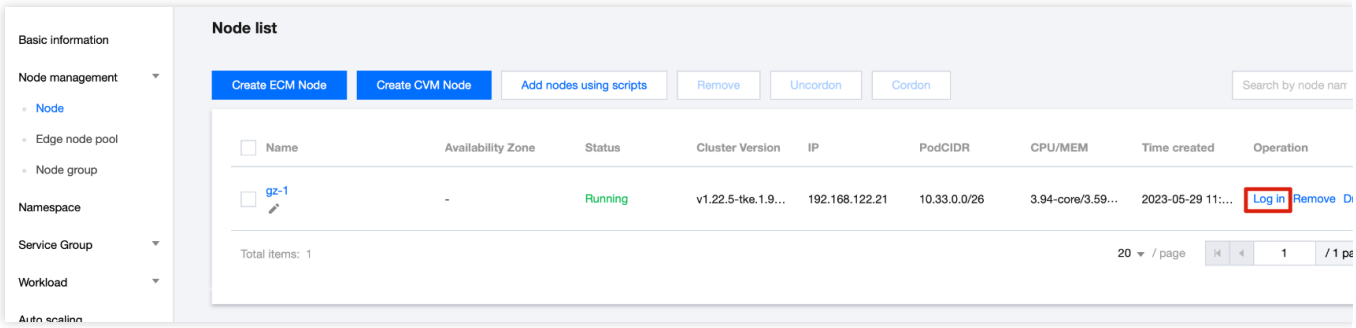


5. 返回页面左侧**节点管理** > **节点**，重新进入节点列表页面，确认**登录**超链接已经显示。如下图所示：

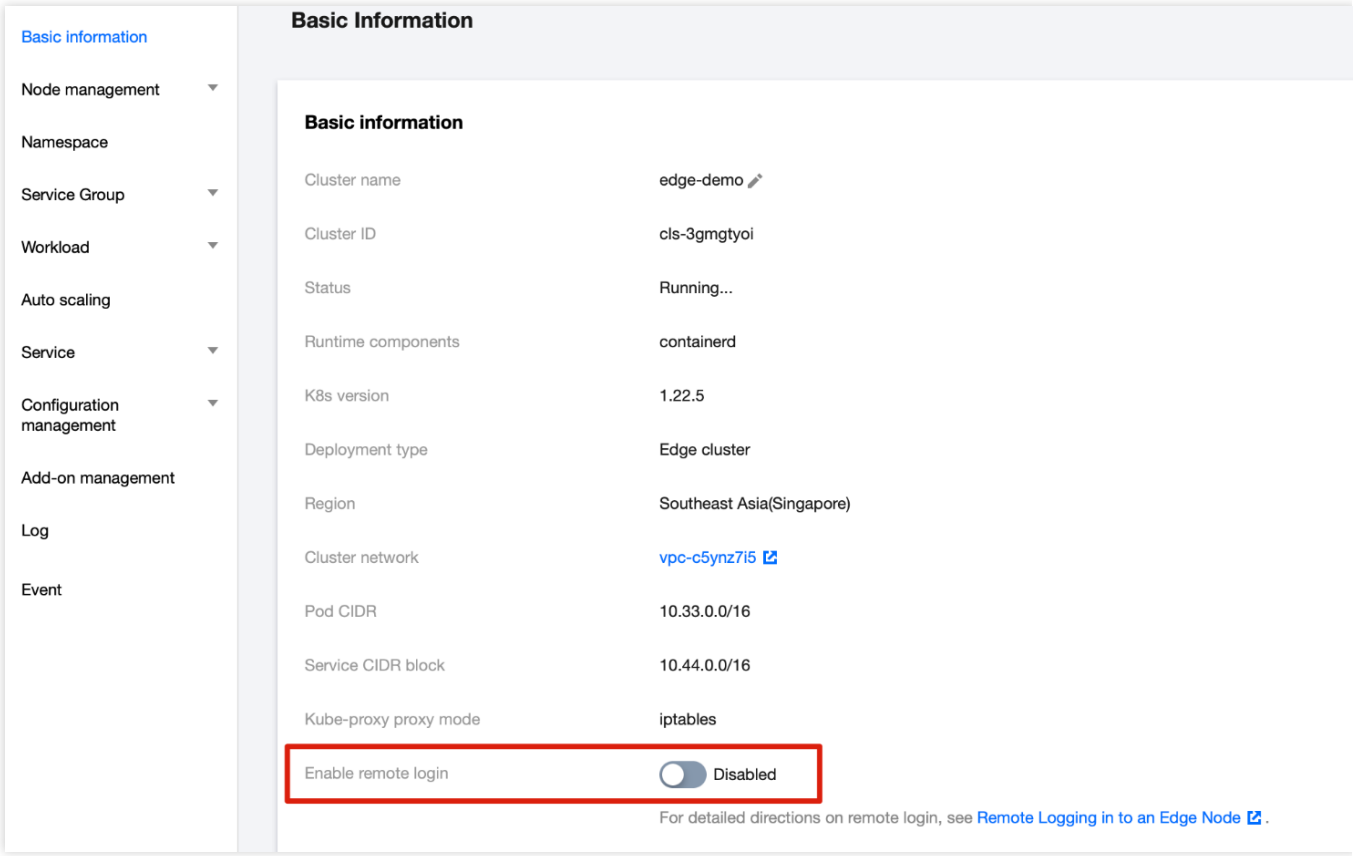


关闭节点远程登录

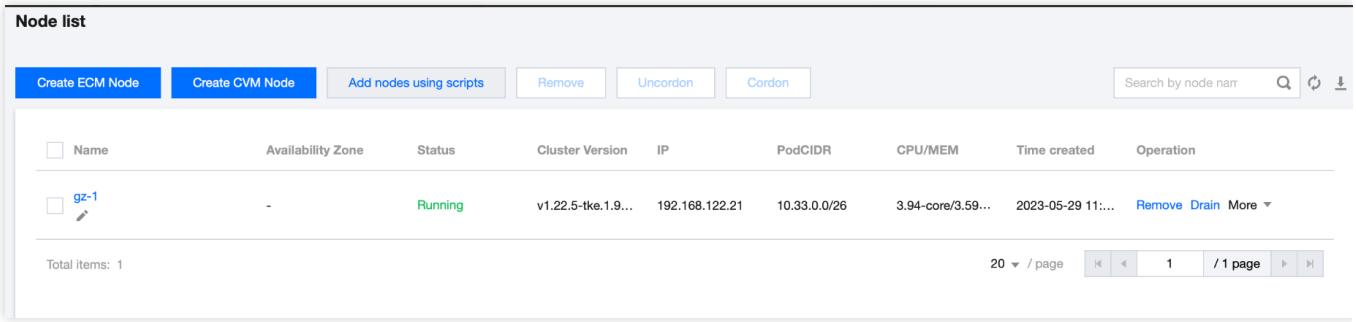
1. 登录 腾讯云容器服务控制台。
2. 在集群管理页面，单击需要远程登录节点的集群 ID，进入该集群详情页。
3. 选择页面左侧**节点管理** > **节点**，进入节点列表页面，确认**操作**列中的**登录**超链接已显示。如下图所示：



4. 选择页面左侧**基本信息**，进入集群基本信息页面，单击**开启远程登录**开关，关闭远程登录功能。如下图所示：

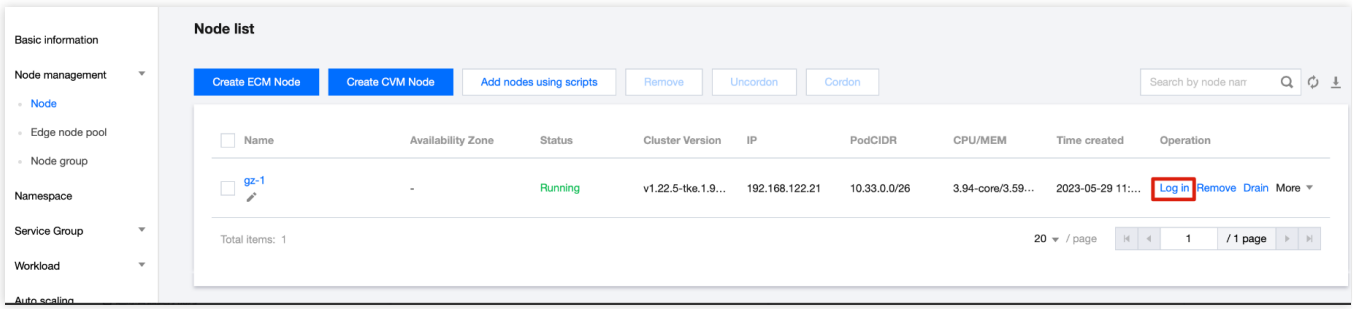


5. 返回页面左侧**节点管理** > **节点**，重新进入节点列表页面，确认**登录**超链接已隐藏。如下图所示：

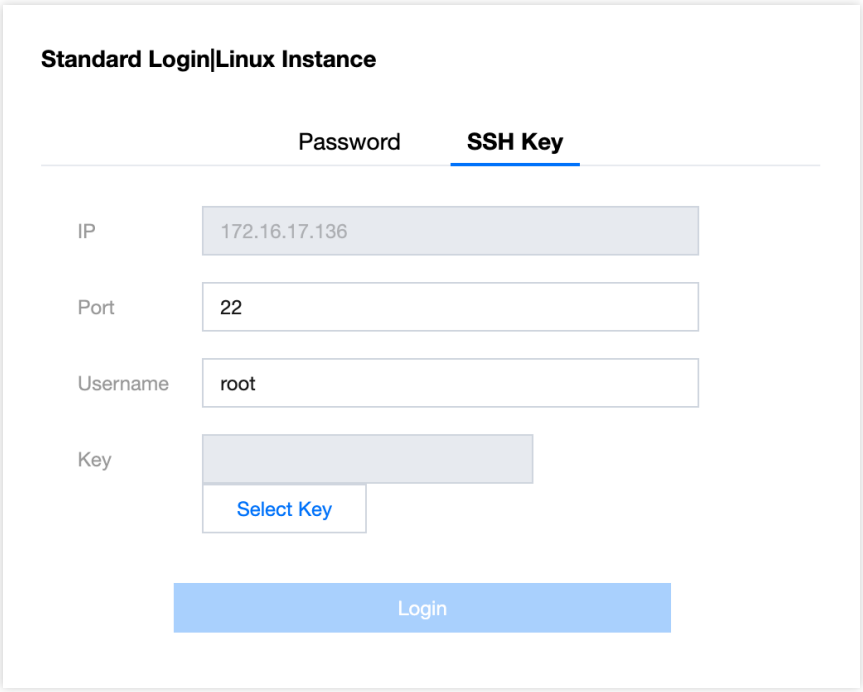


远程登录边缘节点

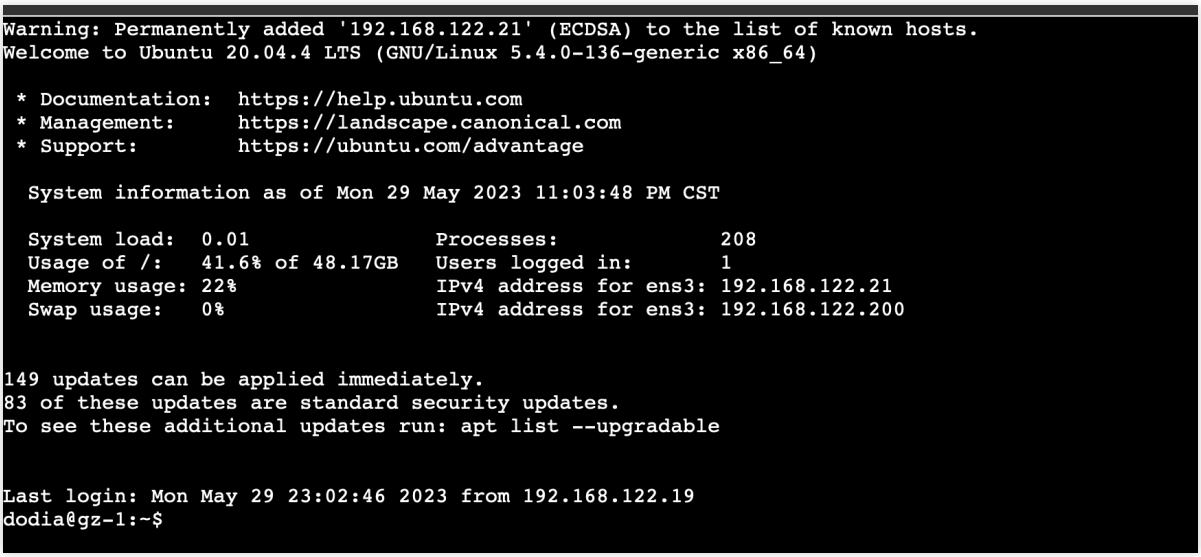
1. 登录 腾讯云容器服务控制台。
2. 在集群管理页面，单击需要远程登录节点的集群 ID，进入该集群详情页。
3. 选择页面左侧**节点管理** > **节点**，进入节点列表页面，确认**操作**列是否显示**登录**超链接，默认均不显示。如下图所示：



4. 在“节点列表”页面，选中需要登录的节点并单击**登录**，跳转至**节点远程登录控制台**页面。如下图所示：



密钥登录成功，会直接进入控制台页面，如下图所示：



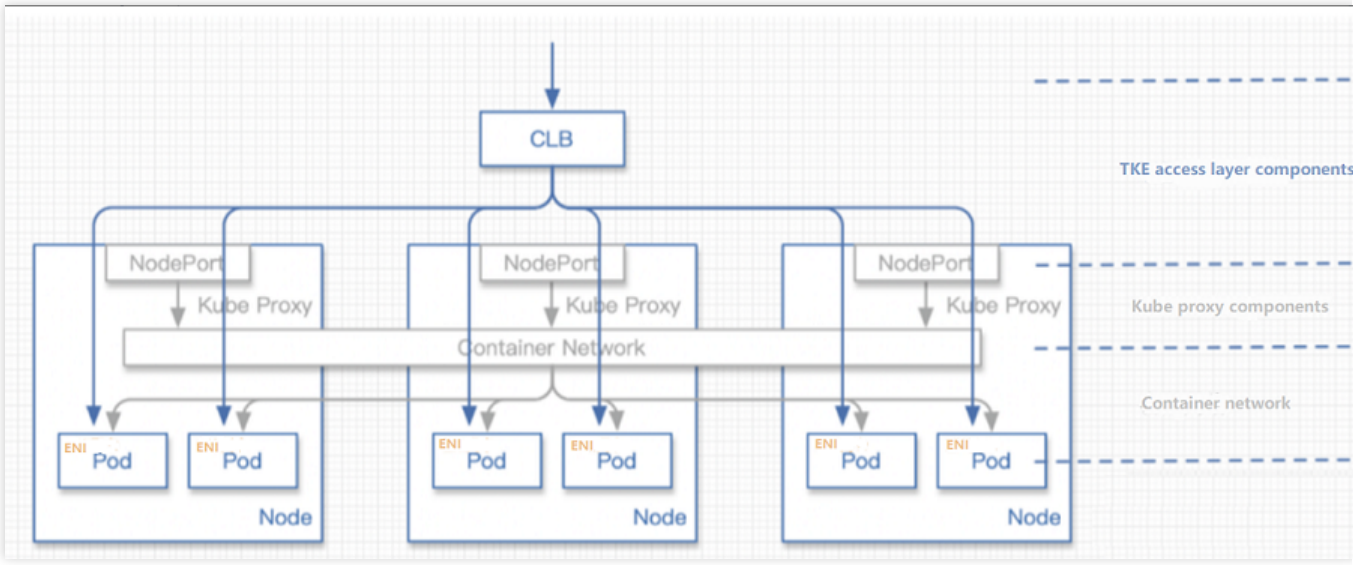
在控制台输入命令如 `kubectl get node` , 并按下**回车键**, 查看命令执行返回结果，如下图所示：

ENI 独立网卡

最近更新时间：2023-06-01 11:22:54

操作场景

如果用户将腾讯云上的 CVM 作为边缘节点加入边缘集群，平台支持启用 ENI 独立网卡，在 CVM 节点部署的 pod 中绑定 ENI 独立网卡，实现高可用网络方案。
具体架构如下图所示：

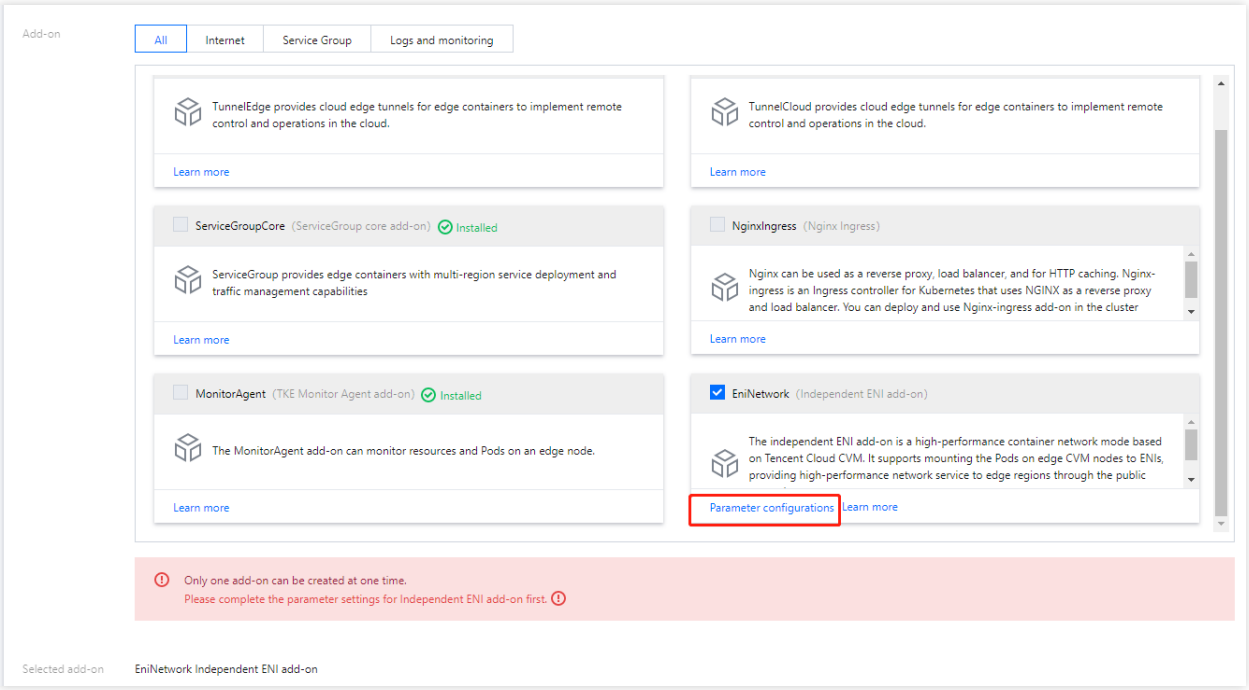


用户也可以在 CVM 所在 VPC 上使用 ENI 对外暴露 Pod，然后将不同的 ENI 网卡绑定到 CLB 上，使能高性能网络转发能力。
启用 ENI 独立网卡
关闭 ENI 独立网卡

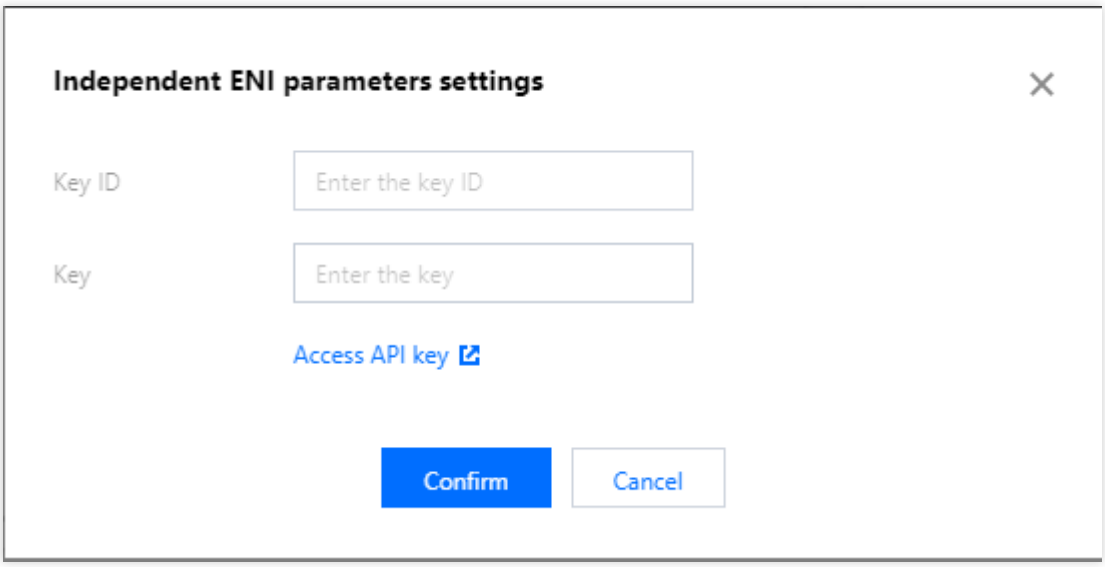
操作步骤

启用 ENI 独立网卡

1. 登录 腾讯云容器服务控制台。
2. 在集群管理页面，单击需要启用 ENI 独立网卡的集群 ID，进入该集群详情页。
3. 选择页面左侧**组件管理**，在组件列表页面单击**新建**。
4. 在新建组件管理页面，勾选 “EniNetwork（Eni 独立网卡组件）”，并单击**参数配置**。如下图所示：



4.1 在 “ENI 独立网卡参数设置”弹窗中，单击**访问 API 密钥**，跳转至密钥信息页面。



4.2 在 API密钥管理 页面中创建并复制 `SecretId` 和 `SecretKey` 。

4.3 在 “ENI 独立网卡参数设置”弹窗中，输入 `SecretId` 和 `SecretKey` 后，单击**确定**。

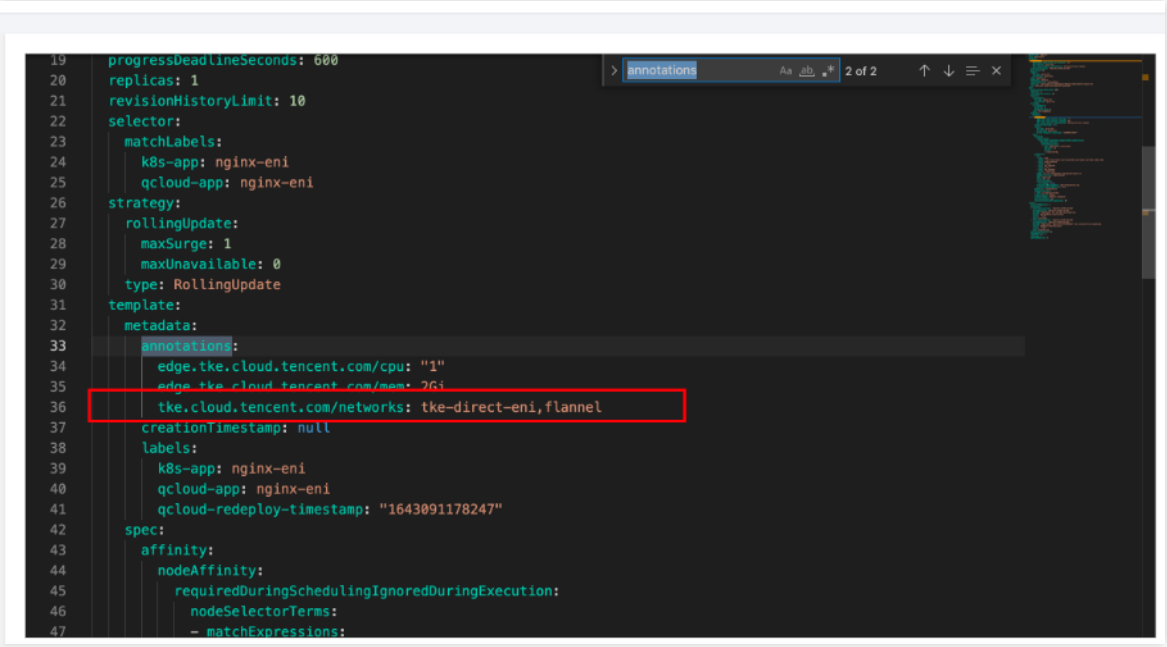
5. 在新建组件管理页面，单击**完成**，启用 ENI 网卡。

6. 选择页面左侧**工作负载 > Deployment**，进入 deployment 列表页。若列表中已有 deployment，则跳过该步骤；否则，新建 deployment。

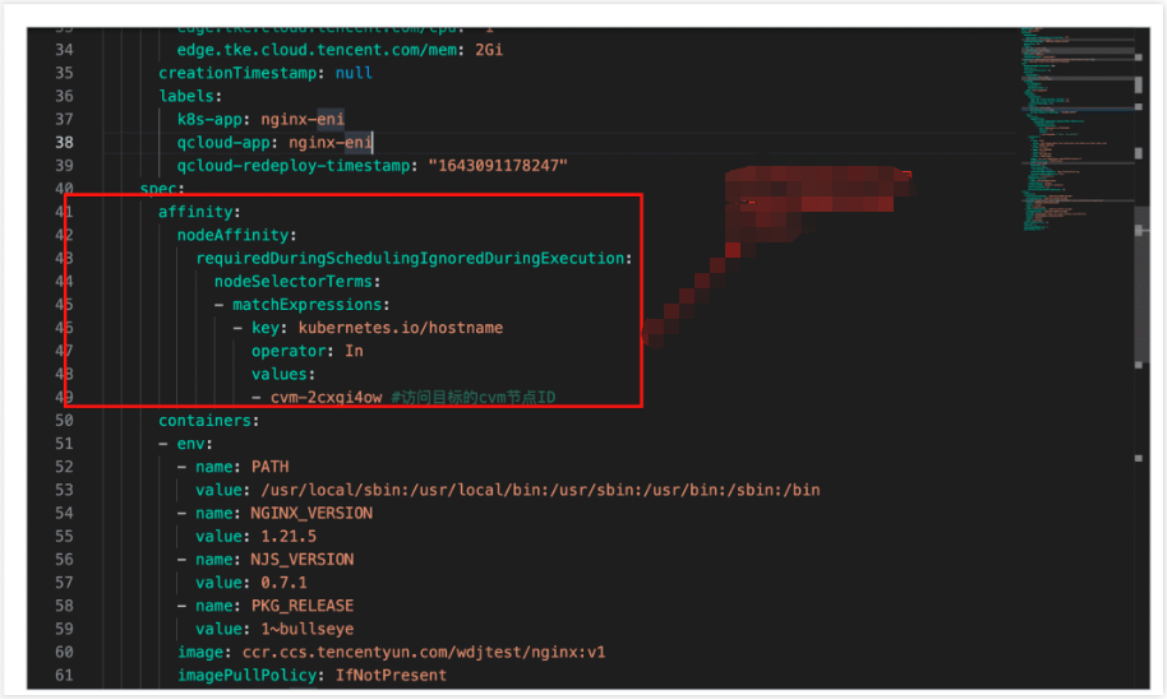
7. 选择页面左侧**节点管理 > 节点**，进入节点列表页。若列表中已有 CVM 节点，则跳过该步骤；否则，新建 CVM 节点。

8. 在目标边缘集群的 Pod 中配置 ENI。

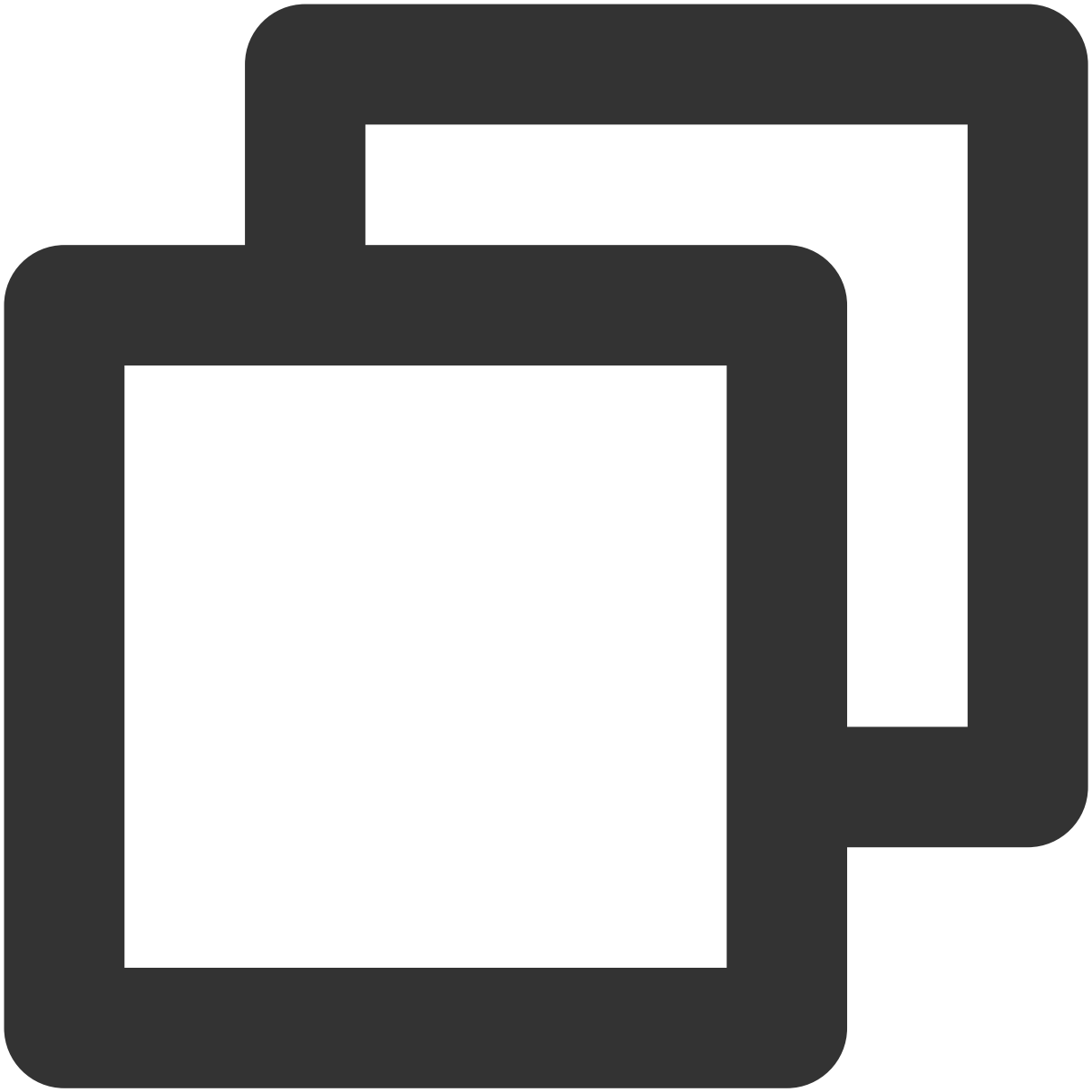
配置截图：



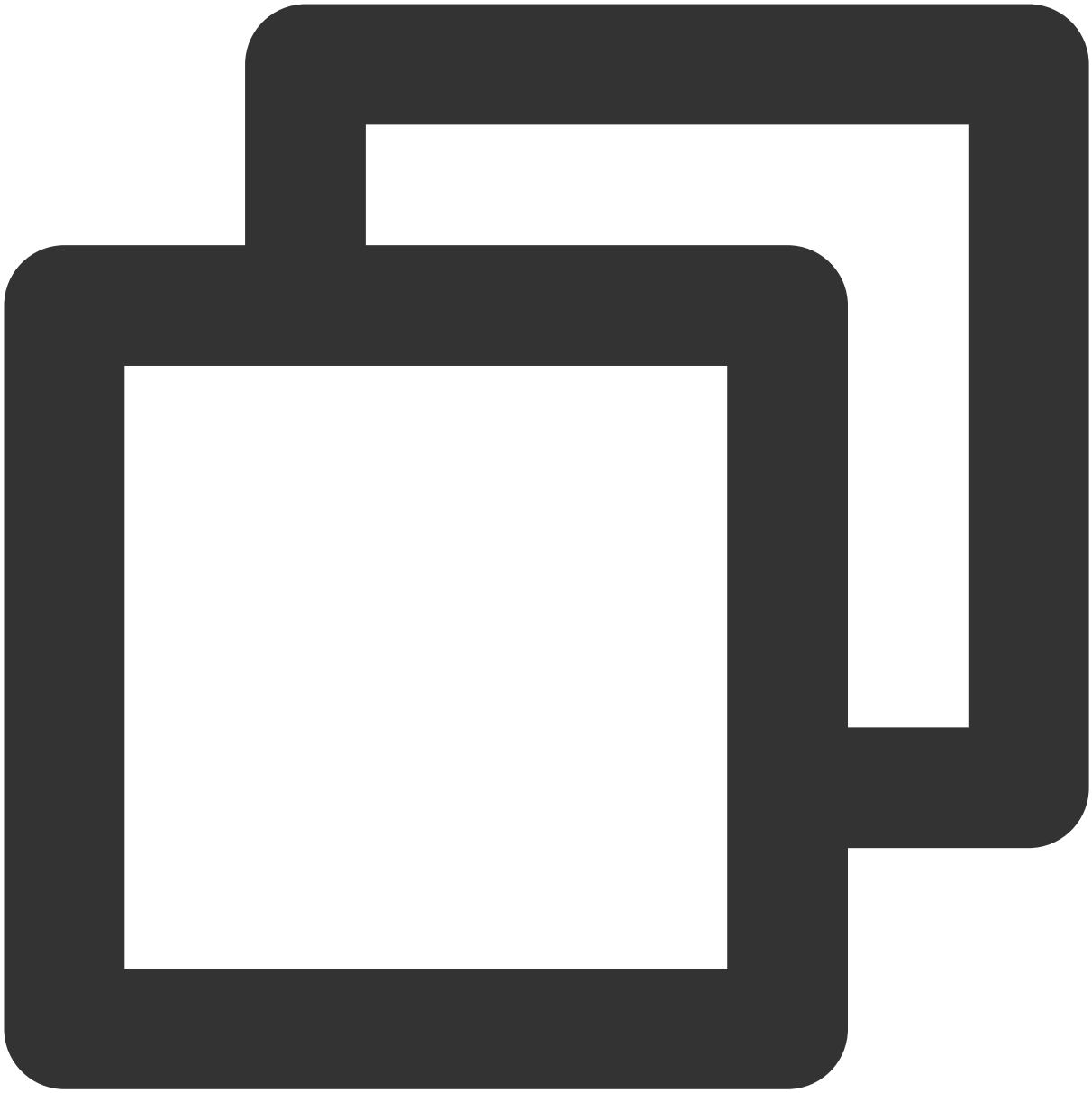
边缘节点 ENI 独立网卡能力，只能支持腾讯云 CVM 类型的节点资源，因此部署应用的时候，需要通过 nodeAffinity 的能力将挂载 ENI 独立网卡的应用 pod 调度到实际的边缘 CVM 节点上，如下图所示（支持填写多个 CVM 节点 ID）：



实际代码：



```
template:
  metadata:
    annotations:
      tke.cloud.tencent.com/networks: tke-direct-eni,flannel
```



```
spec:
  affinity:
    nodeAffinity:
      requiredDuringSchedulingIgnoredDuringExecution:
        nodeSelectorTerms:
        - matchExpressions:
          - key: kubernetes.io/hostname
            operator: In
            values:
            - cvm-2cxgi4ow #访问目标的 cvm 节点 ID
```

关闭 ENI 独立网卡

1. 登录 腾讯云容器服务控制台。
2. 在集群管理页面，单击需要关闭 ENI 独立网卡的集群 ID，进入该集群详情页。
3. 选择页面左侧**组件管理**，在组件列表页面单击组件右侧的**删除**。
4. 在“删除资源”弹窗中，单击**确定**，即可关闭 ENI 独立网卡功能。

边缘多地域部署
节点池管理
边缘节点池（NodeUnit）

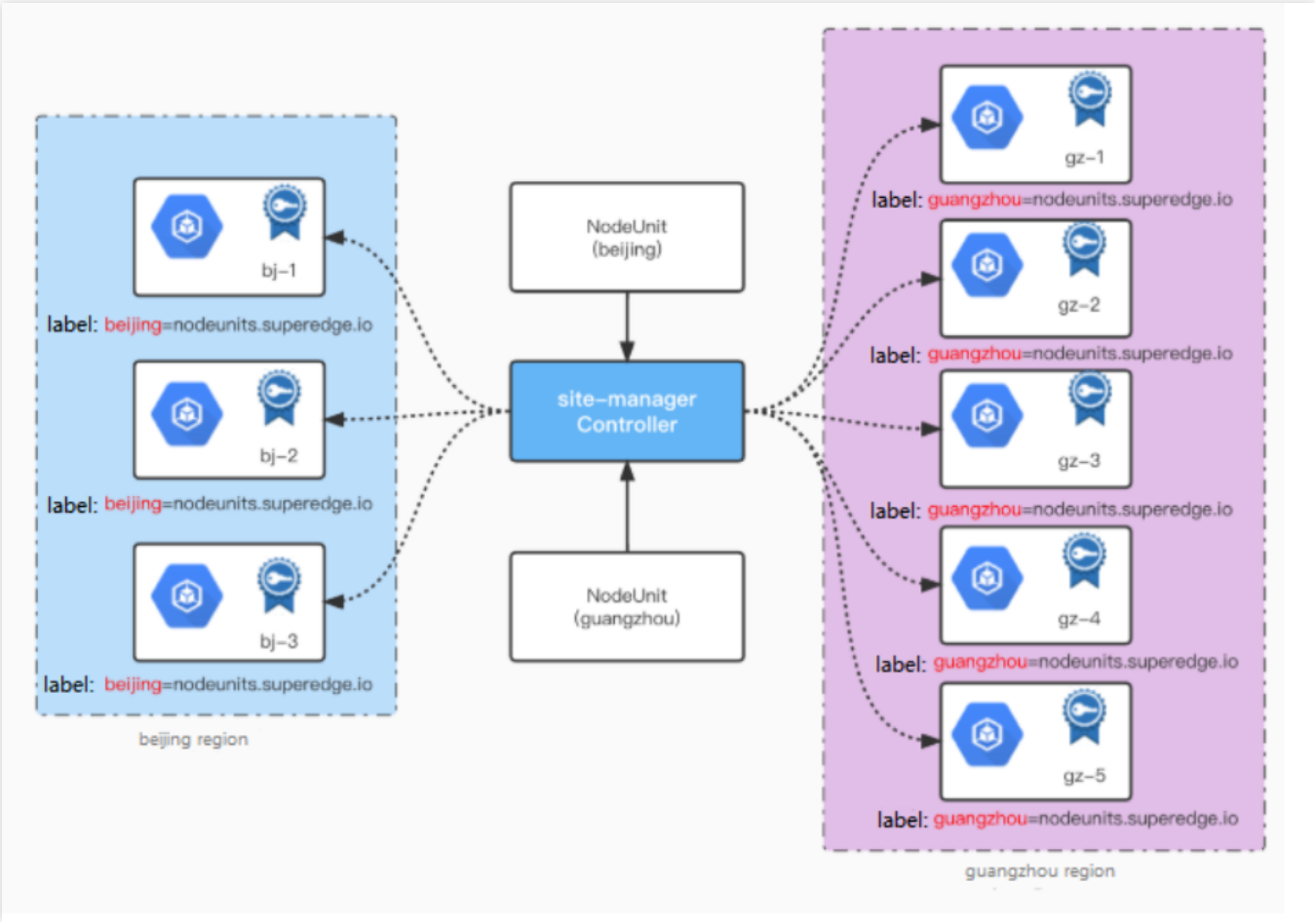
最近更新时间：2023-06-01 11:40:41

操作场景

本文介绍如何通过容器服务控制台，对边缘容器的边缘节点进行节点池管理。
本次更新重构了旧版本的 NodeGroup 和 NodeUnit 的界面交互，从2022年3月29日之后的新建集群将使用新的交互逻辑，使用旧版本的集群不受影响。其中，“边缘节点池”对应 SuperEdge 的 NodeUnit 设计，“节点池分类”对应 SuperEdge 的 NodeGroup 的设计。

概述

在边缘场景下，节点会被赋予更多的属性：例如不同的节点位于不同的网络环境、不同的体系结构、不同的云服务上等。在某些特定场景下，用户可以将节点分成不同的分组，每个分组的节点具有同样的特性：例如 3 个节点在 beijing 地域，另外 5 个节点在 guangzhou 地域，根据地域属性，我们可以分为 beijing 和 guanzhou 两个不同的节点池，不同的节点池可以调度部署不同的应用，我们将节点池命名为 `NodeUnit` 资源，作为 Kubernetes 的一个 CRD 资源实现。平台通过这个 CRD 和用户交互，通过 Kubernetes 给节点打标签来对节点进行分组管理，具体如下图所示：



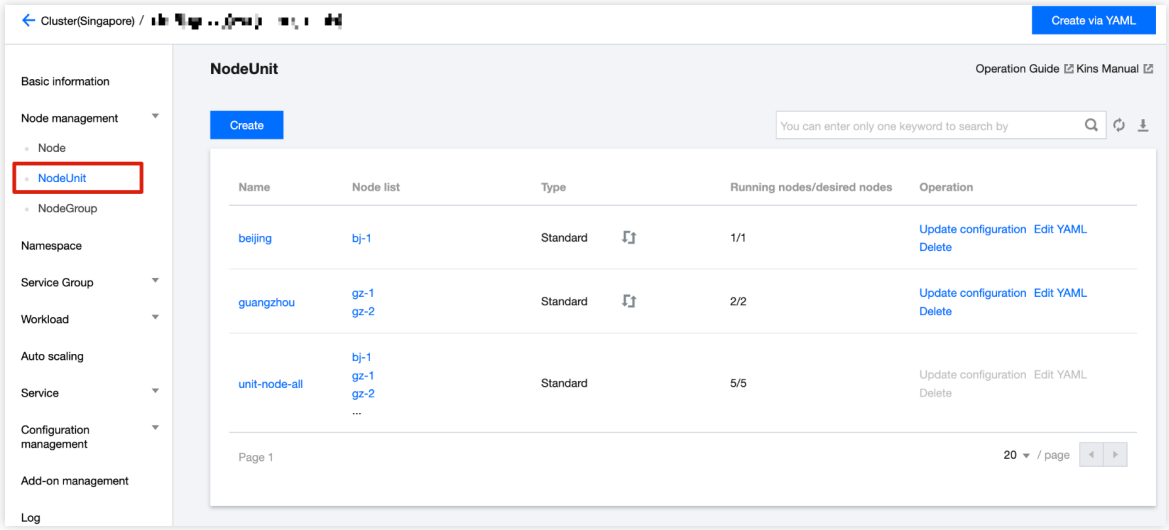
操作步骤

注意

您对边缘节点池的操作，都会对节点上的 Label 标签产生影响，请谨慎操作。

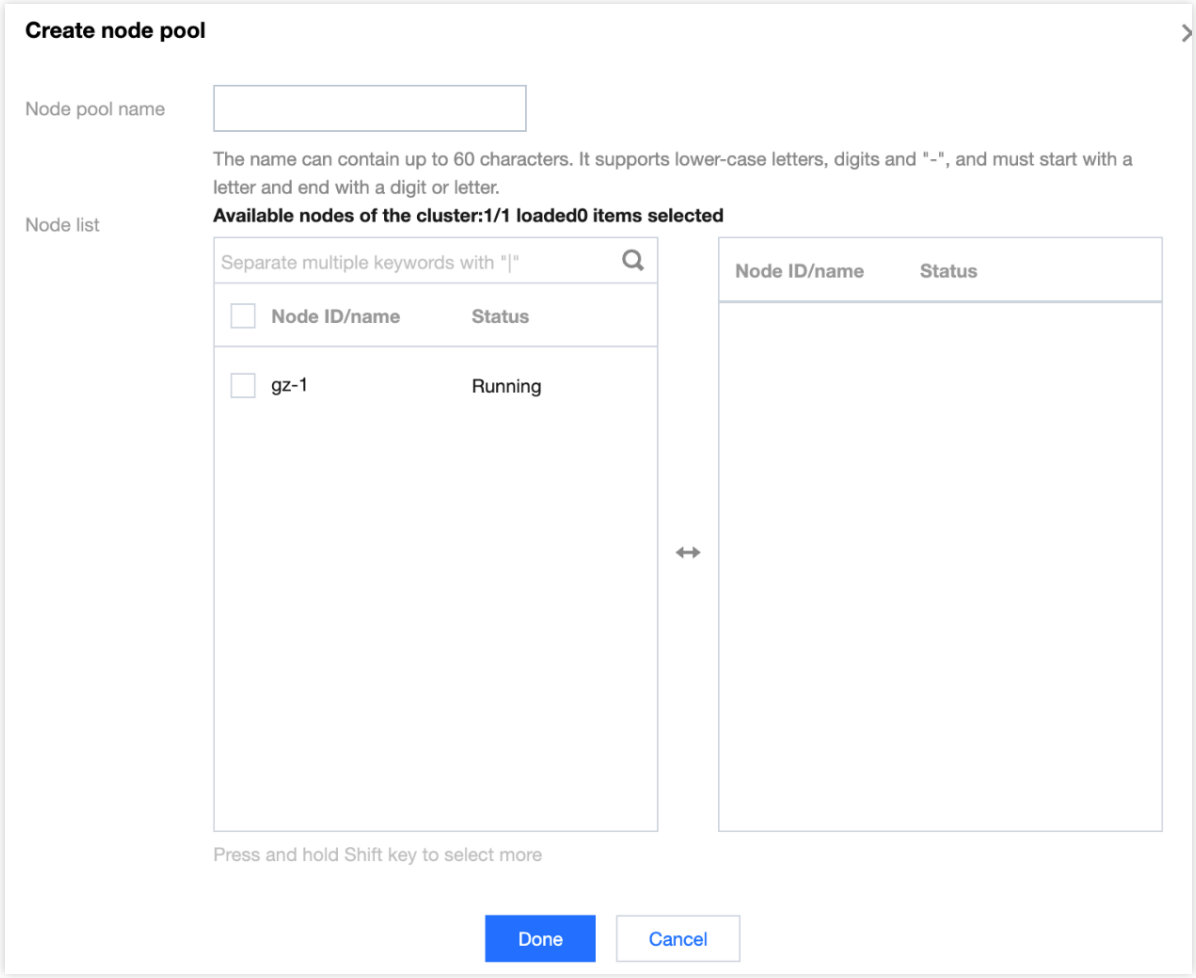
新建边缘节点池

1. 登录 腾讯云容器服务控制台。
2. 在集群管理页面，单击集群 ID，进入该集群详情页。
3. 选择页面左侧**节点管理 > 边缘节点池**，进入边缘节点池列表页面，如下图显示：



每个集群包含一个默认的节点池（unit-node-all），节点池中包含所有加入此集群的边缘节点。

4. 单击**新建**，在“创建节点池”页中输入节点池名称，并按照需求加入可用节点，如下图显示：



5. 单击**完成**，在边缘节点池列表中查看已添加节点。

管理边缘节点池

1. 登录 腾讯云容器服务控制台。
2. 在集群管理页面，单击集群 ID，进入该集群详情页。
3. 选择页面左侧**节点管理 > 边缘节点池**，进入边缘节点池列表页面。
4. 单击边缘节点池名称右侧的**更新配置**，在“更新节点池”页中对已有节点进行添加和删除。

删除边缘节点池

1. 登录 腾讯云容器服务控制台。
2. 在集群管理页面，单击集群 ID，进入该集群详情页。
3. 选择页面左侧**节点管理 > 边缘节点池**，进入边缘节点池列表页面。
4. 单击边缘节点池名称右侧的**删除**，即可删除指定的节点池。

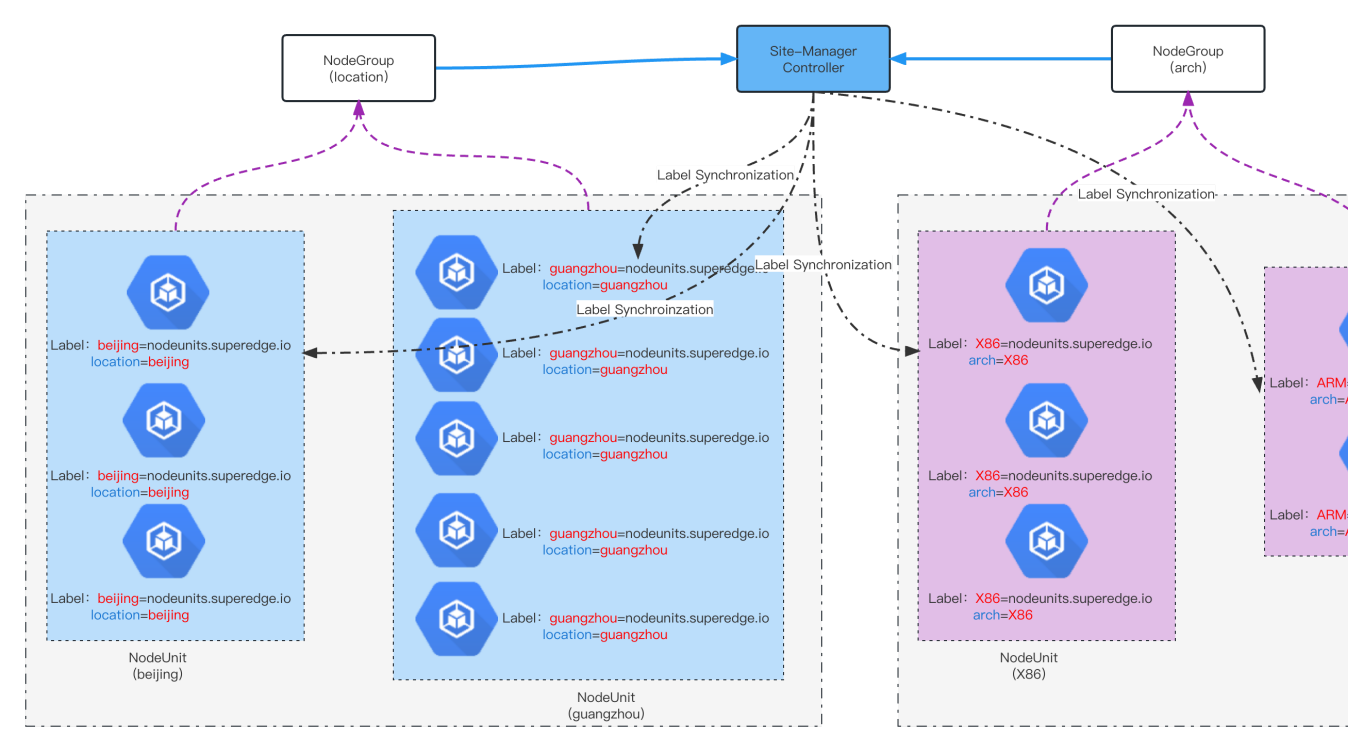
节点池分类（NodeGroup）

最近更新时间：2023-06-01 11:44:52

本文介绍如何通过容器服务控制台，对边缘容器的边缘节点池进行分组管理。

概述

节点池分类可以对不同的节点池再进行一级分组，将互斥的不同节点池加入到同一个分组后，应用可以指定在此分组上进行部署，实现一键批量在多个节点池部署应用的能力。
如下图：

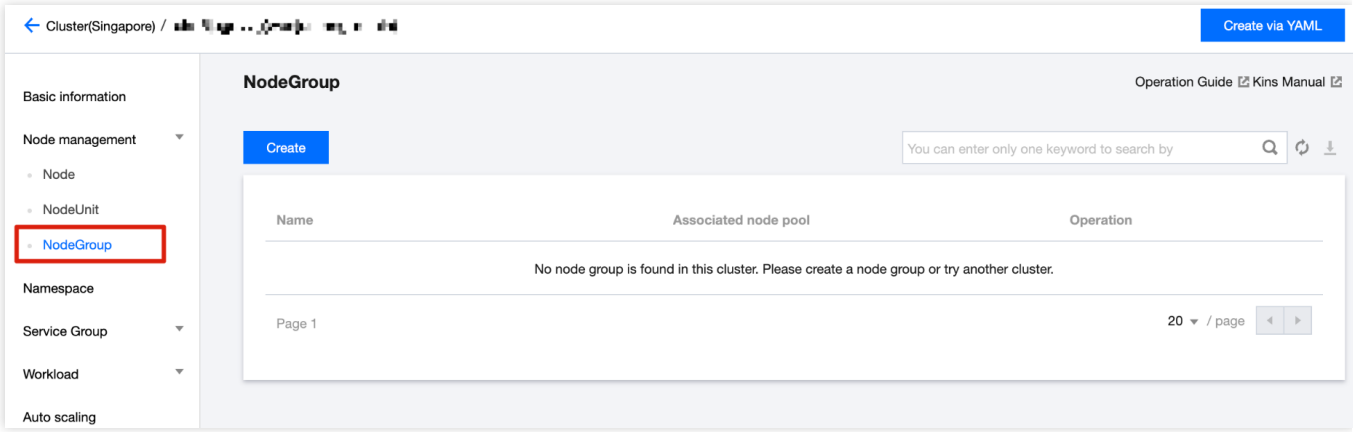


例如集群中的节点，您可以按照需求进行两种场景的分类，一种场景是按照地域（location），另一种场景是按照体系结构（arch）。每一种场景就是一个 NodeGroup 的 CR 实例 - `location` 和 `arch` 。然后可以将不同的 NodeUnit 加入到相应的 NodeGroup 分类中。
根据此结构创建不同的 NodeGroup 后，就可以使用边缘容器相关的 DeploymentGrid/StatefusetGrid/ServiceGrid 能力进行多地域的分发部署。例如您可以创建 Nginx 实例的 DeploymentGrid，绑定到 `location` 的 NodeGroup 中，beijing 和 guangzhou 这两个NodeUnit 都会同时部署完全相同的 Nginx 的 Deployment 实例，在两个地域同时提供接入服务。

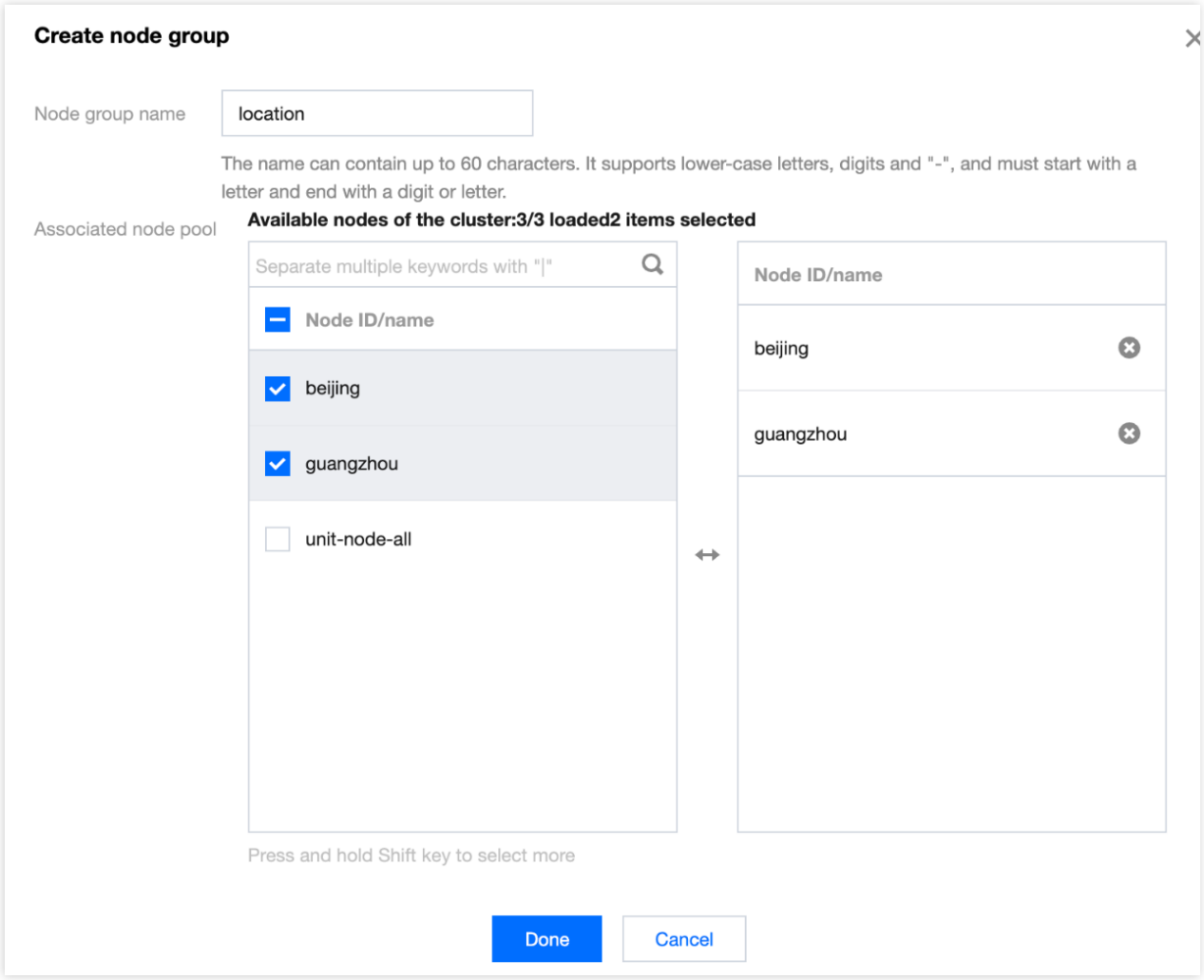
操作步骤

新建节点池分类

1. 登录 腾讯云容器服务控制台。
2. 在集群管理页面，单击集群 ID，进入该集群详情页。
3. 选择页左侧**节点管理 > 节点池分类**，进入节点池分类列表页，如下图显示：



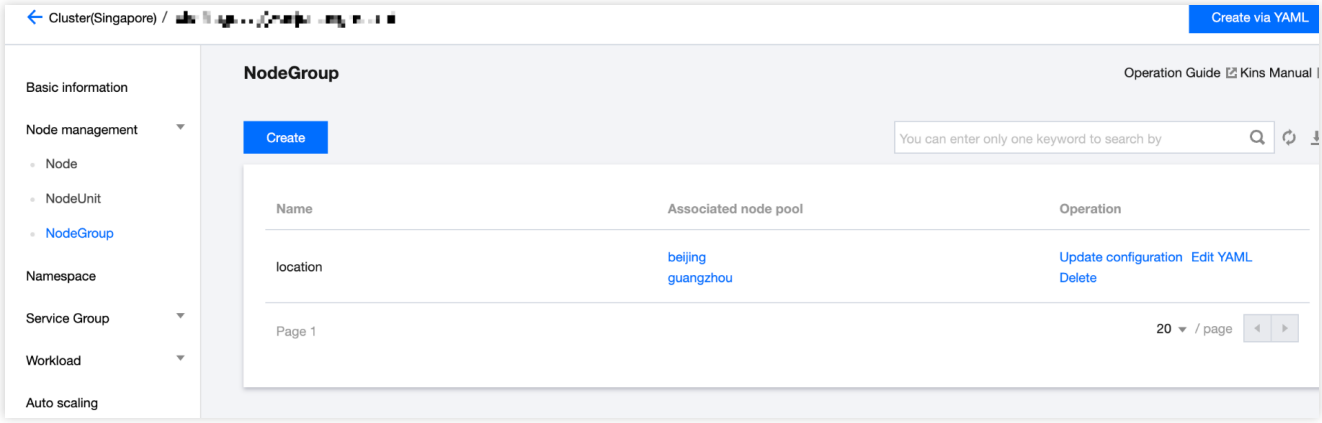
4. 单击**新建**，在“新建节点池分类”页中输入节点池分类名称，并按照需求关联节点池。如下图显示：



5. 单击**完成**，在节点池分类列表中查看分类。

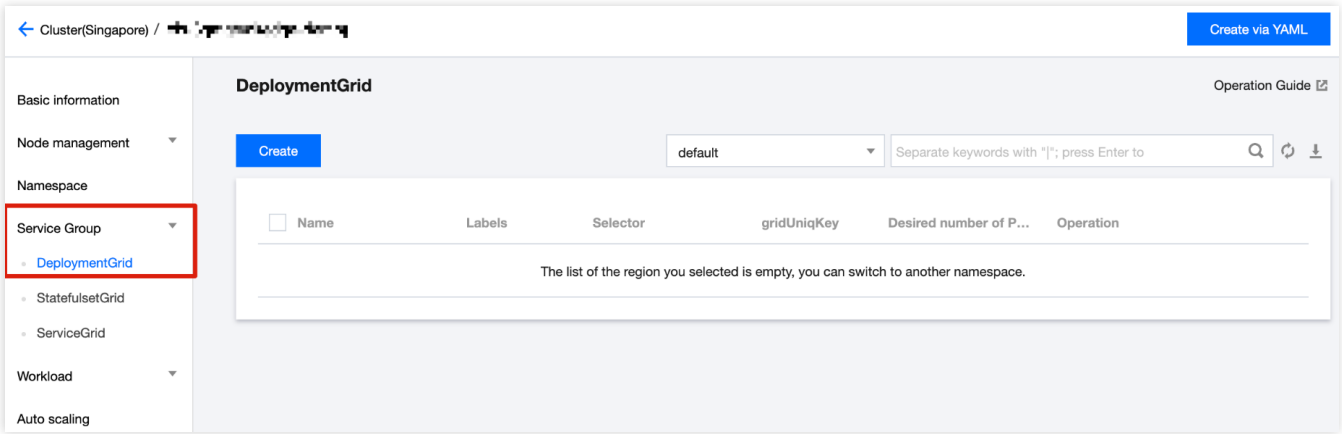
管理节点池分类

1. 登录 腾讯云容器服务控制台。
2. 在集群管理页面，单击集群 ID，进入该集群详情页。
3. 选择页左侧**节点管理 > 节点池分类**，进入节点池分类列表页。
4. 点击**右侧操作>更新配置**，在“修改节点池分类（NodeGroup）”页中对指定节点池分类进行修改操作。

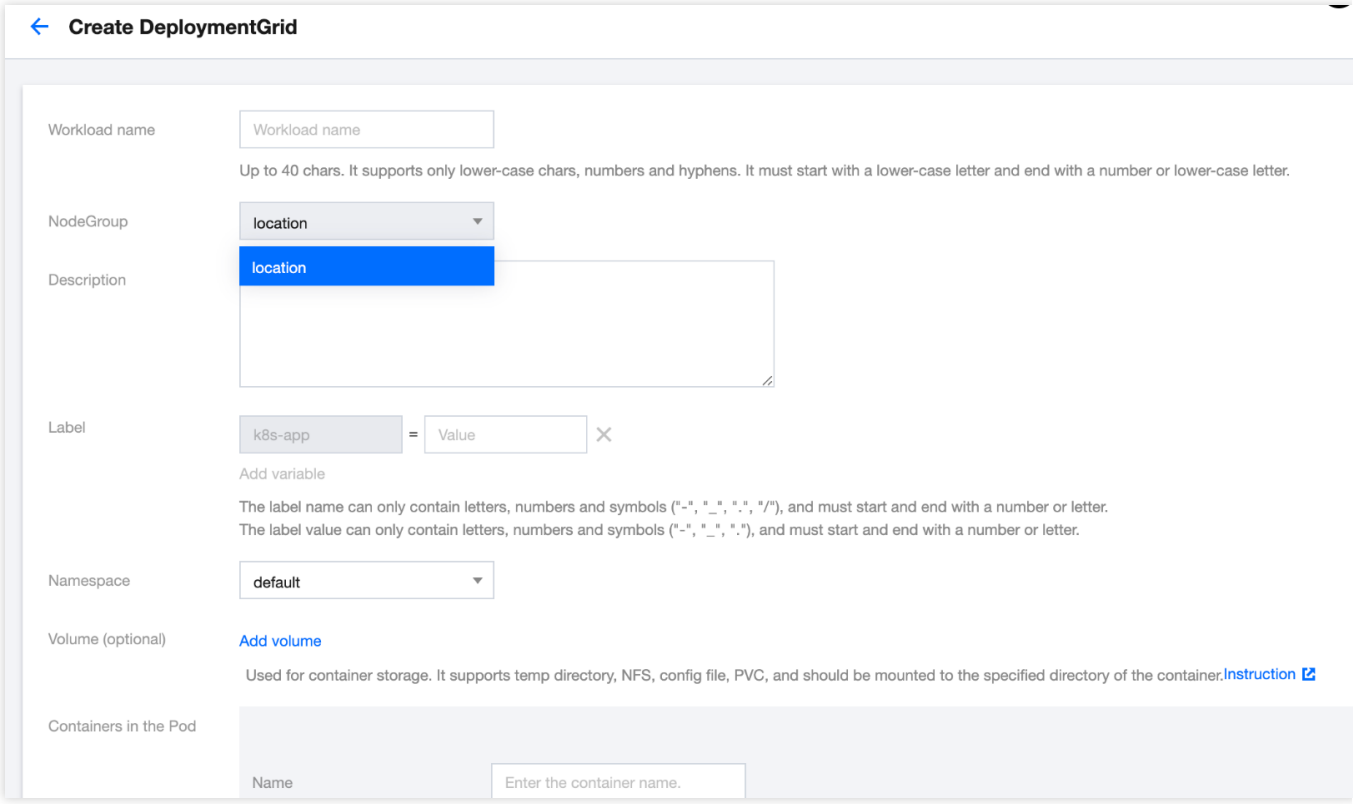


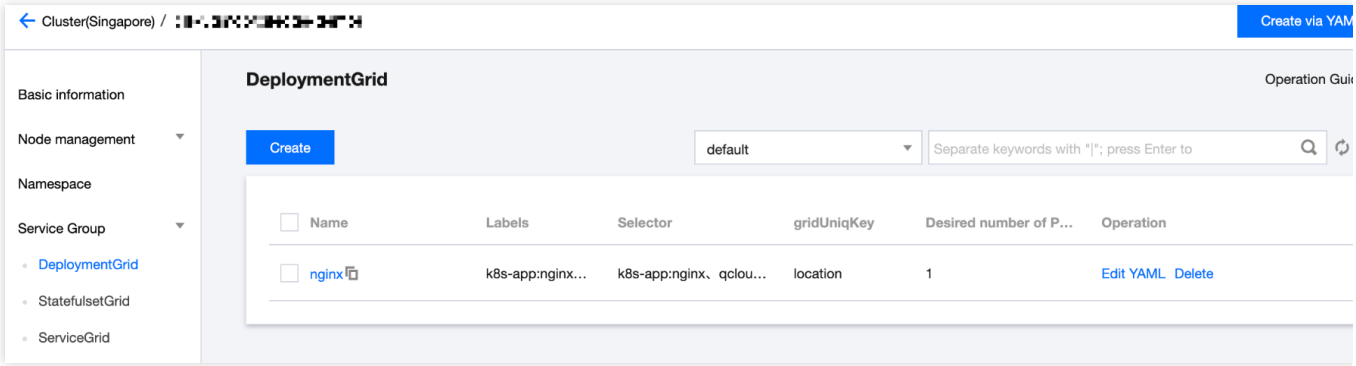
使用节点池分类

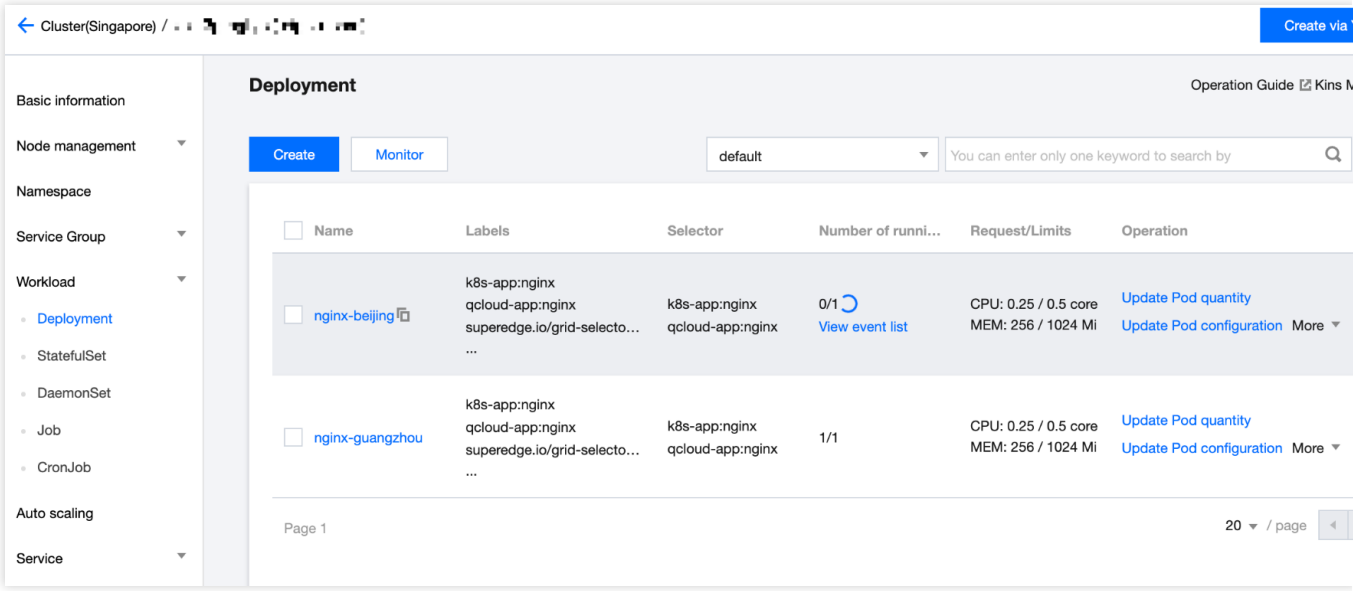
1. 登录 腾讯云容器服务控制台。
2. 在集群管理页面，单击集群 ID，进入该集群详情页。
3. 选择页左侧**Service Group > DeploymentGrid**，进入 DeploymentGrid 列表页。



4. 单击**新建**，在“新建DeploymentGrid”中按照需求输入 DeploymentGrid 相关参数，选择相应的 NodeGroup 即可进行部署，如下图所示：


Create DeploymentGrid
Workload name
Workload name
Up to 40 chars. It supports only lower-case chars, numbers and hyphens. It must start with a lower-case letter and end with a number or lower-case letter.
NodeGroup
location
location
Description
Label
k8s-app
=
Value
Add variable
The label name can only contain letters, numbers and symbols ("-", "_", ".", "/"), and must start and end with a number or letter.
The label value can only contain letters, numbers and symbols ("-", "_", "."), and must start and end with a number or letter.
Namespace
default
Volume (optional)
Add volume
Used for container storage. It supports temp directory, NFS, config file, PVC, and should be mounted to the specified directory of the container.Instruction
Containers in the Pod
Name
Enter the container name.


Cluster(Singapore) /
Create via YAM
Basic information
Node management
Namespace
Service Group
DeploymentGrid
StatefulsetGrid
ServiceGrid
DeploymentGrid
Operation Gui
Create
default
Separate keywords with "|"; press Enter to
Name
Labels
Selector
gridUniqKey
Desired number of P...
Operation
nginx
k8s-app:nginx...
k8s-app:nginx、qclou...
location
1
Edit YAML
Delete


Cluster(Singapore) /
Create via
Basic information
Node management
Namespace
Service Group
Workload
Deployment
StatefulSet
DaemonSet
Job
CronJob
Auto scaling
Service
Deployment
Operation Guide
Create
Monitor
default
You can enter only one keyword to search by
Name
Labels
Selector
Number of runni...
Request/Limits
Operation
nginx-beijing
k8s-app:nginx
qcloud-app:nginx
superedge.io/grid-selecto...
...
k8s-app:nginx
qcloud-app:nginx
0/1
View event list
CPU: 0.25 / 0.5 core
MEM: 256 / 1024 Mi
Update Pod quantity
Update Pod configuration
More
nginx-guangzhou
k8s-app:nginx
qcloud-app:nginx
superedge.io/grid-selecto...
...
k8s-app:nginx
qcloud-app:nginx
1/1
CPU: 0.25 / 0.5 core
MEM: 256 / 1024 Mi
Update Pod quantity
Update Pod configuration
More
Page 1
20
/ page

ServiceGroup 多地域部署

最近更新时间：2023-07-14 18:26:16

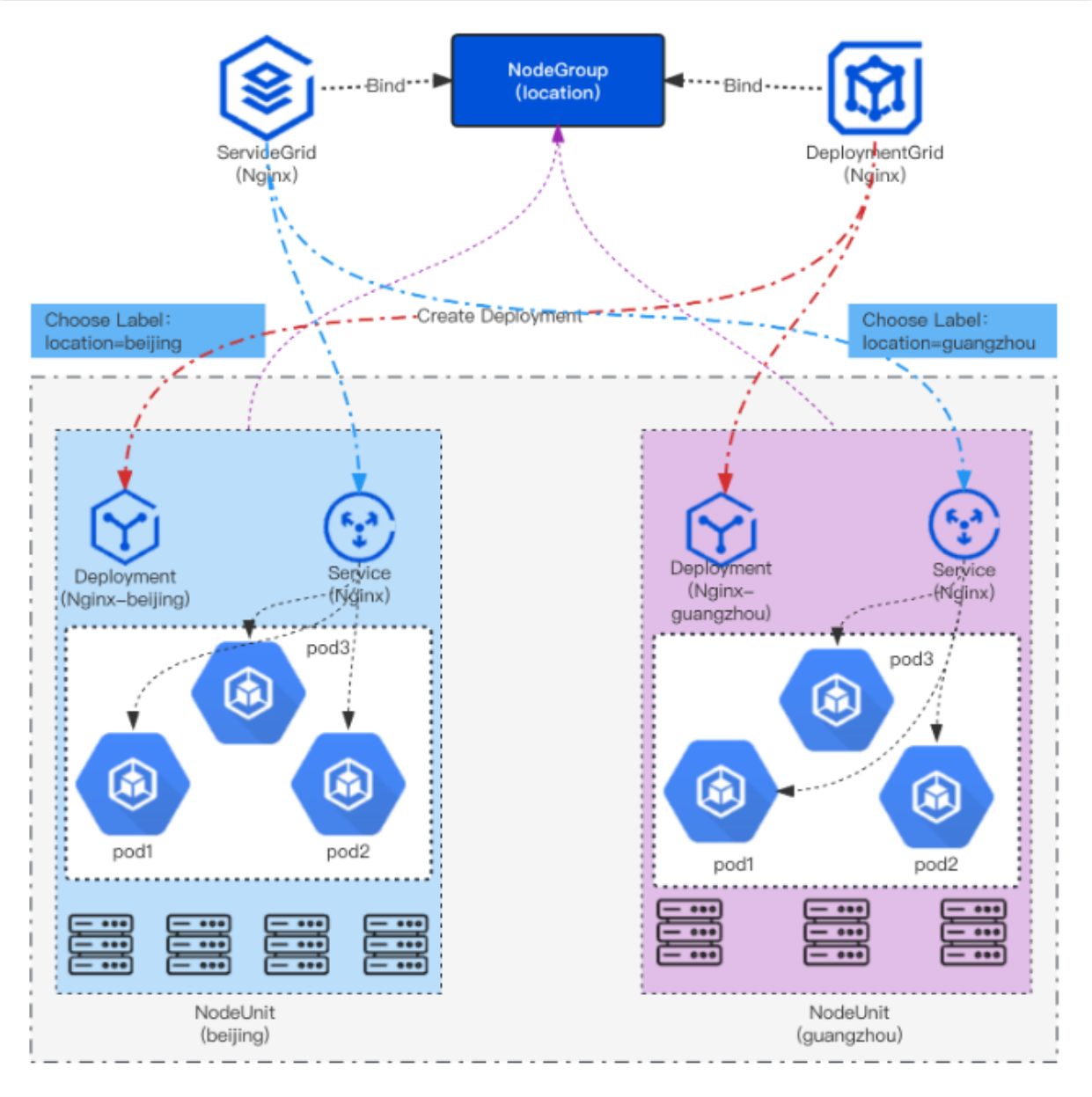
功能特点

边缘计算场景中，往往会在同一个集群中管理多个边缘站点，每个边缘站点内有一个或多个计算节点。
同时希望在每个站点中都运行一组有业务逻辑联系的服务，每个站点内的服务是一套完整的功能，可以为用户提供服务。
由于受到网络限制，有业务联系的服务之间不希望或者不能跨站点访问。
由于以上边缘计算的 3 个特点，腾讯云边缘容器专门设计了一套 ServiceGroup 的自定义资源逻辑，来解决边缘容器在多地域场景下遇到的应用分发和服务治理的问题。

操作场景

ServiceGroup 可以便捷地在共属同一个集群的不同机房或区域中各自部署一组服务，并且使得各个服务间的请求在本机房或本地域内部即可完成，避免服务跨地域访问。
原生 Kubernetes 无法控制 Deployment 的 Pod 创建的具体节点位置，需要通过统筹规划节点的亲和性来间接完成。当边缘站点数量以及需要部署的服务数量过多时，管理和部署方面的极为复杂，甚至仅存在理论上的可能性。与此同时，为了将服务间的相互调用限制在一定范围，业务方需要为各个 Deployment 分别创建专属的 Service，管理方面的工作量巨大且极容易出错并引起线上业务异常。
ServiceGroup 针对此场景设计，用户仅需使用 ServiceGroup 提供的 DeploymentGrid 和 ServiceGrid 两种 TKE Edge 自研 Kubernetes 资源，即可方便地将服务分别部署到这些节点组中，并进行服务流量管控，同时还可保证各区域服务数量及容灾。

整体架构



基本概念

ServiceGroup 需要和 NodeUnit 以及 NodeGroup 结合使用，概念详情如下：

NodeUnit（边缘节点池）

NodeUnit 通常是位于同一边缘站点内的一个或多个计算资源实例，需要保证同一 NodeUnit 中的节点内网是通的。
更多内容可参考 边缘节点池。

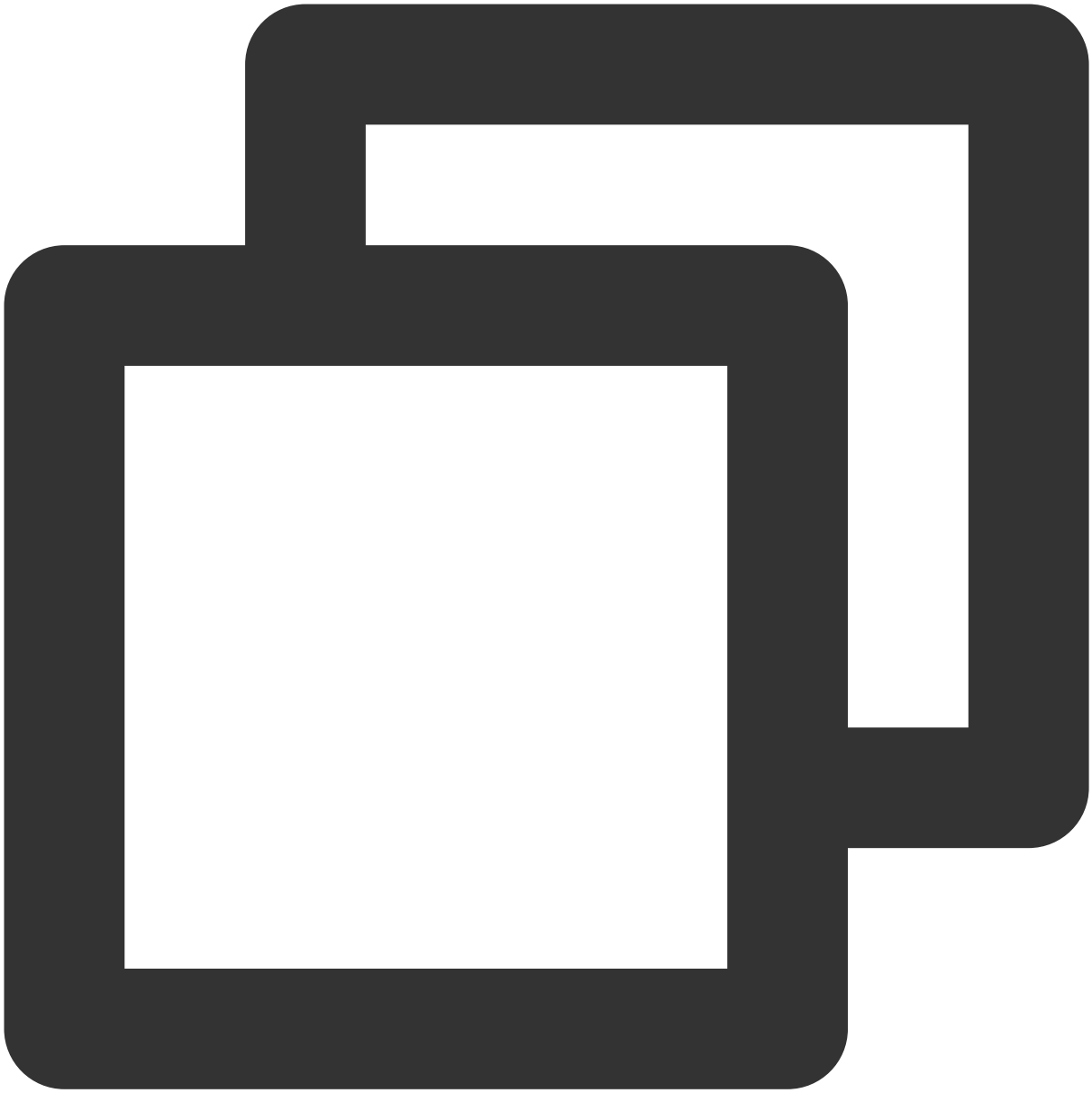
NodeGroup（边缘节点池分类）

NodeGroup 包含一个或者多个 NodeUnit。
保证在集合中每个 NodeUnit 上均部署 ServiceGroup 中的服务。

集群中增加 NodeUnit 时自动将 ServiceGroup 中的服务部署到新增 NodeUnit。
更多内容可参考 节点池分类。

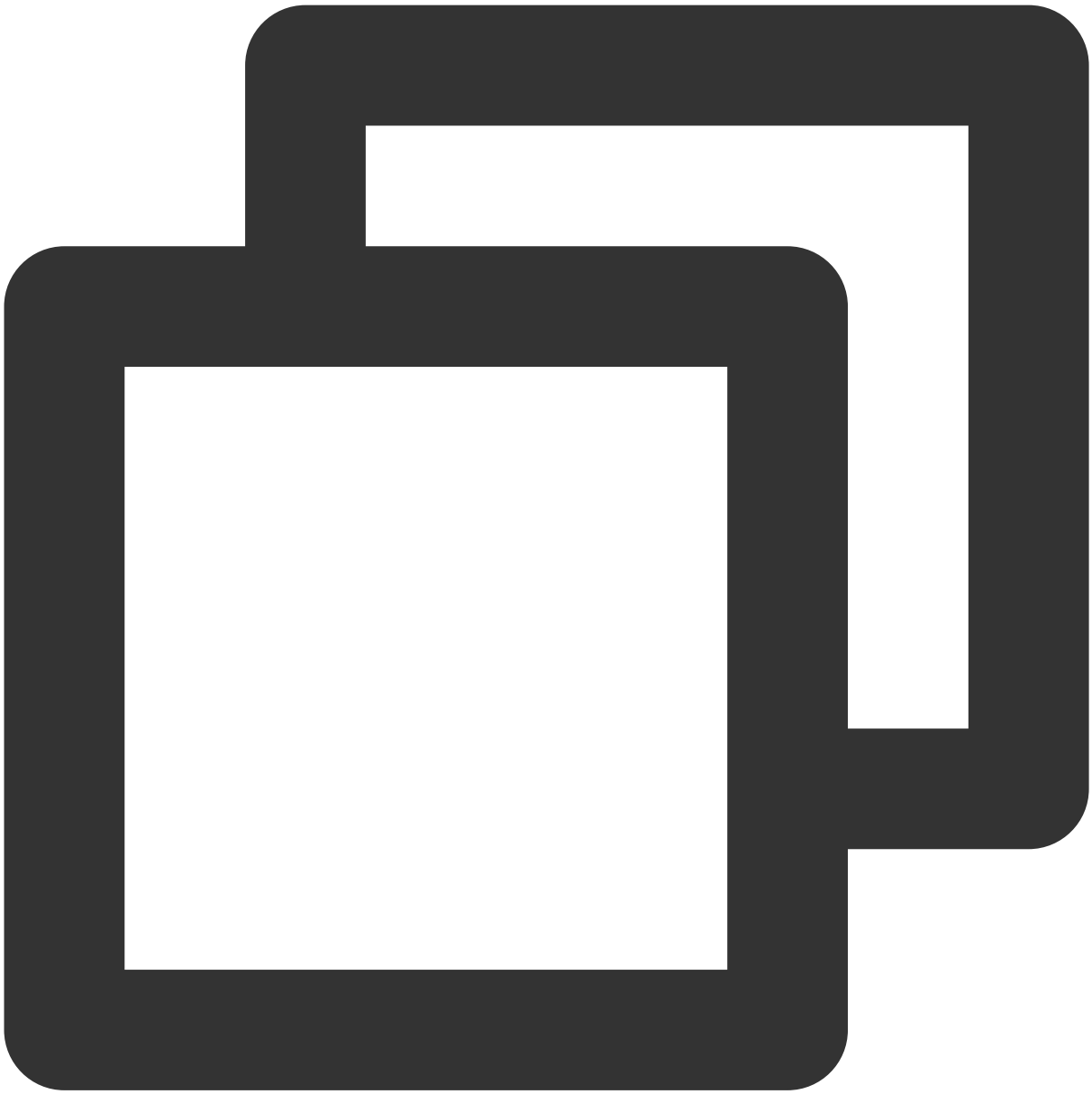
ServiceGroup

ServiceGroup 并不是一个实体的资源定义，是集中 Kubernetes 自定义资源的集合。ServiceGroup 是一种抽象资源，一个集群中可以创建多个 ServiceGroup。
ServiceGroup 包含一个或者多个业务服务。适用场景如下：
业务需要打包部署。
业务需要在每一个 NodeUnit 中运行起来并且保证 pod 数量。
业务需要将服务之间的调用控制在同一个 NodeUnit 中，不能将流量转发到其他 NodeUnit。
ServiceGroup 涉及的资源类型包括如下三类：
DeploymentGrid
StatefulSetGrid
ServiceGrid
DeploymentGrid 的格式与 Deployment 类似，<deployment-template> 字段就是原先 deployment 的 template 字段，比较特殊的是 gridUniqKey 字段，该字段指明了节点分组的 label 的 key 值：



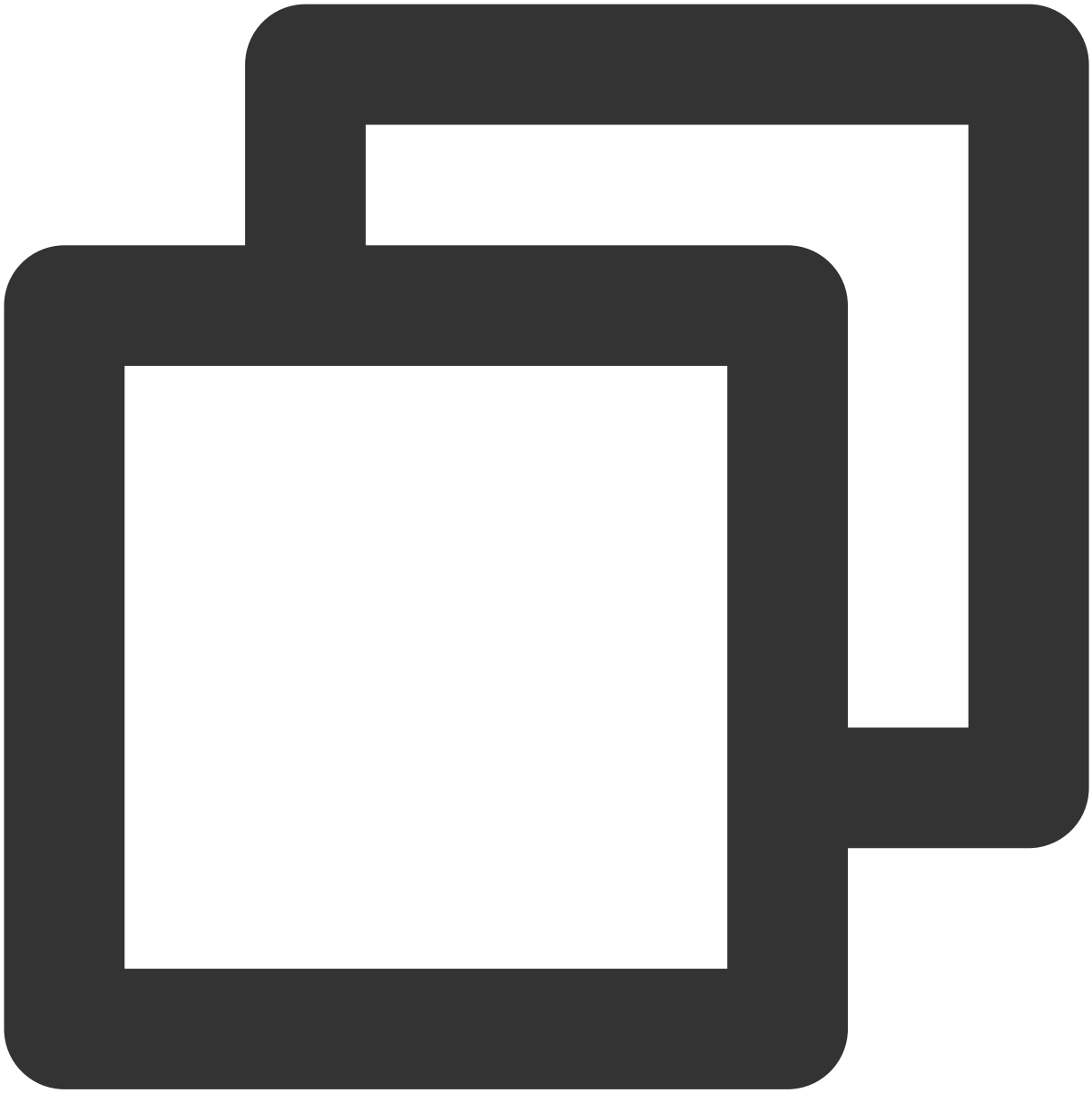
```
apiVersion: superedge.io/v1
kind: DeploymentGrid
metadata:
  name:
  namespace:
spec:
  gridUniqKey: <NodeLabel Key>
  <deployment-template>
```

StatefulSetGrid 的格式与 StatefulSet 类似，<statefulset-template>字段就是原先 statefulset 的 template 字段，比较特殊的是 gridUniqKey 字段，该字段指明了节点分组的 label 的 key 值：



```
apiVersion: superedge.io/v1
kind: StatefulSetGrid
metadata:
  name:
  namespace:
spec:
  gridUniqKey: <NodeLabel Key>
  <statefulset-template>
```

ServiceGrid 的格式与 Service 类似，<service-template>字段就是原先 service 的 template 字段，比较特殊的是 gridUniqKey 字段，该字段指明了节点分组的 label 的 key 值：



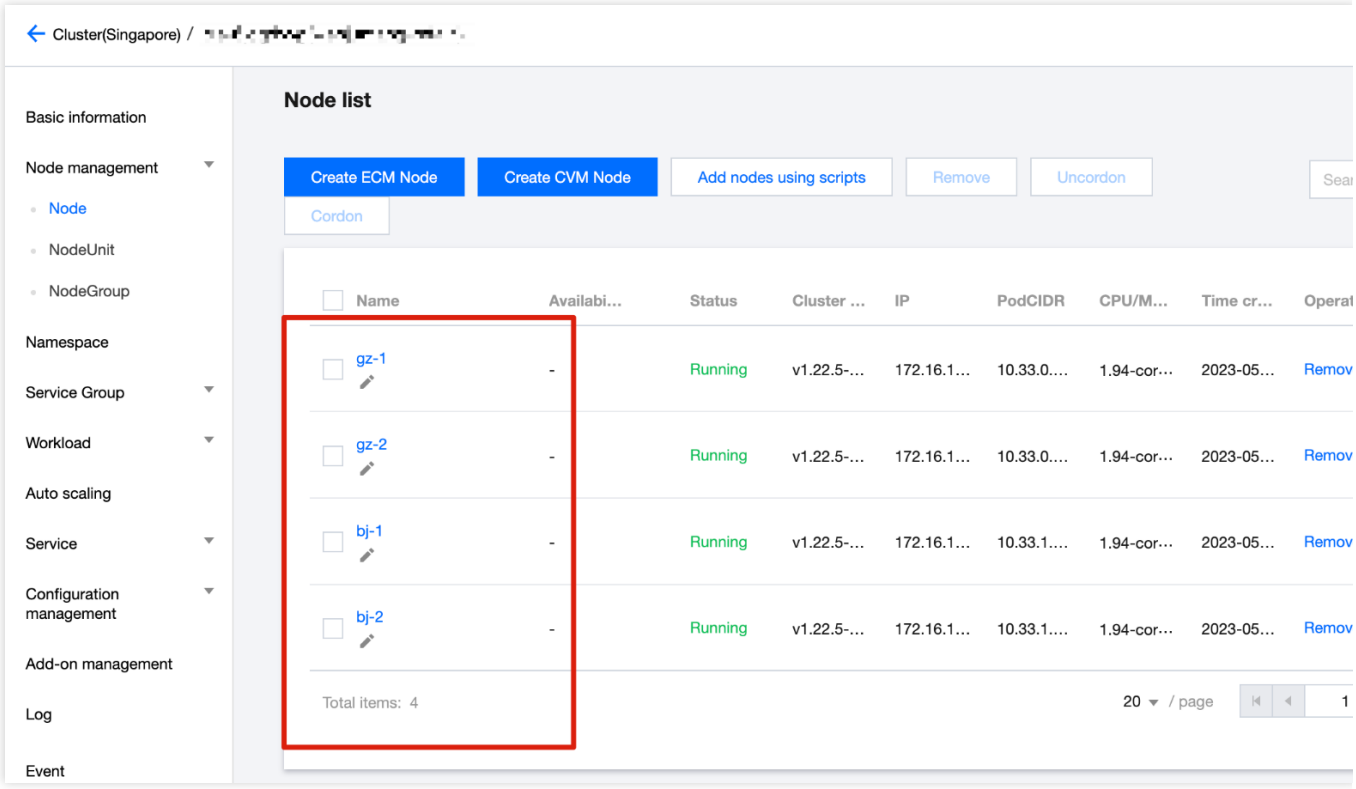
```
apiVersion: superedge.io/v1
kind: ServiceGrid
metadata:
  name:
  namespace:
spec:
  gridUniqKey: <NodeLabel Key>
  <service-template>
```

操作步骤

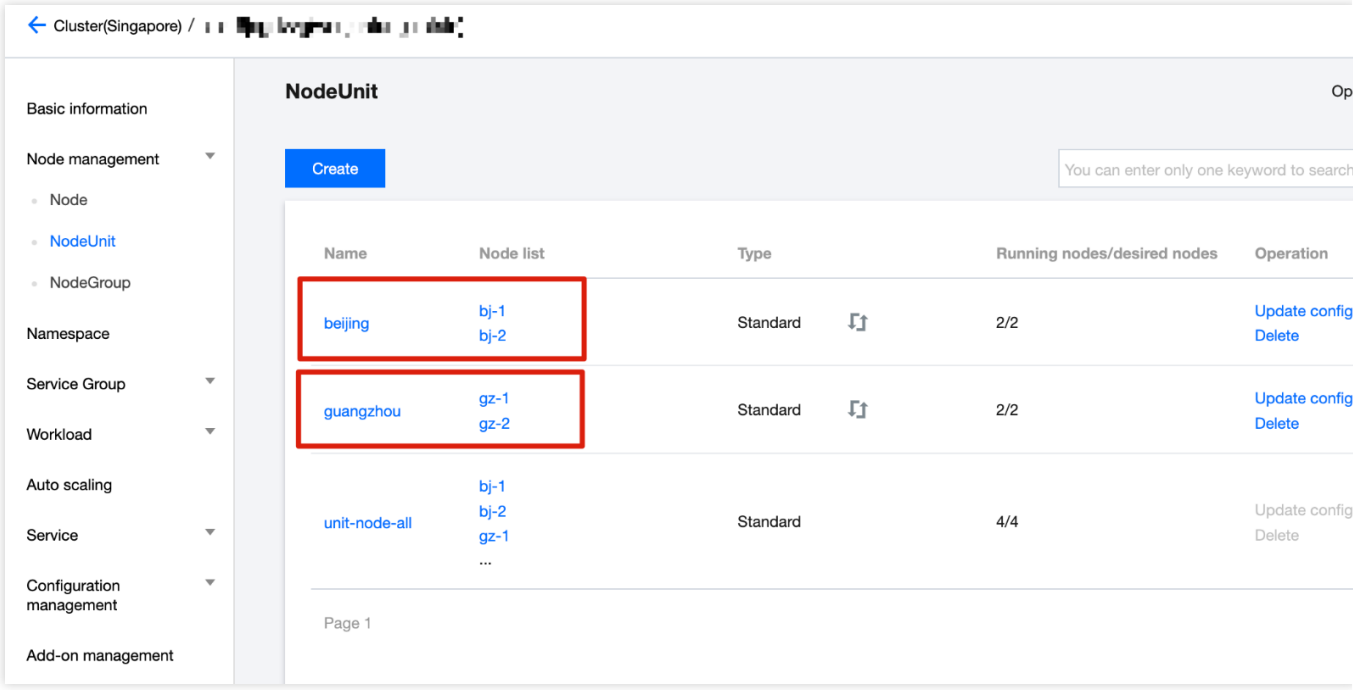
以在边缘部署 Nginx 服务为例，我们希望在多个节点池内分别一套完整的 Nginx 服务，需要如下操作：

将边缘节点分组

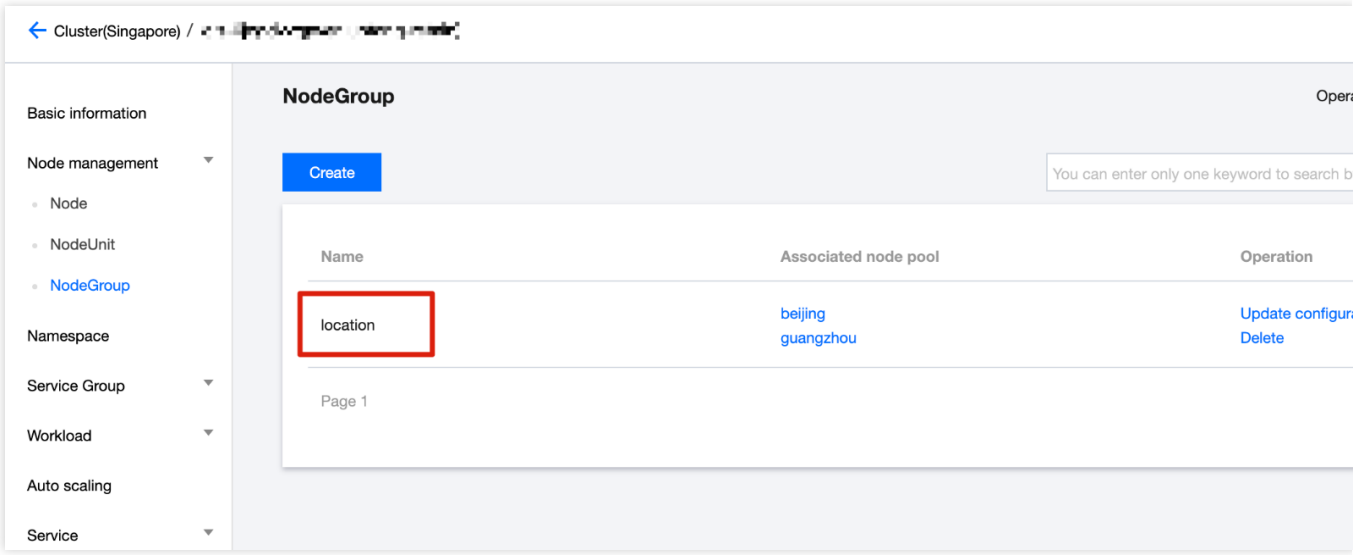
我们以一个边缘集群为例，将集群中的节点添加到**边缘节点池**以及**节点池分类**中。
此集群包含 5 个边缘节点，分别位于 `beijingguangzhou` 2 个地域，节点名为 `bj-1` 、 `bj-2` 、 `gz-1` 、 `gz-2` 、 `gz-3` 。如下图所示：



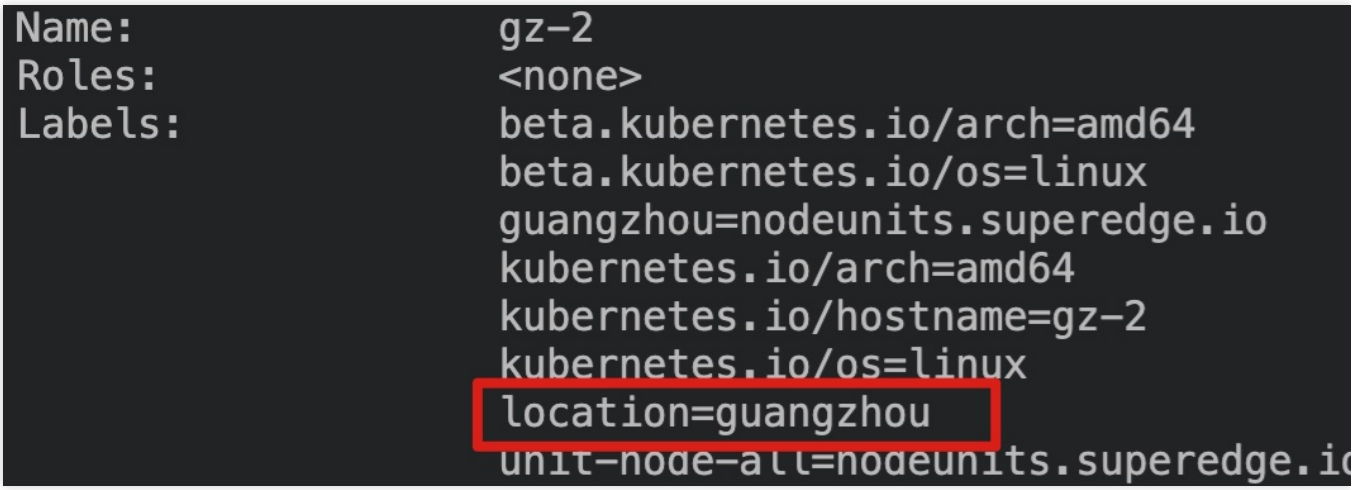
分别创建 2 个 NodeUnit（边缘节点池）： `beijing` 、 `guangzhou` ，分别将相应的节点加入对应的 NodeUnit（边缘节点池）中，如下图所示：



创建名称为 `location` 的 NodeGroup（边缘节点池分类），将 `beijing`、`guangzhou` 这两个边缘节点池划分到 `location` 这个分类中，如下图所示：



进行上述操作后，每个节点上会被打上相应的标签，节点 gz-2 的标签如下图所示：



说明

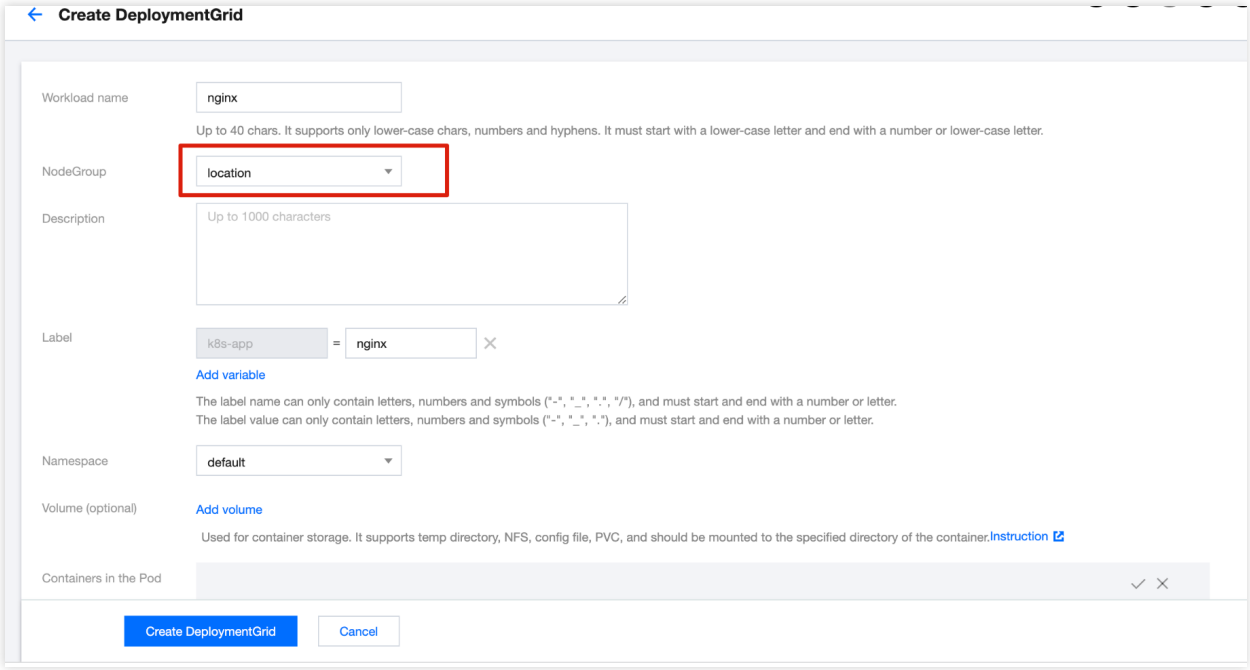
label 的 key 就是 NodeGroup 的名字，value 是 NodeUnit 的名字，value 相同的节点表示属于同一个 NodeUnit。

如果同一个集群中有多个 NodeGroup 请创建不同的 NodeGroup 名字作为唯一标记，部署 ServiceGroup 相关资源的时候会通过 NodeGroup 的名字这个唯一标记来绑定指定的 NodeGroup 进行部署。

无状态 ServiceGroup

部署 DeploymentGrid

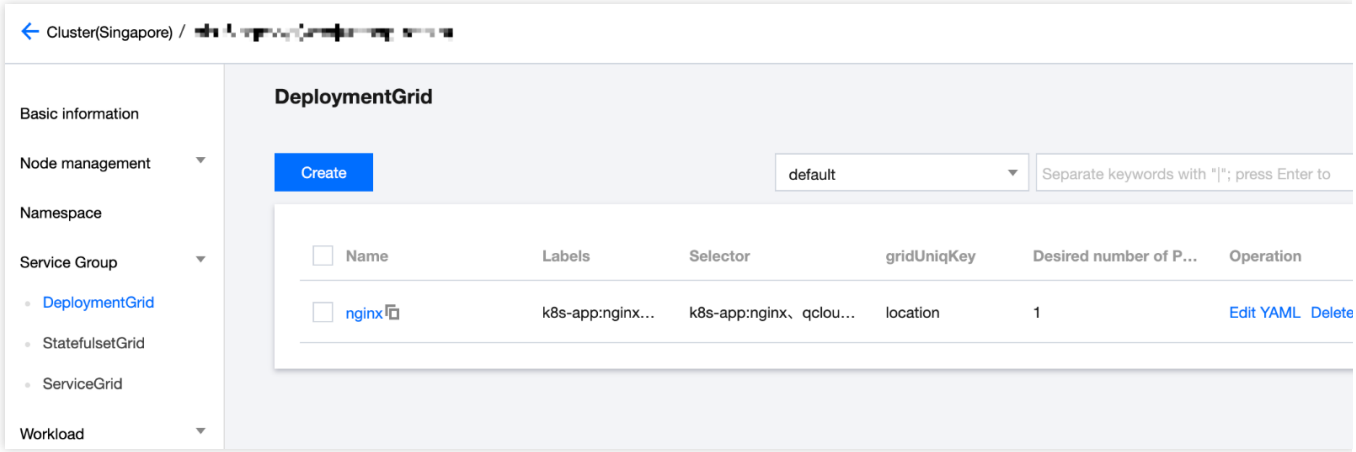
1. 选择 **ServiceGroup** > **DeploymentGrid**，进入列表页。
2. 单击**新建**，创建名称为 `Nginx` 的 DeploymentGrid。



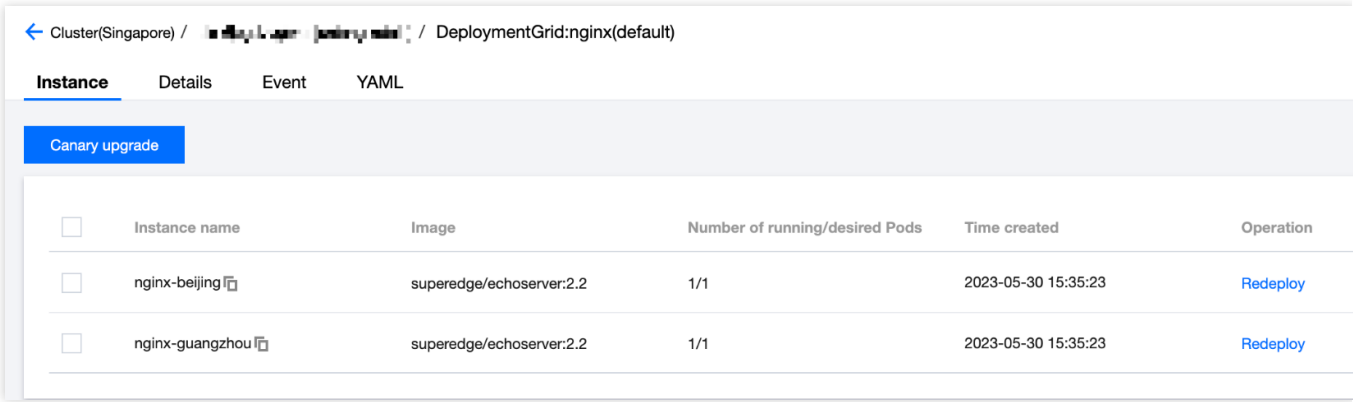
NodeGroup：这里选择需要批量部署 Nginx 服务的 NodeGroup 分组；这里选择 location，意味着将会在 `beijing` 和 `guangzhou` 两个 NodeUnit 下分别部署相应的 Deployment。

其余参数：其余参数和 TKE 部署应用的方式完全一致，这里不再详述；这里作为示例，实例数量设置为了3。

3. 单击**创建 DeploymentGrid**，等待部署完成。如下图所示：



4. 单击**nginx**链接，进入详情页，可以查看具体创建的**Deployment 详情**。如下图所示：



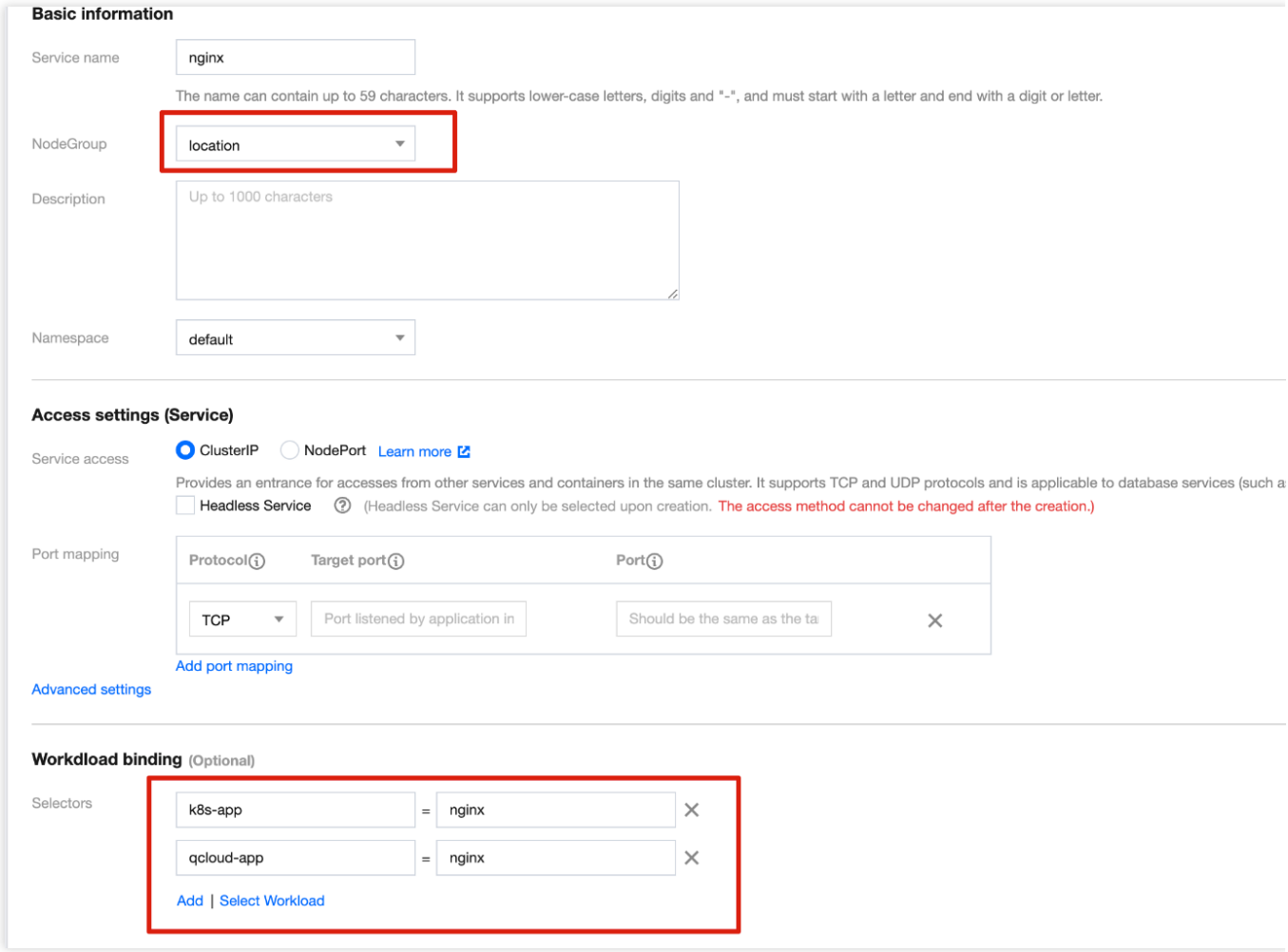
说明

平台在 NodeGroup 包含的每个 NodeUnit 下都分别创建了一个 Kubernetes 标准的 Deployment，名字为 **DeploymentGrid-NodeUnit**。

根据示例，此处示例名分别为 `nginx-beijing` 和 `nginx-guangzhou` 。

部署 ServiceGrid

1. 选择 **ServiceGroup** > **ServiceGrid**，单击**新建**，如下图所示：



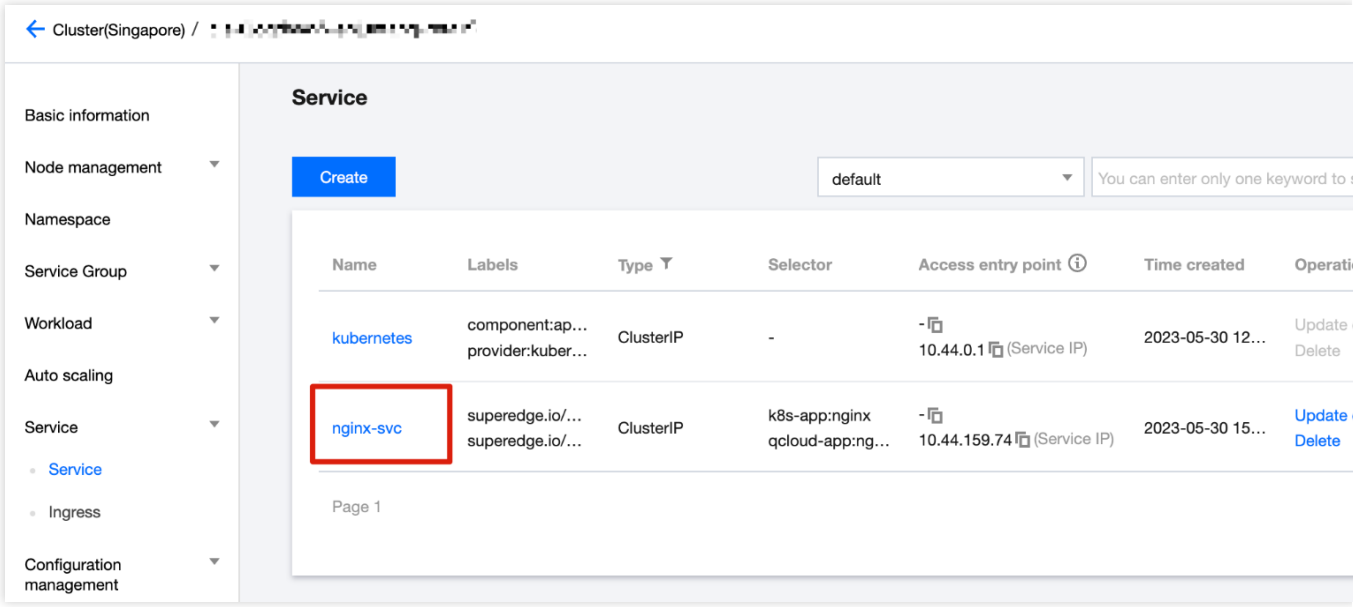
NodeGroup：选择需要的 NodeGroup，和上面的 DeploymentGrid 选择一样的 NodeGroup。

设置访问：标准 Kubernetes 的 Sevice 配置信息，选择需要的端口，这里 nginx 服务默认是 80。

Workload 绑定：选择 Service 通过 Seletor 选择需要的 Pod，可以手动输入添加；也可以选择 `引用 Workload` 添加已部署的 DeploymentGrid > nginx。

2. 单击**创建 ServiceGrid**，创建成功，显示事件详情页。

3. 在**服务** > **Service**中查看具体创建出来的 `Service` 信息，如下图所示：



4. 通过使用 `nginx-svc` 这个 Service，能够实现下面的目的：

从 `beijing` 地域的 Pod 中访问此 svc，后端只会访问到 `beijing` 地域的 3 个 pod 中。

从 `guangzhou` 地域的 Pod 中访问此 svc，后端只会访问到 `guangzhou` 地域的 3 个 pod 中。

每个地域访问这个Service 都会被限制在本 NodeUnit 范围内

健康状态检测

最近更新时间：2023-06-01 11:22:54

边缘场景下的弱网络会触发 Kubernetes 驱逐机制，引起不符合预期的 Pod 驱逐动作。边缘计算情景下，边缘节点与云端的网络环境十分复杂，网络质量无法保证，容易出现 API Server 和节点连接中断等问题。如果不加改造直接使用原生 Kubernetes，节点状态会经常出现异常，引起 Kubernetes 驱逐机制生效，导致 Pod 的驱逐和 EndPoint 的缺失，最终造成服务的中断和波动。
为解决这个问题，边缘容器服务 首创分布式节点状态判定机制。该机制可以更好地识别驱逐时机，保障系统在弱网络下正常运转，避免服务中断和波动。

需求痛点

原生 Kubernetes 处理方式

云边弱网络是影响了运行在边缘节点上的 kubelet 与云端 APIServer 之间通信，云端 APIServer 无法收到 kubelet 的心跳或者进行续租，无法准确获取该节点和节点上 Pod 的运行情况，如果持续时间超过设置的阈值， APIServer 会认为该节点不可用，并做出如下作：
失联的节点状态被置为 NotReady 或者 Unknown 状态，并被添加 NoSchedule 和 NoExecute 的 taints。
失联的节点上的 Pod 被驱逐，并在其他节点上进行重建。
失联的节点上的 Pod 从 Service 的 Endpoint 列表中移除。

解决方案

设计原则

在边缘计算场景中，仅依赖边缘端和 APIServer 的连接情况来判断节点是否正常并不合理，为了让系统更健壮，需要引入额外的判断机制。
相较于云端和边缘端，边缘端节点之间的网络更稳定，可利用更稳定的基础设施提高准确性。边缘容器服务首创了边缘健康分布式节点状态判定机制，除了考虑节点与 APIServer 的连接情况，还引入了边缘节点作为评估因子，以便对节点进行更全面的状态判断。经过测试及大量的实践证明，该机制在云边弱网络情况下提高了系统在节点状态判断上的准确性，为服务稳定运行保驾护航。该机制的主要原理如下：
每个节点定期探测其他节点健康状态
集群内所有节点定期投票决定各节点的状态
云端和边缘端节点共同决定节点状态
首先，节点内部之间进行探测和投票，共同决定具体某个节点是否存在状态异常，保证大多数节点的一致判断才能决定节点的具体状态。其次，即使节点之间的网络状态通常情况下优于云边网络，但也应该考虑边缘节点复杂的网

络情况，其网络并非100%可靠。因此，也不能完全信赖节点之间的网络，节点的状态不能只由节点自行决定，云边共同决定才更为可靠。基于这个考虑，做出如下设计：

节点内网判定	云端判定正常	云端判定异常
节点内网判定正常	最终判定正常	K8s 显示 NotReady，但是行为和标准 K8s 流程不同：不再调度新的 Pod 到该节点，但是并不驱逐 pod 以及处理 svc。
节点内网判定异常	最终判定正常	K8s 显示 NotReady，行为和标准 K8s 流程一致：驱逐存量 Pod；从 EndPoint 列表摘除；不再调度新的 Pod 到该节点。

前提条件

该功能需要打开节点的51005端口，以便节点之间进行分布式智能健康探测。

操作步骤

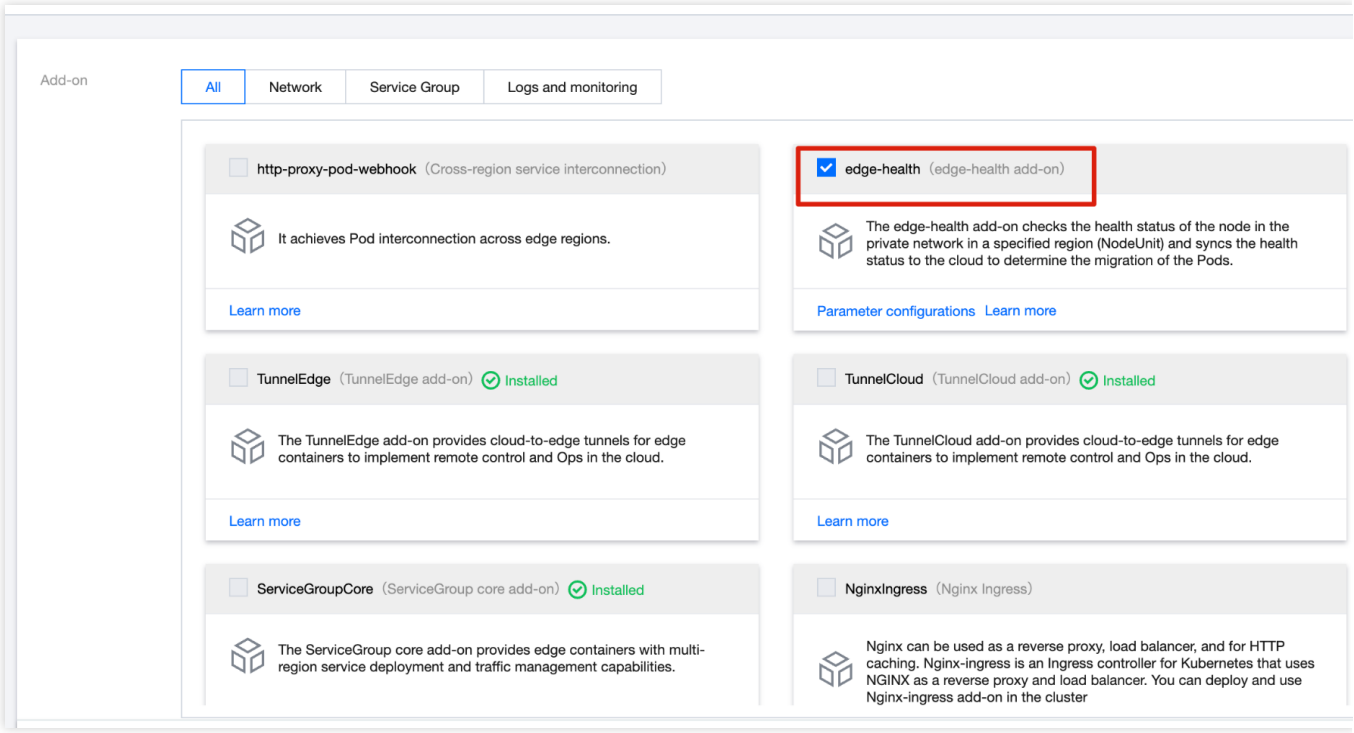
注意

边缘检查和多地域检查功能需要一定的部署和配置时间，并非即时生效。

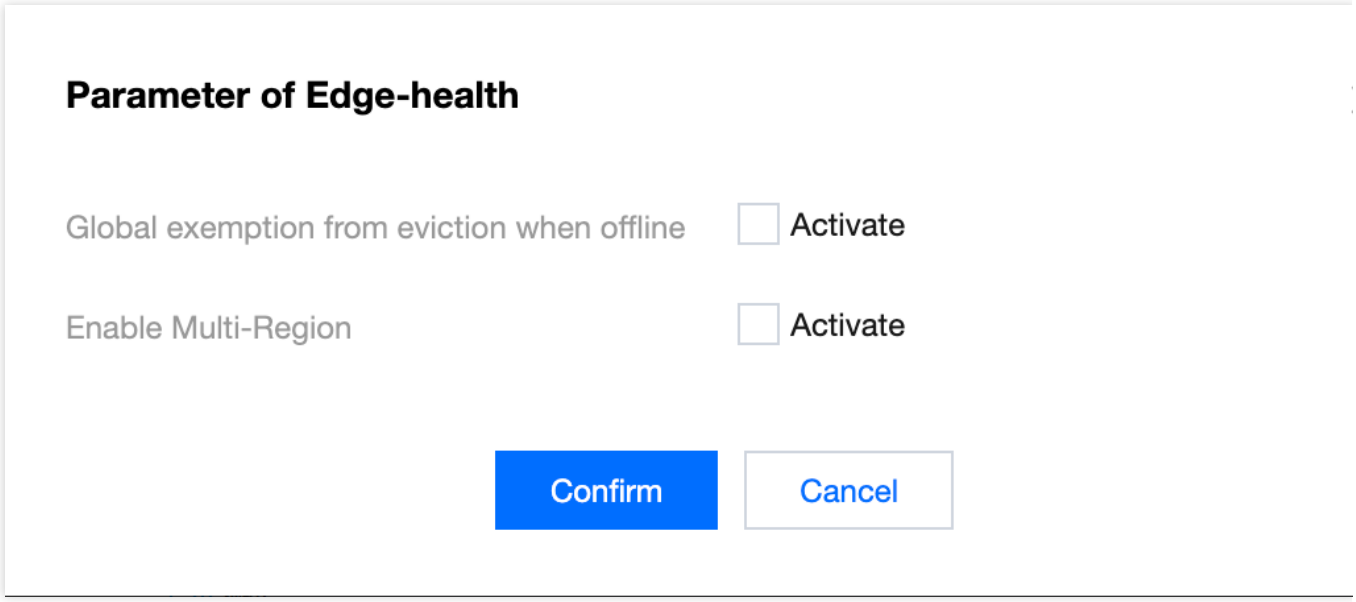
开启边缘检查功能

边缘检查功能默认关闭，请参考以下步骤手动开启：

1. 登录 容器服务控制台。
2. 在**集群列表**页面，选择目标边缘集群 ID，进入集群详情页面。
3. 选择左侧菜单栏中的**组件管理**，进入组件管理页面。
4. 单击**新建**，进入**新建组件**页面。
5. 勾选 **edge-health**，如下图所示：



在参数配置中您可以根据业务需求选择开启**全局节点离线免驱逐**和**使用多地域**。如下图所示：



6. 单击**完成**即可部署组件到目标边缘集群。

开启全局节点离线免驱逐

开启此功能后，会通过 webhook 机制拦截全部的节点驱逐标志即 **NoExecute** 的 taint，从而防止节点频繁断网重连导致的 Pod 驱逐和重新调度，此功能和多地域检查互斥，开启免驱逐则不会在边缘实际探测节点健康状态。

开启多地域检查功能

这里的多地域在 edge-health 中实际上是 NodeUnit 概念，通过 NodeUnit 才区分不同的节点地域属性，详情请参见 NodeUnit 文档。**开启 Edge Health** 能力会在所有边缘节点上部署 edge-health 健康监测 Pod。如果这个时候，不使能**开启多地域**，会默认互相检测所有节点的状态，此时并没有多地域概念，可以将所有节点看作同一地域下的节点进行互相检测。

如果想要在不同的地域内部独立进行节点间的健康状态互检，则需要使能**开启多地域**，此时会修改一个 `edge-health-config` 的 ConfigMap，通知 K8s 要处理多地域的健康检测能力。开启后，节点会根据节点上的 `<nodeunit-name>: nodeunits.superedge.io` 标签来区分地域。例如，`beijing: nodeunits.superedge.io` 表明将节点划分到 NodeUnit **beijng**。标签取值相同的节点视为同一个 NodeUnit。开启多地域功能时，同一个地域内的节点会相互探测和投票。

注意

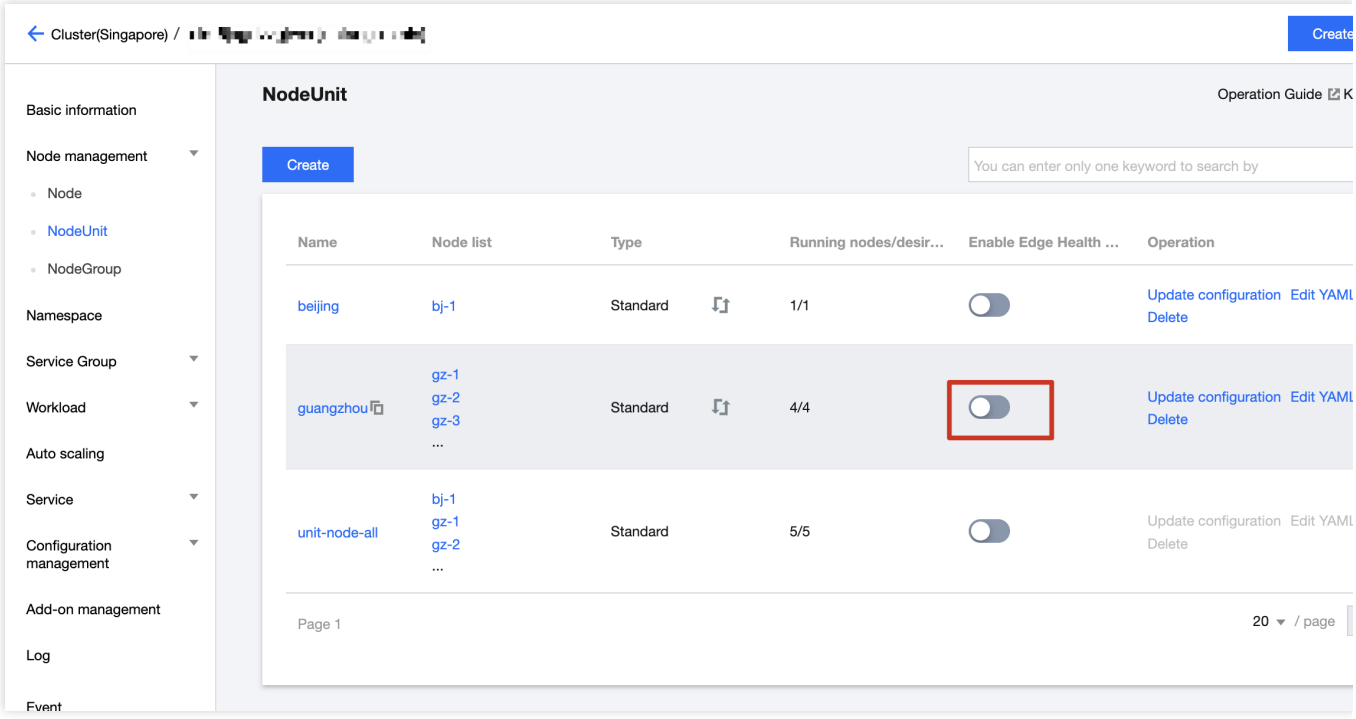
如果使能**开启多地域**能力后，节点不属于任何一个 NodeUnit，该节点不会进行健康检查。

如果没有使能**开启多地域**能力，则一个集群内的所有节点会进行相互检查，即使节点属于 NodeUnit。

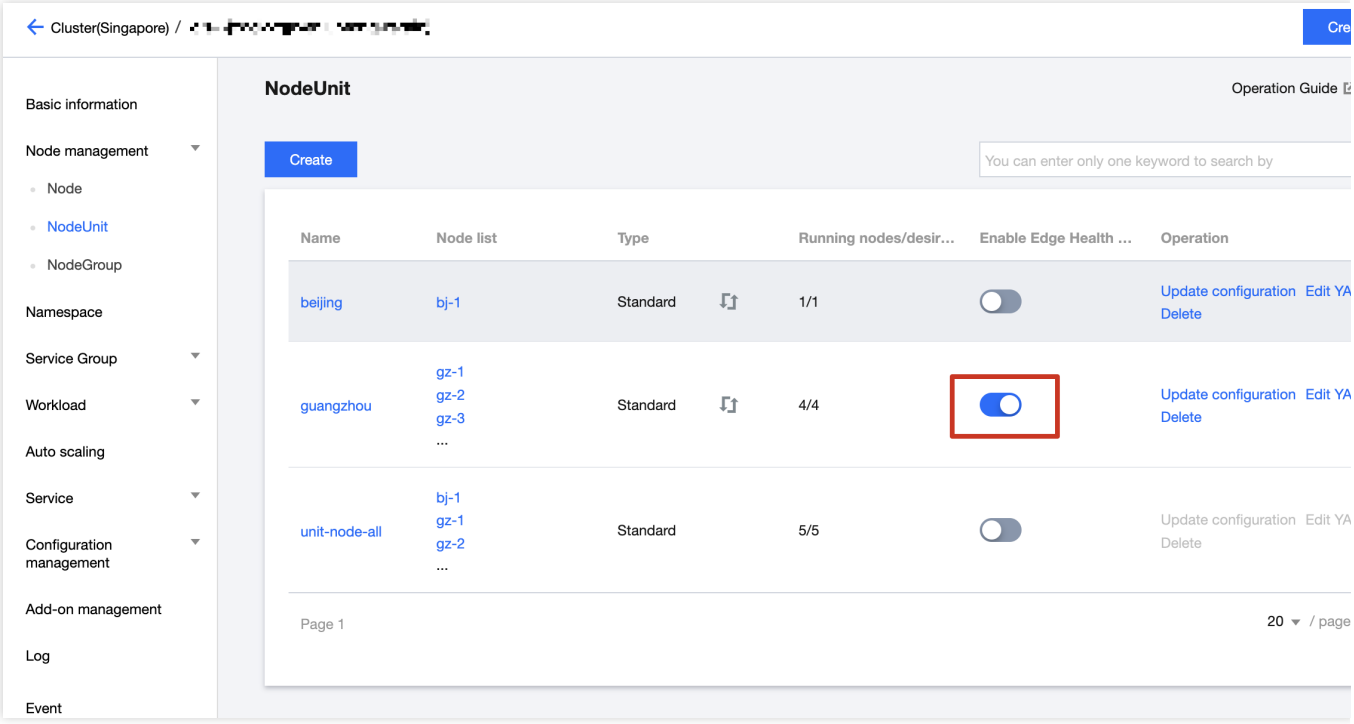
在 NodeUnit 上开启健康状态检测

下面给出一个示例，说明**健康状态检测**的实际效果：

1. 登录 容器服务控制台
2. 在集群管理页，单击需要远程登录节点的集群 ID，进入该集群详情页。
3. 选择**"节点管理">"边缘节点池"**, 可以看到开关"使能健康状态检查"

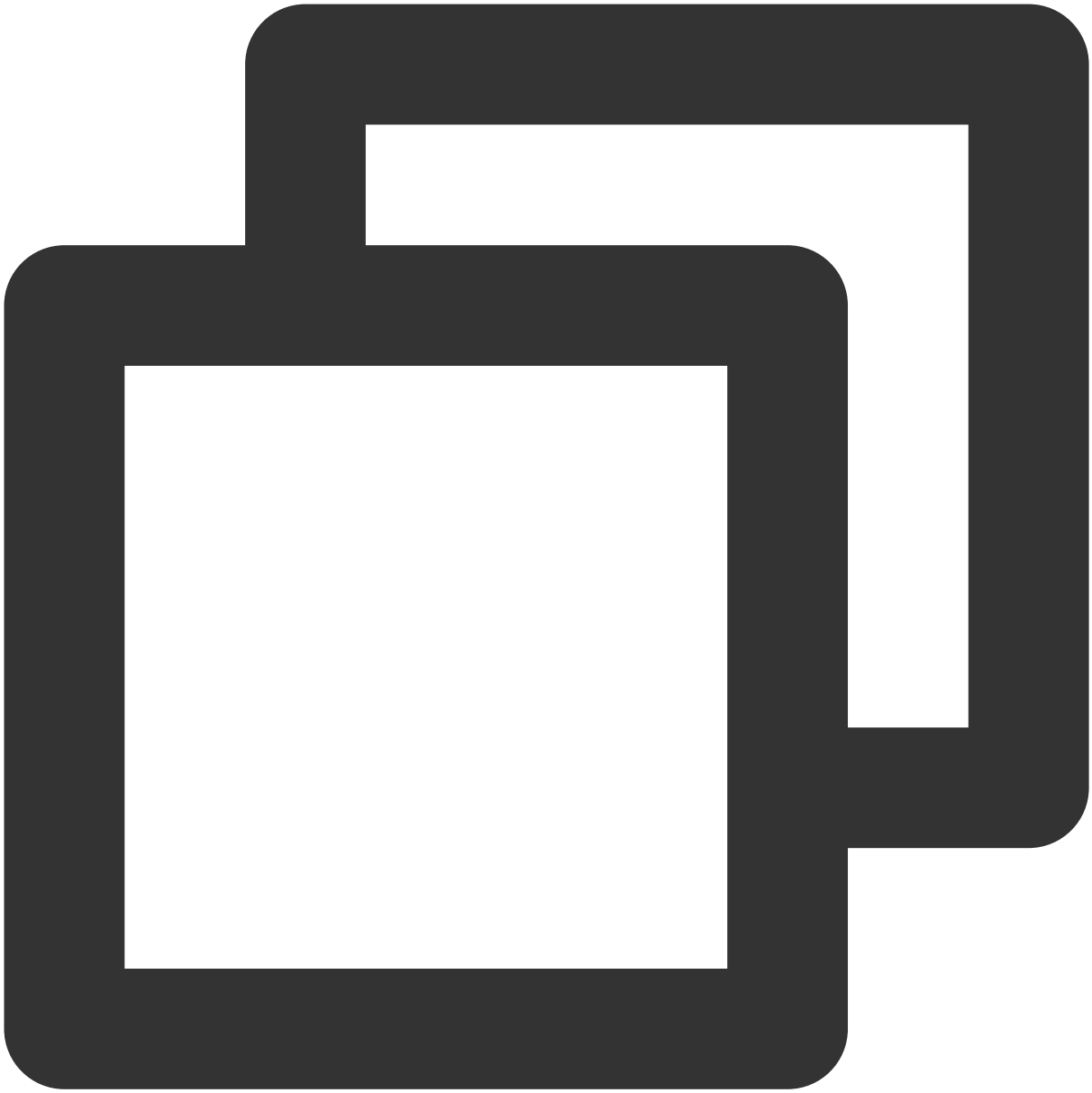


4. 您可以选择所需的 NodeUnit 并启用“边缘健康检查”开关。这时，您可以在每个节点的边缘健康状态中查看，NodeUnit 下的所有节点将相互进行探测。



5. 这个时候就完整使能了**多地域健康状态**能力，使用 `gz-3` 节点模拟测试如下：

可以使用 `iptables` 规则模拟和 apiserver 断连：



```
iptables -I INPUT -s xxx.xxx.xxx.xxx -j DROP  # 这里为云端 apiserver 的 IP 地址
```

过一段时间后，观察 `gz-3` 节点为 **NotReady** 状态，但是 `Taint` 并没有添加 `node.kubernetes.io/unreachable:NoExecute` ，所以此节点上的 Pod 并不会被 evict，此节点已经不会被调度新的 Pod 了。.

```
Every 2.0s: kubectl get nodes

NAME   STATUS     ROLES    AGE     VERSION
bj-1   Ready      <none>   3h59m   v1.22.5-tke.1.9+5f76a1720f9839
gz-1   Ready      <none>   7h57m   v1.22.5-tke.1.9+5f76a1720f9839
gz-2   Ready      <none>   7h57m   v1.22.5-tke.1.9+5f76a1720f9839
gz-3   NotReady   <none>   4h3m    v1.22.5-tke.1.9+5f76a1720f9839
gz-4   Ready      <none>   4h3m    v1.22.5-tke.1.9+5f76a1720f9839
```

```
Name:               gz-3
Roles:              <none>
Labels:             beta.kubernetes.io/arch=amd64
                    beta.kubernetes.io/os=linux
                    guangzhou=nodeunits.superedge.io
                    kubernetes.io/arch=amd64
                    kubernetes.io/hostname=gz-3
                    kubernetes.io/os=linux
                    location=guangzhou
                    unit-node-all=nodeunits.superedge.io
Annotations:        flannel.alpha.coreos.com/backend-data: {"VtepMAC":"2a:44:f6:f2:ce
                    flannel.alpha.coreos.com/backend-type: vxlan
                    flannel.alpha.coreos.com/kube-subnet-manager: true
                    flannel.alpha.coreos.com/public-ip: 172.16.16.106
                    node.alpha.kubernetes.io/ttl: 0
                    superedge.io/node-quota: false
                    volumes.kubernetes.io/controller-managed-attach-detach: true
CreationTimestamp:  Tue, 30 May 2023 16:04:42 +0800
Taints:             node.kubernetes.io/unreachable:NoSchedule
Unschedulable:      false
```

关闭 `gz-3` 节点，模拟云端和内网检测均失败的场景，观察节点状态，这个时候为 Kubernetes 标准的节点失效状态，会迁移节点上的 Pod。

```
Name:               gz-3
Roles:              <none>
Labels:             beta.kubernetes.io/arch=amd64
                    beta.kubernetes.io/os=linux
                    guagnzhou=nodeunits.superedge.io
                    kubernetes.io/arch=amd64
                    kubernetes.io/hostname=gz-3
                    kubernetes.io/os=linux
                    location=guagnzhou
                    superedgehealth/topology-zone=guangzhou
                    unit-node-all=nodeunits.superedge.io
Annotations:        flannel.alpha.coreos.com/backend-data: {"VtepMAC":"92:57:d3:57:97:ce"}
                    flannel.alpha.coreos.com/backend-type: vxlan
                    flannel.alpha.coreos.com/kube-subnet-manager: true
                    flannel.alpha.coreos.com/public-ip: 192.168.122.23
                    node.alpha.kubernetes.io/ttl: 0
                    nodeunhealth: yes
                    superedge.io/node-quota: false
                    volumes.kubernetes.io/controller-managed-attach-detach: true
CreationTimestamp:  Wed, 16 Nov 2022 15:20:28 +0800
Taints:             node.kubernetes.io/unreachable:NoExecute
                    node.kubernetes.io/unreachable:NoSchedule
```

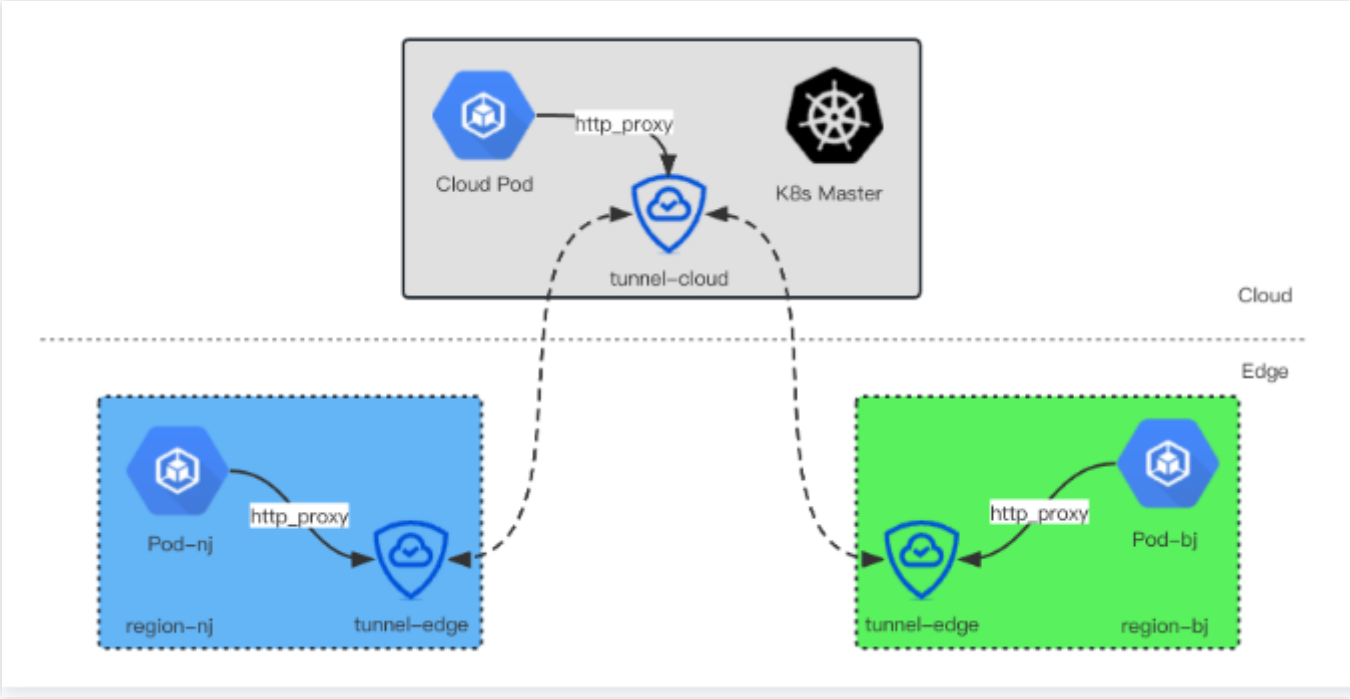
跨地域服务互访

最近更新时间：2023-06-01 11:22:54

操作场景

本文介绍如何通过云边 Tunnel 隧道，实现不同边缘地域的 Pod/Svc 互访（现在暂时仅支持7层 http/https 协议）。例如在南京地域的 Pod 可以成功访问北京地域的 Nginx 服务，反之亦然。

架构原理



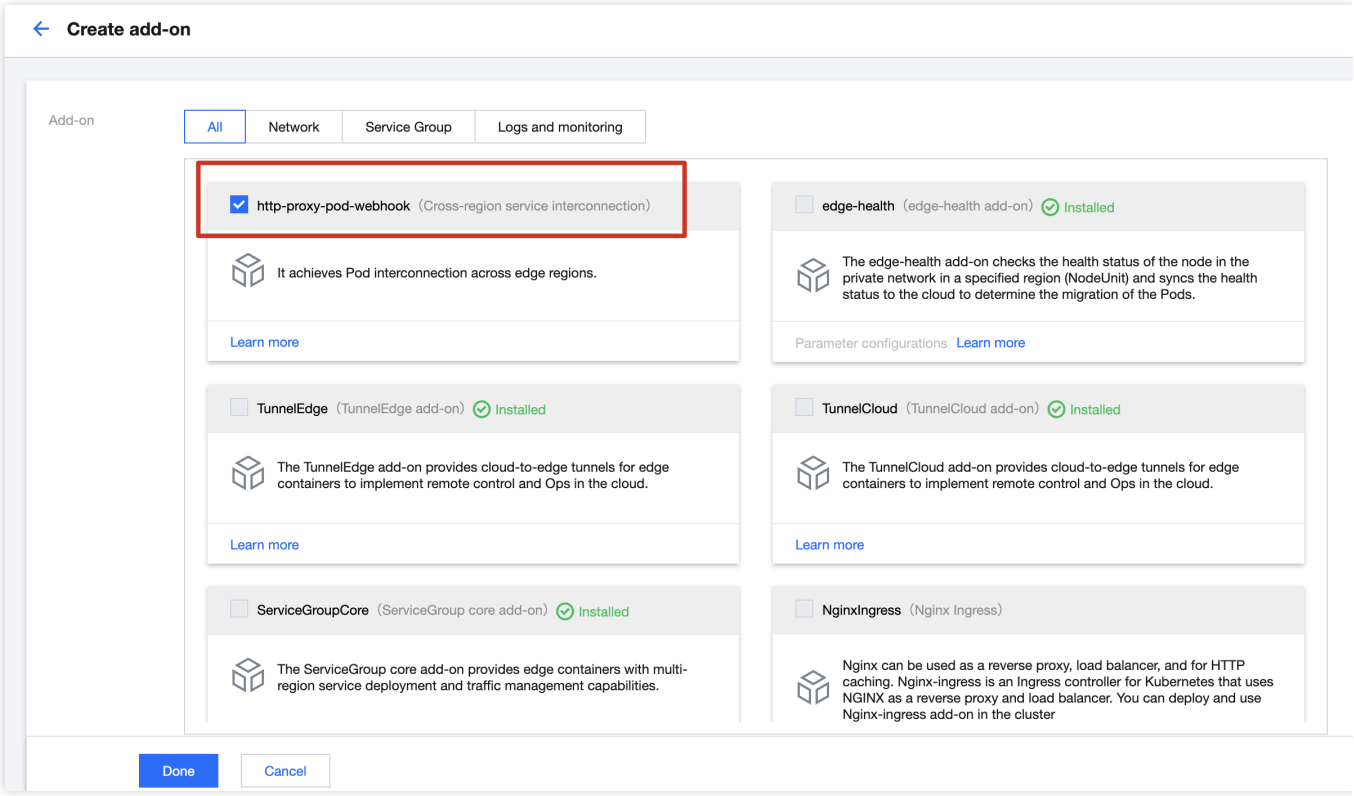
基本原理如上图所示，边缘节点 tunnel-edge 会和云端 tunnel-cloud 建立双向隧道，tunnel-edge 会在本地开放 `:8080` 端口作为 http/https 代理端口，如果边缘 Pod 需要通过代理访问其他地域的 Pod ip 或者 svc 的时候，可以在 Pod 内指定 `http_proxy` 或者 `https_proxy` 环境变量，把 http/https 流量通过 tunnel 转发到云端或者其他地域的边缘节点上。

说明

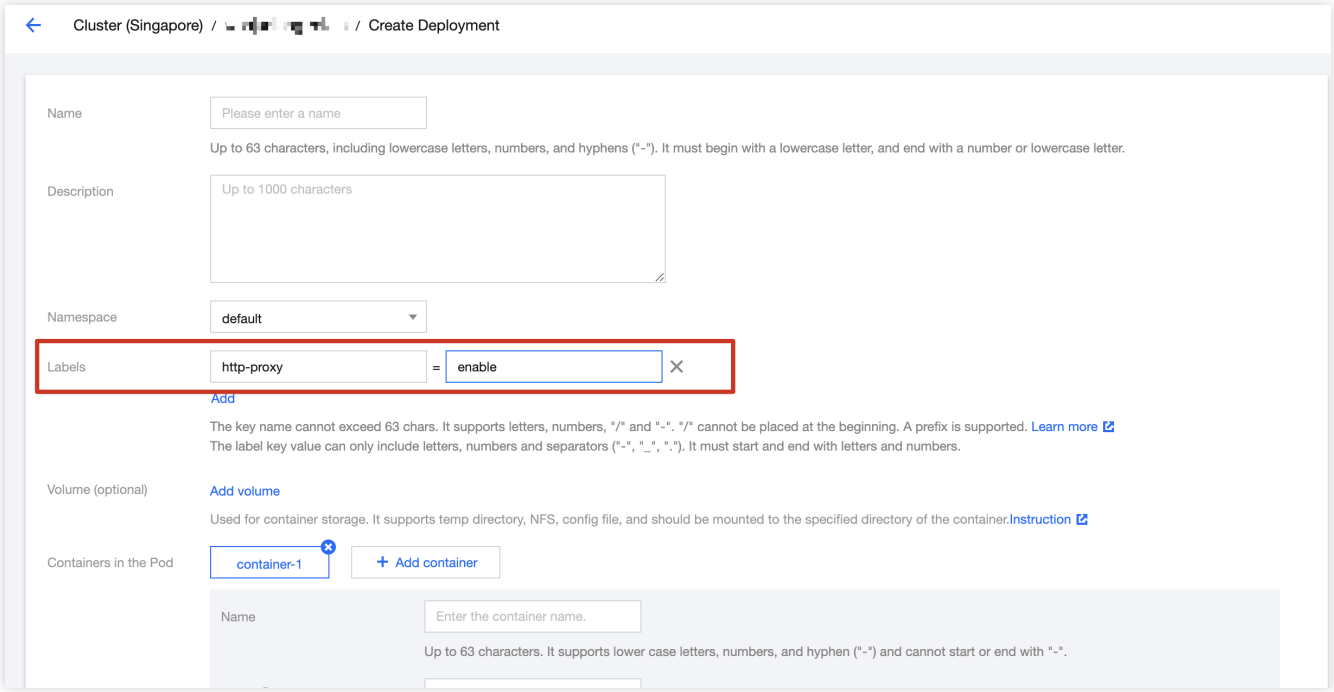
在产品层面，平台通过 webhook 的方式，方便您指定代理：您可以设置 workload 中的 label `http-proxy=enable` 来开启代理模式，此时 workload 相应的 Pod 会自动注入相应环境变量，例如 `http_proxy=169.254.20.11:8080` 。

操作步骤

1. 登录 容器服务控制台，选择左侧导航栏中的**集群**。
2. 在集群管理页，单击需要远程登录节点的集群 ID，进入该集群详情页。
3. 选择页面左侧**组件管理**
4. 单击**新建**，进入 Add-on 创建页面
5. 在“新建组件”页中，选择 **http-proxy-pod-webhook（跨地域服务互访）**，点击确定。如下图所示：



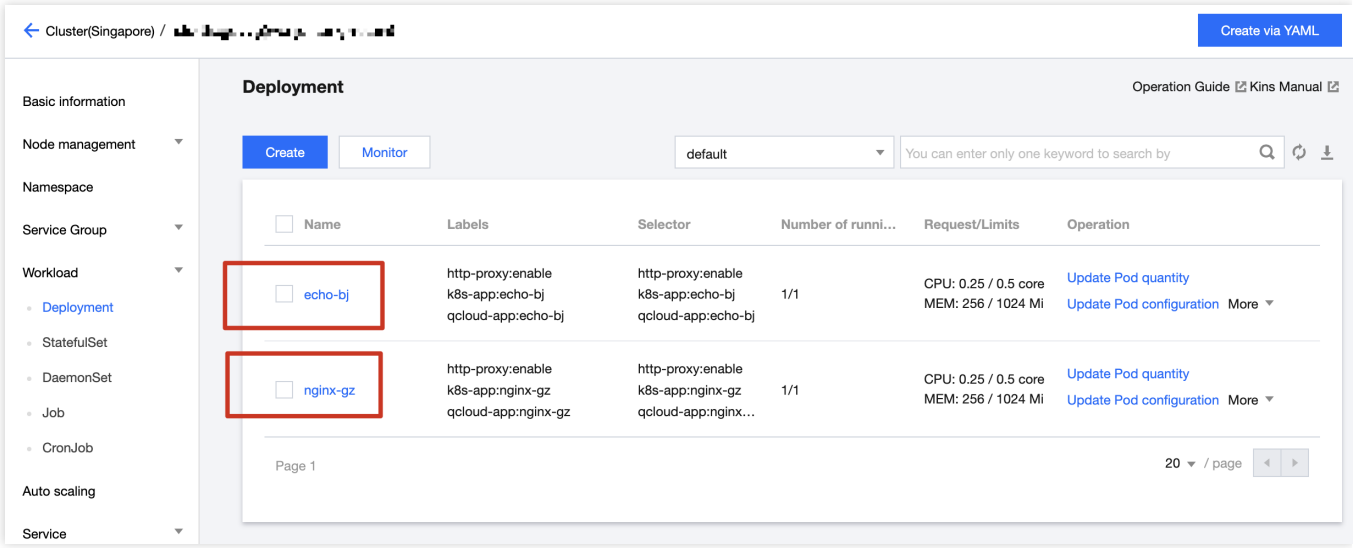
6. 为了验证跨地域互访特性，在两个不同地域分别创建 deployment，同时添加 label 如下：



将 "http-proxy=enable" 标签添加到deployment 中，就会通过第 5 步的 webhook 在 Pod 环境中注入 http_proxy，如下所示：

```
[root@bj-1 ~]# crictl exec -it d9ba3b2aa2d7a /bin/sh
/ # env
KUBERNETES_PORT=tcp://10.44.0.1:443
KUBERNETES_SERVICE_PORT=443
HOSTNAME=echo-bj-55c4bc47bf-8sq5p
SHLVL=1
HOME=/root
ECHO_BJ_SERVICE_HOST=10.44.224.177
ECHO_BJ_PORT_8080_TCP_ADDR=10.44.224.177
ECHO_BJ_PORT_8080_TCP_PORT=8080
ECHO_BJ_PORT_8080_TCP_PROTO=tcp
http_proxy=http://169.254.20.11:8080
TERM=xterm
ECHO_BJ_SERVICE_PORT=8080
ECHO_BJ_PORT=tcp://10.44.224.177:8080
KUBERNETES_PORT_443_TCP_ADDR=10.44.0.1
NGINX_VERSION=1.15.3
ECHO_BJ_SERVICE_PORT_8080_8080_TCP_3T8IH1V68HM=8080
PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
KUBERNETES_PORT_443_TCP_PORT=443
ECHO_BJ_PORT_8080_TCP=tcp://10.44.224.177:8080
KUBERNETES_PORT_443_TCP_PROTO=tcp
KUBERNETES_PORT_443_TCP=tcp://10.44.0.1:443
KUBERNETES_SERVICE_PORT_HTTPS=443
PWD=/
KUBERNETES_SERVICE_HOST=10.44.0.1
```

当两个不同地域的 deployment 创建成功后，可以检查状态：



注意：

每个部署必须使用调度器将 pod 分配到特定节点。例如，echo-bj 被分配到 "bj-1"，而 nginx-gz 被分配到 "gz-2"。

7. 登录到 "bj-1" 节点并访问 "gz-2" 的 pod，如下所示

```
NAME                          READY   STATUS    RESTARTS   AGE     IP            NODE   NOMINATED NODE   READ
echo-bj-55c4bc47bf-8sq5p      1/1     Running   0          9m18s   10.33.2.3     bj-1   <none>           <non
nginx-gz-d94bf4fc7-c492d      1/1     Running   0          7m49s   10.33.0.72    gz-2   <none>           <non
```

当访问 gz-2 的 pod 时，您会看到请求被重定向到 http_proxy "169.254.20.11:8080"。

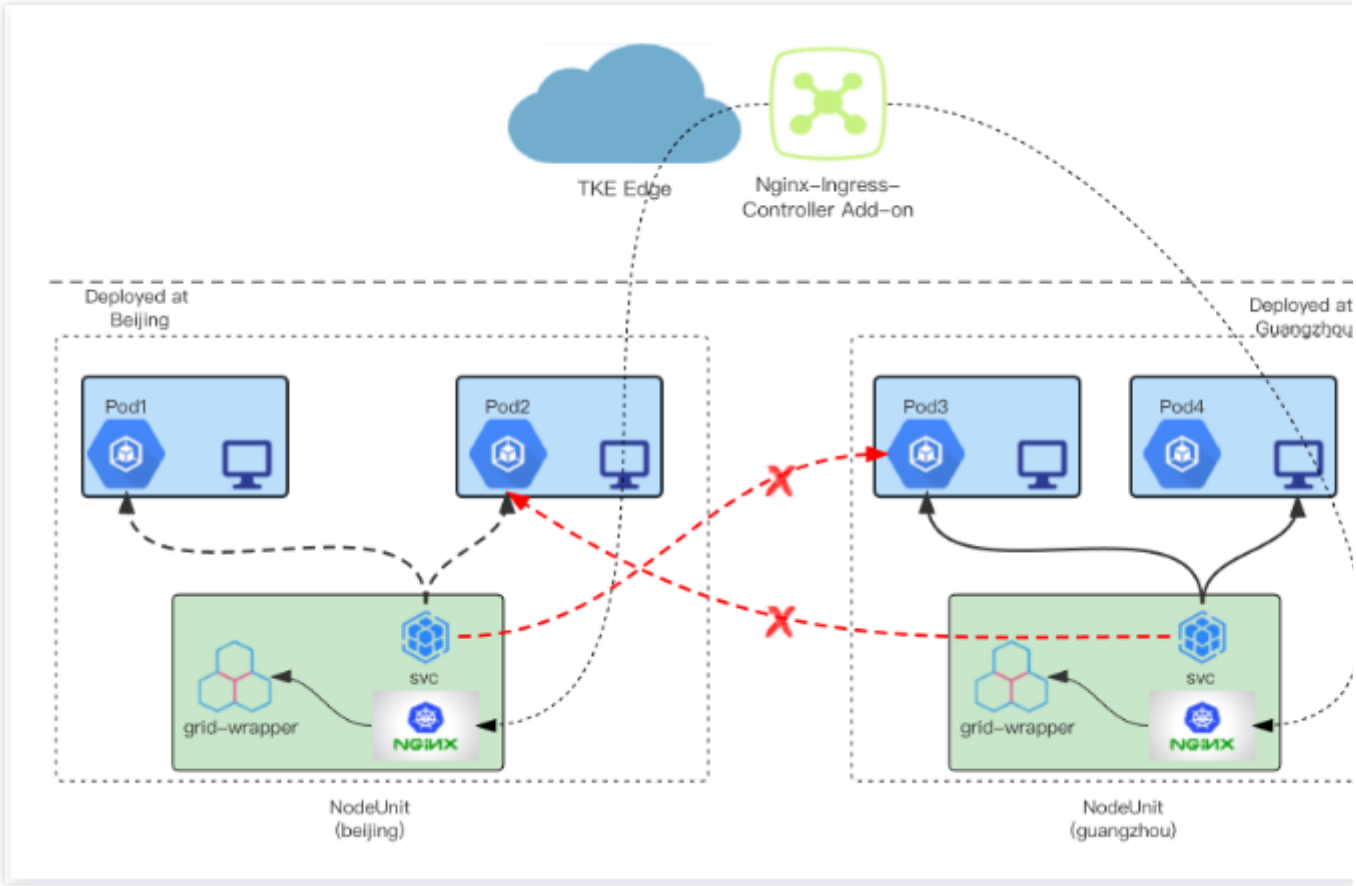
```
[root@bj-1 ~]# crictl exec -it d9ba3b2aa2d7a /bin/sh
/ # wget -O- http://10.33.0.72:8080
Connecting to 169.254.20.11:8080 (169.254.20.11:8080)


Hostname: nginx-gz-d94bf4fc7-c492d

Pod Information:
        -no pod information available-

Server values:
        server_version=nginx: 1.12.2 - lua: 10010

Request Information:
        client_address=10.33.0.65
        method=GET
        real path=/
        query=
        request_version=1.1
        request_scheme=http
        request_uri=http://10.33.0.72:8080/

Request Headers:
        connection=close
        host=10.33.0.72:8080
        user-agent=Wget

Request Body:
        -no body in request-

-                   100% |*****************************************
/ #
```

多地域 Ingress

最近更新时间：2023-06-01 11:22:54

场景介绍

在边缘节点区分多个地域的情况下，每个地域都有独立的网络架构，需要在本地域对外提供 Ingress 服务能力，腾讯云边缘容器服务基于 `NodeUnit` 的相关概念，在产品上支持在不同地域创建 `Nginx-Ingress-Controller` 的能力，同时通过 `application-grid-wrapper` 组件，可以将 Ingress-Controller 对 pod 的访问限制在本地域(NodeUnit 范围)之内，具体架构如下图：

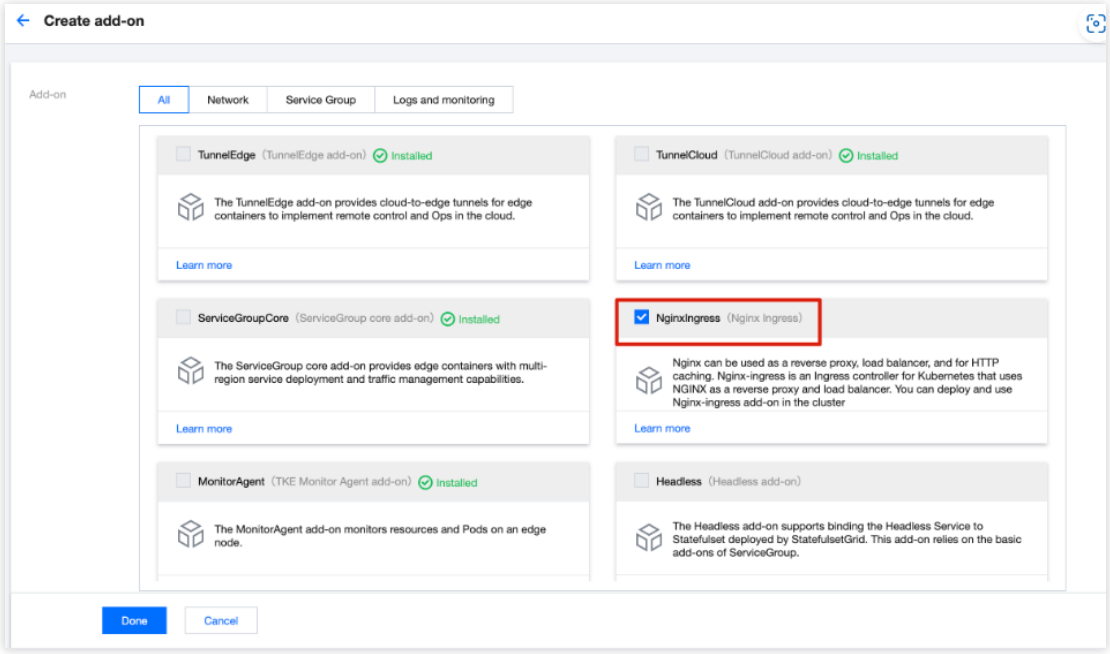


操作步骤

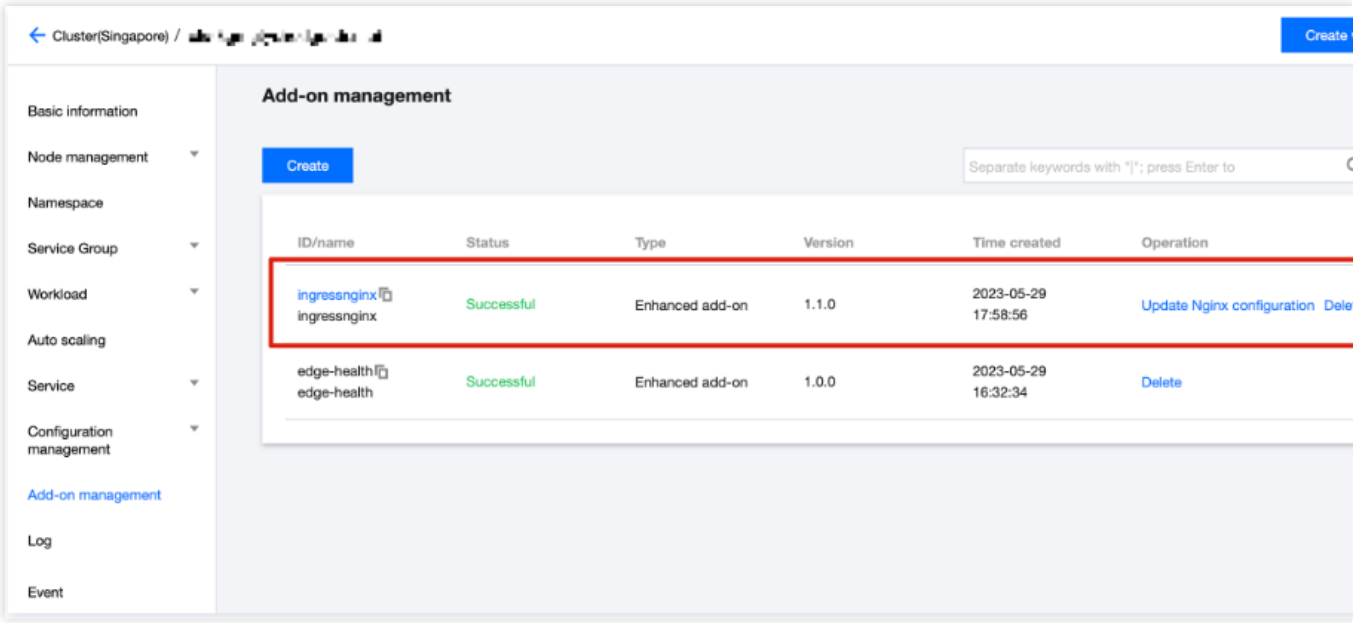
Nginx-Ingress-Controller 组件安装

1. 登录 腾讯云容器服务控制台。

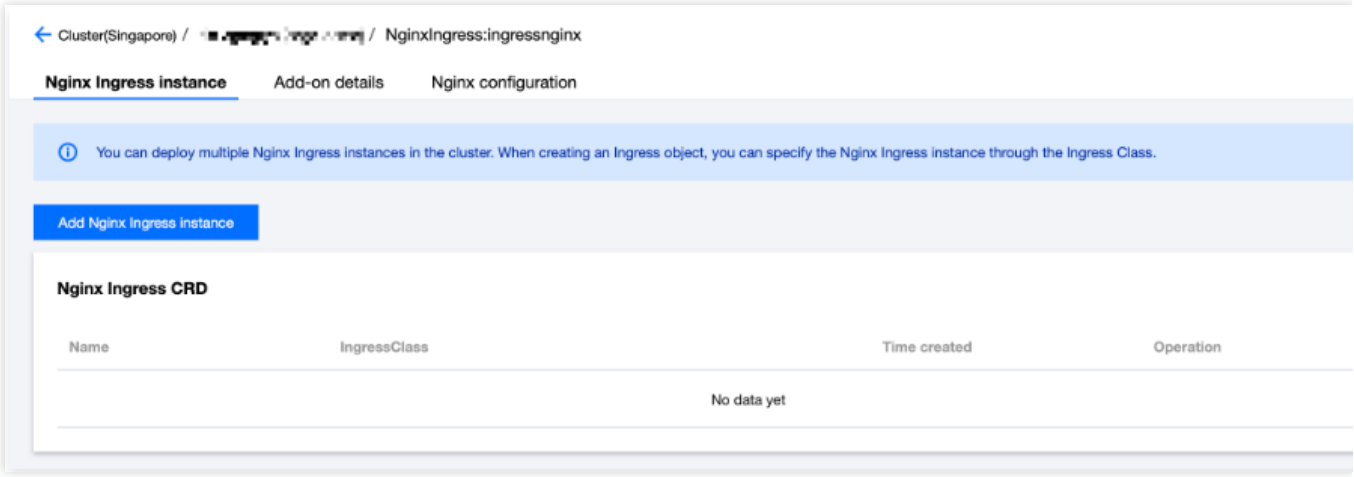
2. 单击目标边缘集群 ID，进入集群详情页。
3. 选择页面左侧**组件管理**，进入组件列表页面。
4. 点击**新建**，进入**创建组件**页面。
5. 选择 **NginxIngress** ，然后单击**完成**，如下图所示：



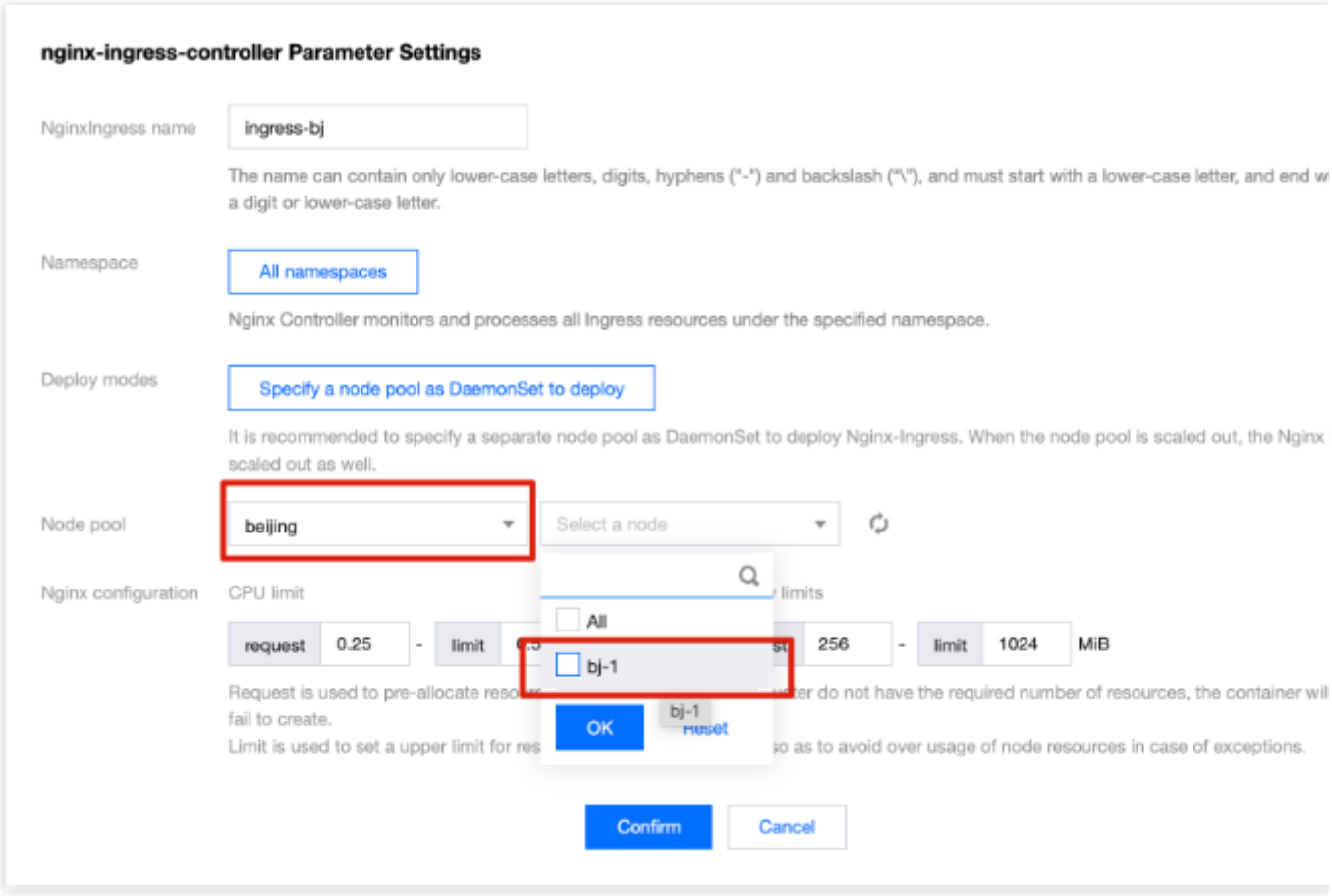
6. 等待边缘组件安装完成。如下图所示：



7. 单击已经部署的组件名称，进入组件详情页面。如下图所示：



8. 单击**新增Nginx Ingress 实例**，在指定的地域（NodeUnit）创建 Nginx-Ingress-Controller 实例。如下图所示：



NginxIngress 名称：指定相应部署的 Ingress-Ingress-Controller 实例的名称，会创建相应的 daemonset。

命名空间：现在默认监听所有命名空间下的 Ingress 资源。

部署选项：现在默认使用 DaemonSet 模式部署，需要用户自行选择 NodeUnit 下的一个或者多个节点提供 Ingress Controller 服务。

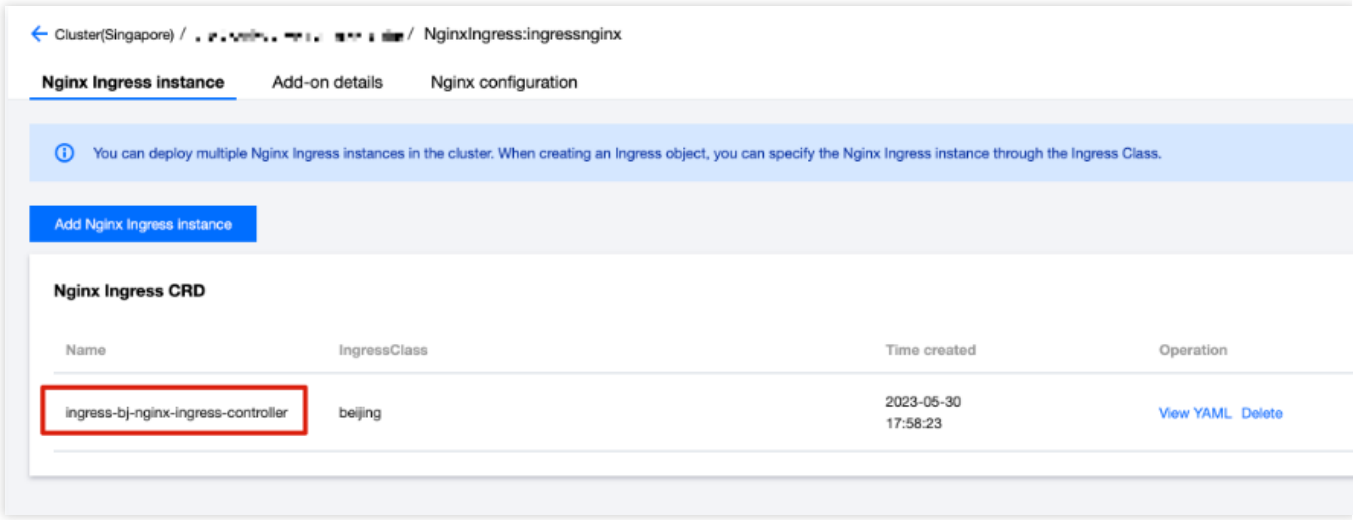
节点池：选择您需要部署的节点池（NodeUnit）进行部署，然后选择需要的节点来部署 Nginx-Ingress-Controller 服务。

注意：

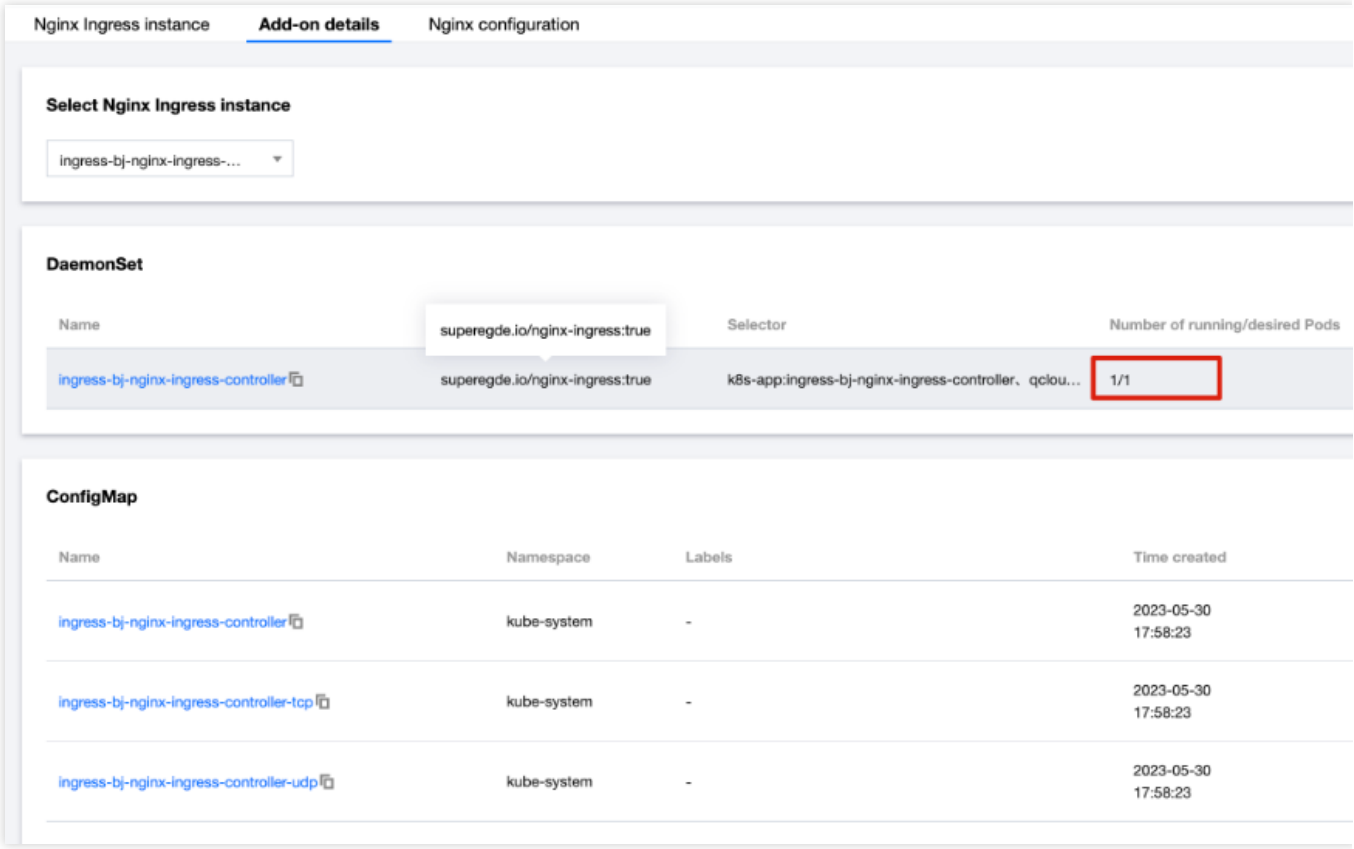
选择的节点需要保证 80/443端口未被占用，否则会导致 Nginx-Ingress-Controller 启动失败。

Nginx 配置：配置 Nginx-Ingress-Controller 的资源占用，请根据您业务压力需求合理配置。

9. 单击**完成**后，即可查看创建的 Ingress-Controller 实例。如下图所示：



10. 查看**组件详情**，可以确认当前controller 的部署状态，当运行的 Pod 数达到期望后，就表示已经部署成功。如下图所示：



部署服务并使用 Ingress 访问

1. 这里以 Ningx 服务为例，创建一个 Deployment，这些 Pod 分别会部署到 beijing地域和 guangzhou 地域。如下图所示：

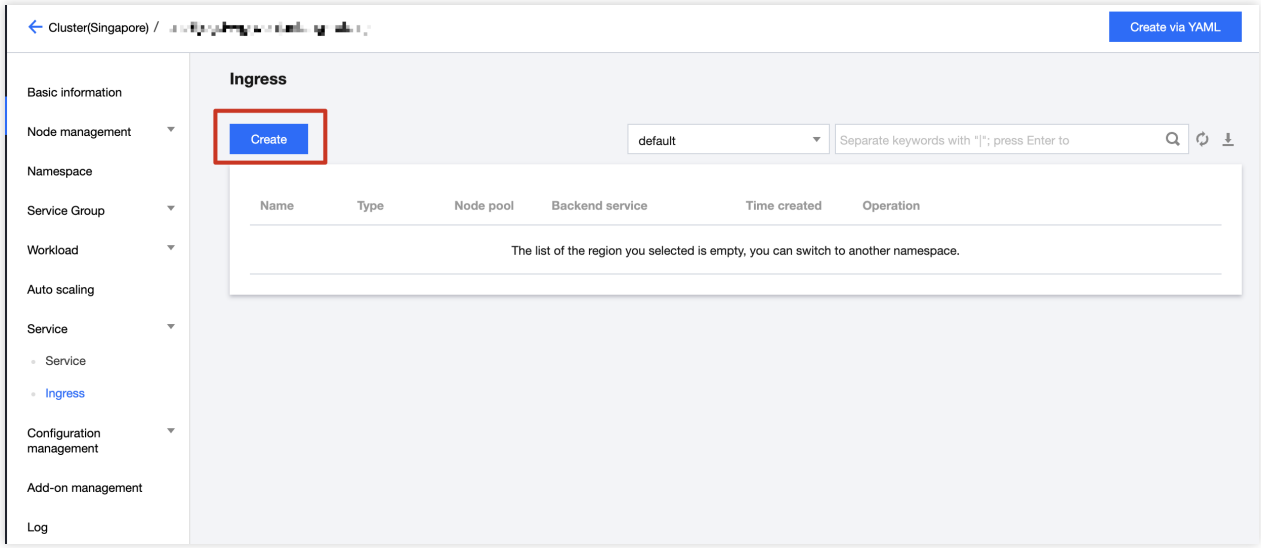
```
nginx-beijing-76cdbbc899-k66c8     1/1     Running   0          2m11s   10.33.2.4    bj-1   <none>
nginx-guangzhou-75f4c49d47-tzkgg   1/1     Running   0          2m11s   10.33.0.6    gz-1   <none>
```

然后使用同一个 svc 提供内部访问服务：

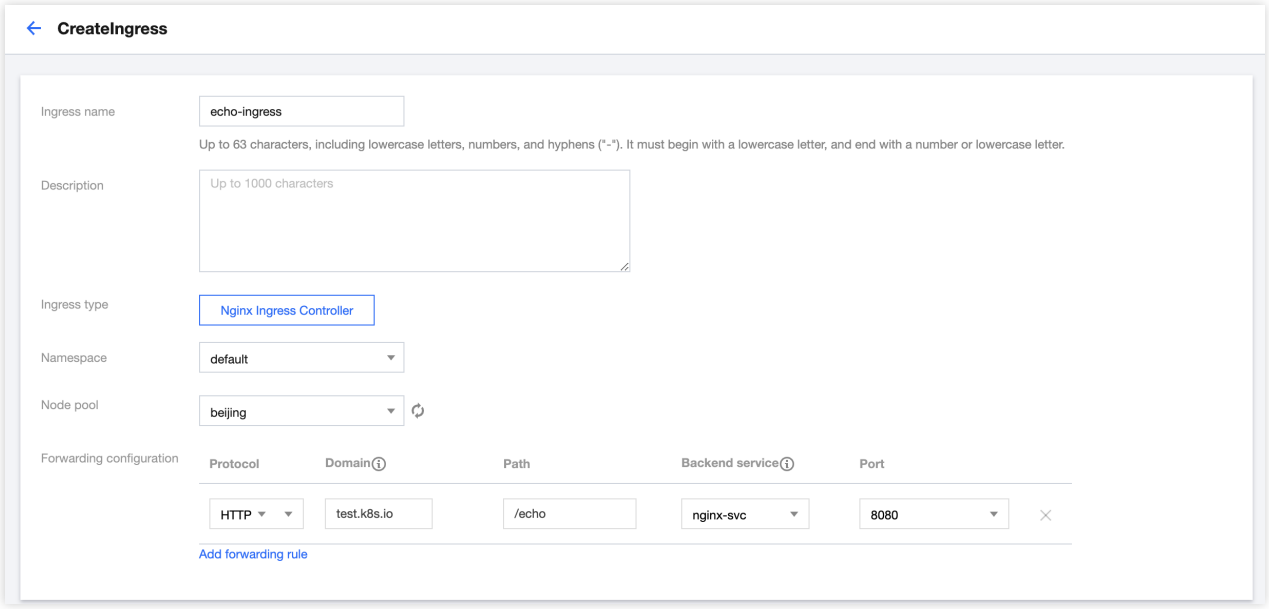
```
Name:                     nginx-svc
Namespace:                default
Labels:                   superedge.io/grid-selector=nginx
                          superedge.io/grid-uniq-key=location
Annotations:              topologyKeys: ["location"]
Selector:                 k8s-app=nginx,qcloud-app=nginx
Type:                     ClusterIP
IP Family Policy:         SingleStack
IP Families:              IPv4
IP:                       10.44.6.125
IPs:                      10.44.6.125
Port:                     tcp-8080-8080  8080/TCP
TargetPort:               8080/TCP
Endpoints:                10.33.0.6:8080,10.33.2.4:8080
Session Affinity:         None
Events:                   <none>
```

这里我们期望从 beijing 地域访问 Ingress 的时候，只会访问到beijing 下的 Pod，即 `nginx-deployment-7c97977fd8-2zv87`，而不会访问到 guangzhou 地域的 pod。

2. 创建 Ingress，进入**集群详情页->服务->Ingress**，单击**新建**。如下图所示：



3. 输入所需的 Ingress 信息。如下图所示：



Ingress 名称：输入需要创建的 Ingress 的名称。

描述：备注描述信息。

Ingress类型：当前默认只支持 `Nginx Ingress Controller` 类型。

命名空间：指定命名空间创建此 Ingress。

节点池：选择此 Ingress 需要绑定的 Ingress-Controller，也就是上面部署的 Nginx-Ingress-Controller 实例。

转发配置：这里按照服务的具体配置输入。例如这里输入自定义域名 `test.k8s.io`，路径为 `echo`，后端服务指定为上面创建的 svc- `nginx` 。

4. 在相应的地域内访问 ingress 服务。例如在beijing地域内，访问 `http://test.k8s.io/echo`，如下图所示：

```
[root@bj-1 ~]# curl http://test.k8s.io/echo


Hostname: nginx-beijing-76cdbbc899-k66c8

Pod Information:
        -no pod information available-

Server values:
        server_version=nginx: 1.12.2 - lua: 10010

Request Information:
        client_address=10.33.2.1
        method=GET
        real path=/
        query=
        request_version=1.1
        request_scheme=http
        request_uri=http://test.k8s.io:8080/

Request Headers:
        accept=*/*
        host=test.k8s.io
        user-agent=curl/7.61.1
        x-forwarded-for=192.168.20.234
        x-forwarded-host=test.k8s.io
        x-forwarded-port=80
        x-forwarded-proto=http
        x-forwarded-scheme=http
        x-real-ip=192.168.20.234
        x-request-id=7fb10970f188c46323106db51c31d5d6
        x-scheme=http

Request Body:
        -no body in request-
```

可以进行多次访问测试，会发现所有的访问都会被限制在 beijing 地域的 Pod 中，不会访问到 guangzhou 地域中。

Kins 边缘独立集群使用指南

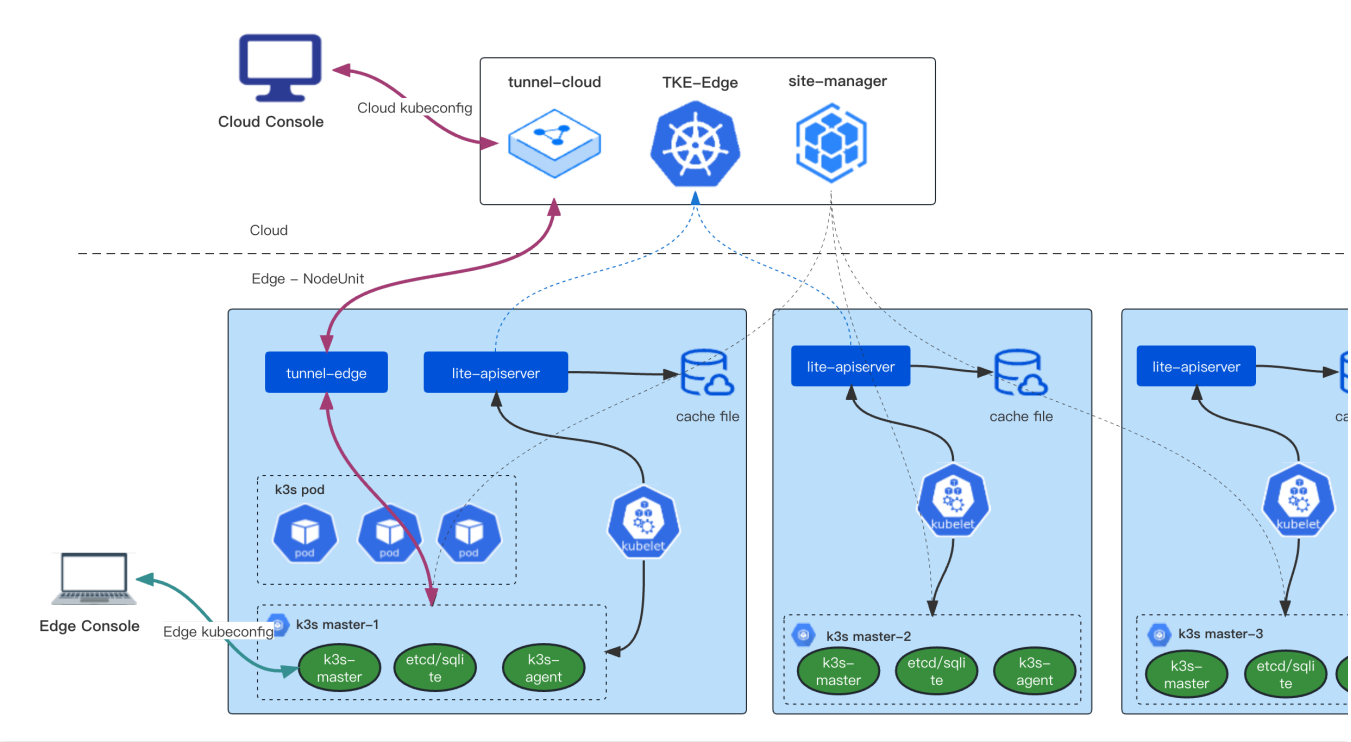
最近更新时间：2023-06-01 11:23:35

操作场景

本文介绍 TKE-Edge 推出的新特性——Kins，其主要功能描述如下：此能力可以将标准的 NodeUnit 边缘节点池，一键提升为独立 K3s 集群，此边缘节点池即可和云端控制面断网，并长期离线运行使用。离线过程中这个边缘节点池可以作为一个标准的K3s 集群进行独立运维；后期如果有运维或者升级的需求，可以将此节点池重新和云端建立连接后，即可从云端实现云上同步升级等远程运维操作。此功能全面改善之前 NodeUnit 断网后无法独立自治的能力缺陷。

架构原理

Kins 的基本原理如下图所示：



边缘侧的三个节点可以划分为一个 NodeUnit。通过边缘节点池相关操作，可以在这三个节点上拉起 K3s 的 master 和 agent 组件，从而组成独立的单 master 或者3 master 的 K3s 集群。这个边缘 K3s 集群可以从云端通过 tunnel 访问，同时也可以直接在边缘侧节点上访问。

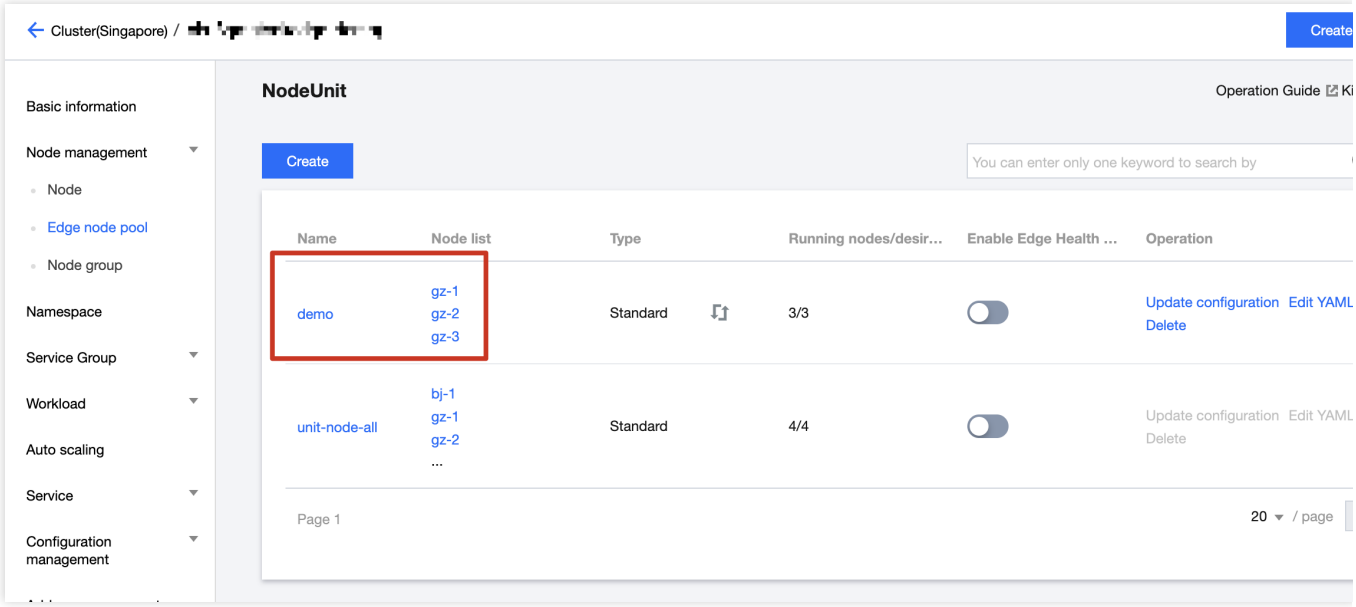
操作步骤

注意：

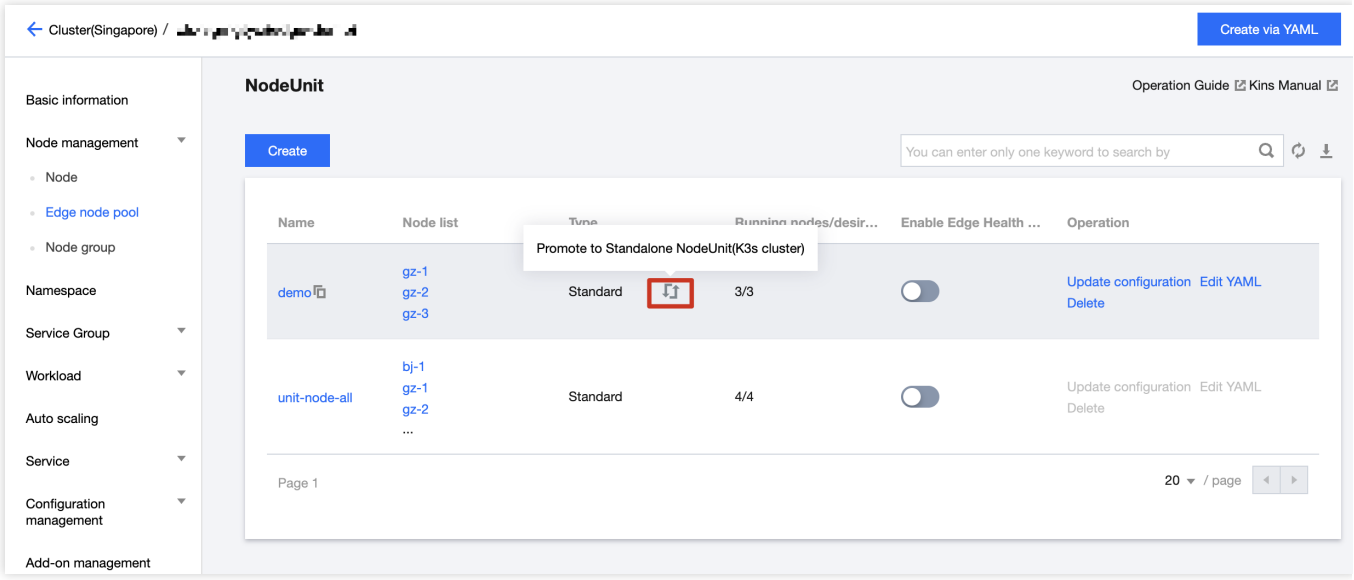
Kins 能力现在暂时只支持 Kubernete 1.22版本，同时运行时必须为 containerd。

创建边缘 K3s 集群

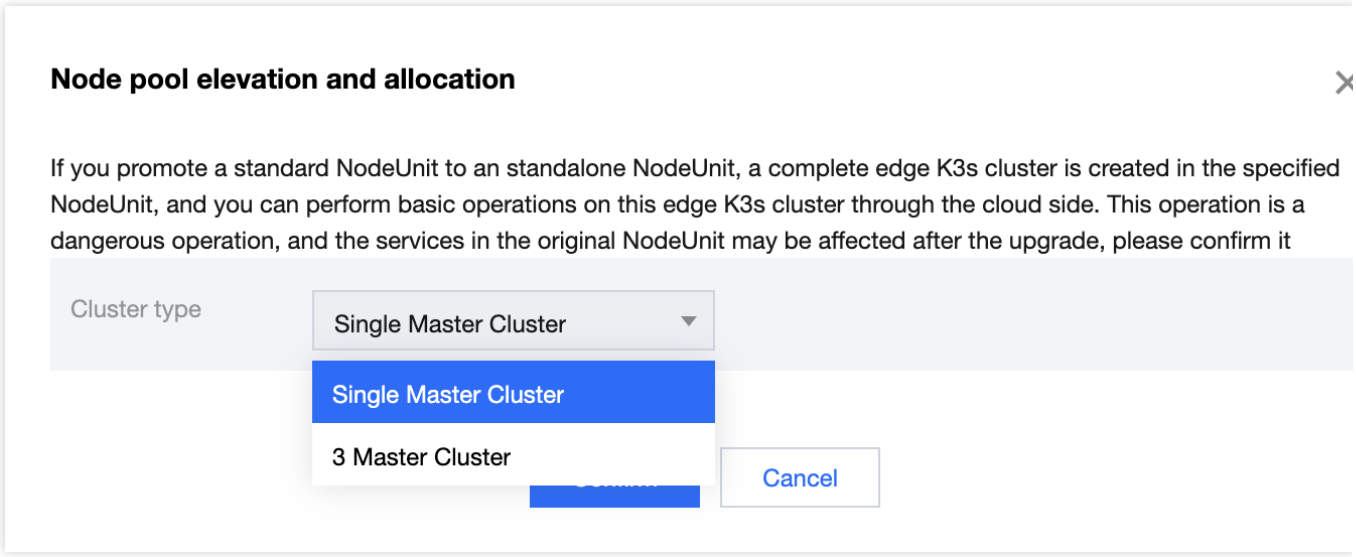
1. 登录 容器服务控制台，选择左侧导航栏中的**集群**。
2. 在集群管理页，选择需要操作的边缘集群，进入该集群详情页。
3. 用户可以选择合适的方式将所有节点加入集群。
4. 选择**节点管理 > 边缘节点池**，单击**新建**，创建普通的边缘节点池 **demo**，并按照标准方式在节点池中添加需要的节点，如下图所示：



5. 选择需要提升为边缘 K3s 独立集群的节点池，本文示例为边缘节点池 **demo**，单击**升级成独立节点池**，如下图所示：

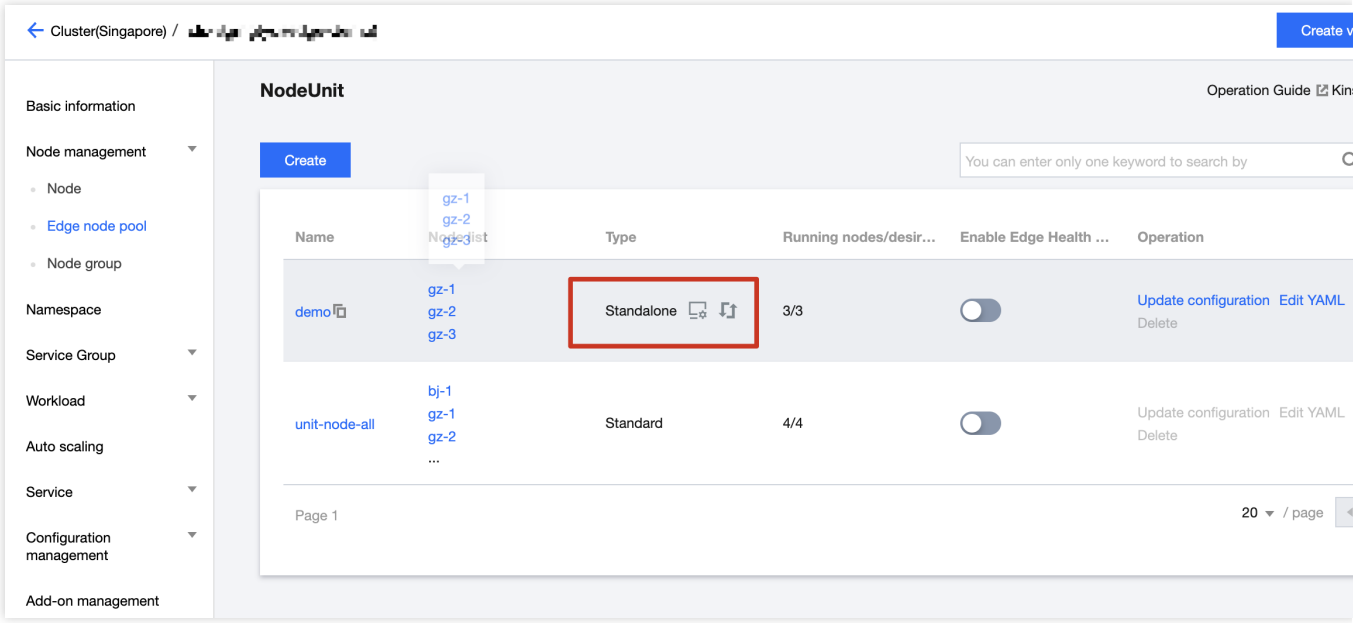


6. 在**节点池升降配**中，选择 K3s 集群模式，如下图所示：



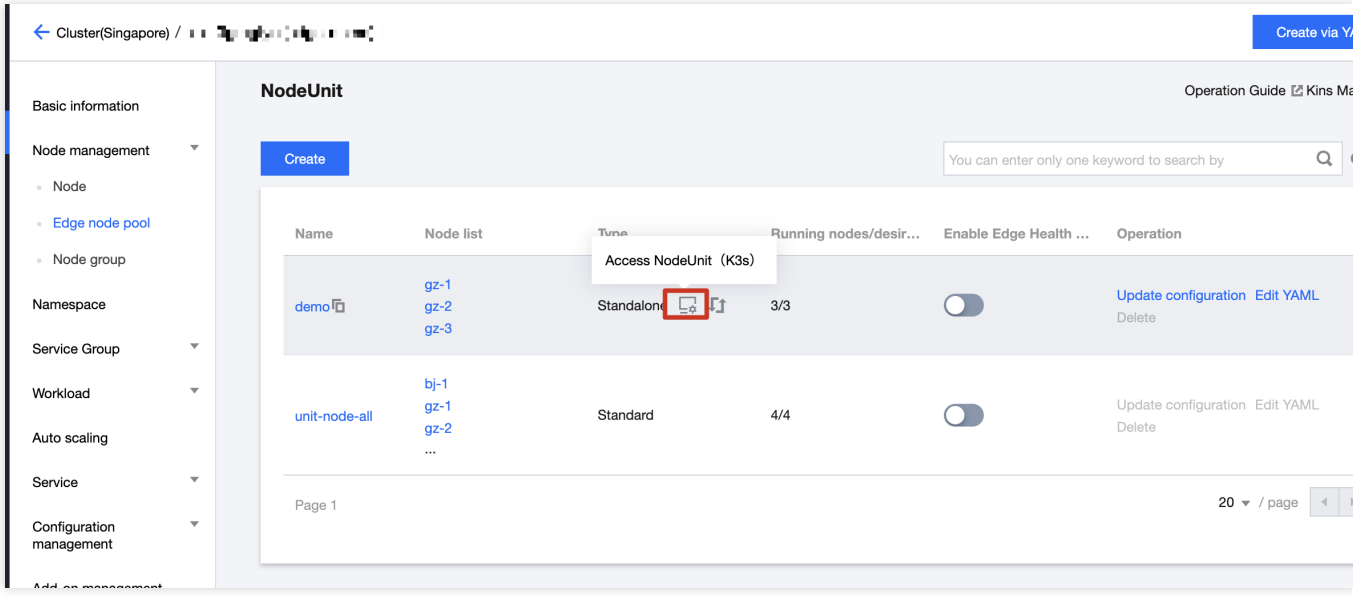
如果 NodeUnit 中只有 1-2 个节点，默认只能使用“单 Master 集群”；如果 NodeUnit 包含 ≥ 3个节点，您可以选择使用“单 Master 集群”或者“3 Master 高可用集群”。您可以根据自己的需用选择使用。

7. 选择后，单击**确定**即可。返回 NodeUnit 列表页后，稍等一段时间，可以观察到，当前 demo 的类型已经变为“独立”，如下图所示：



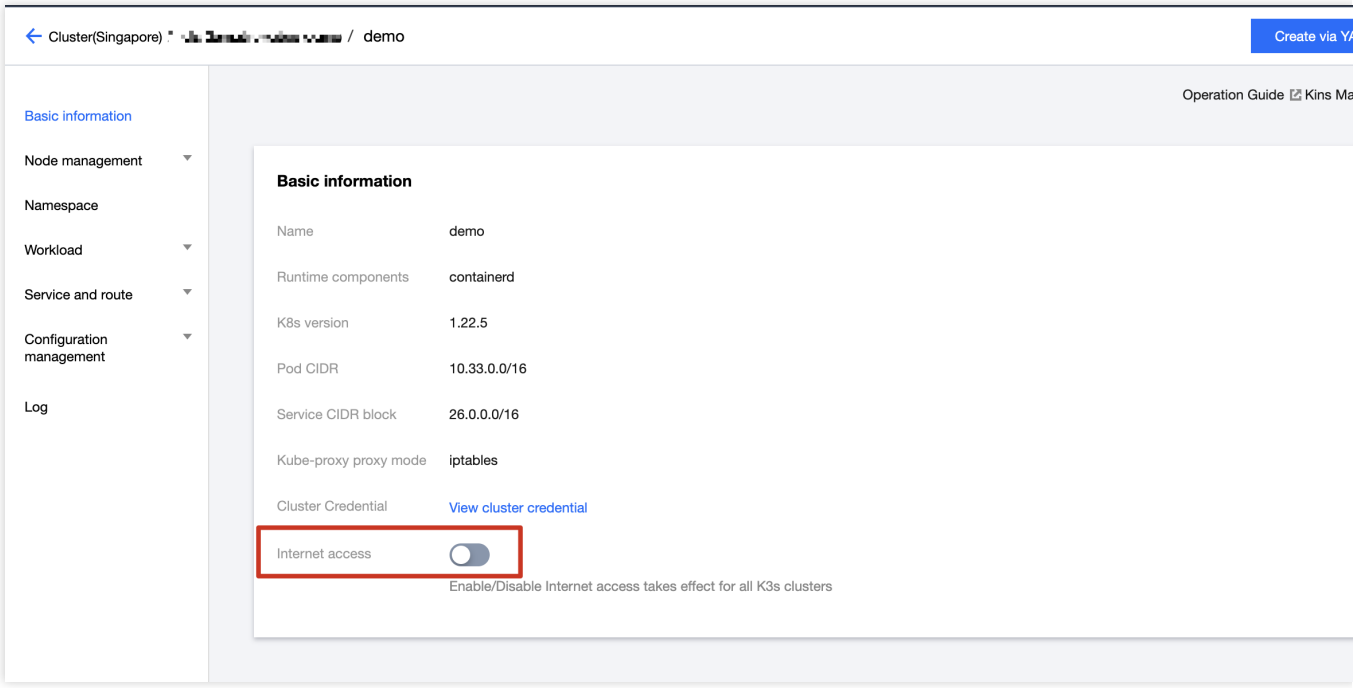
访问边缘 K3s 集群

1. 在边缘节点池列表页，单击**访问边缘节点池**，即可跳转到边缘 K3s 集群的控制页面，如下图所示：

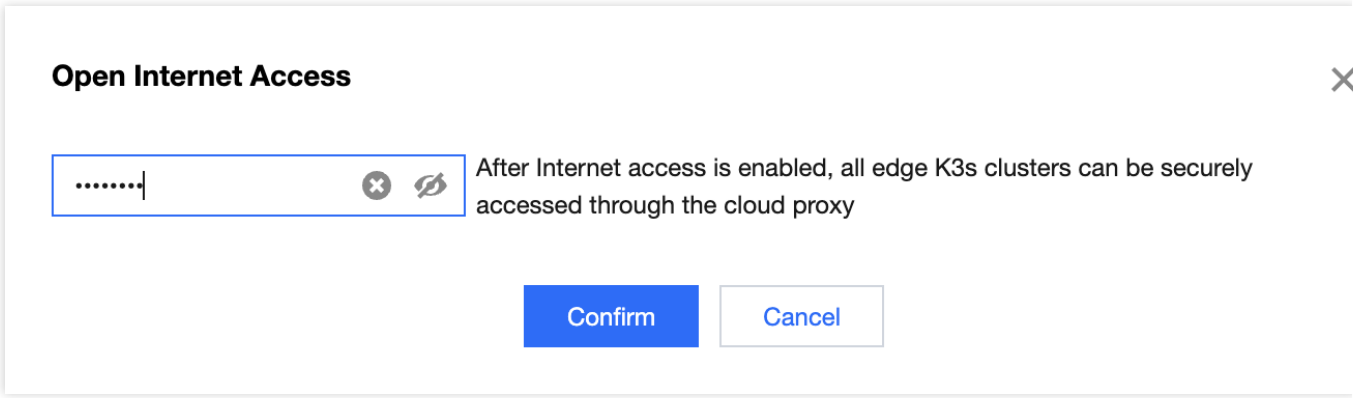


2. 在边缘 K3s 集群的**基本信息**中，单击

开启边缘 K3s 集群的**外网访问**能力。开启后，您可以在公有云上任意客户端使用 kubectl 命令，通过 TKE-Edge 的外网代理能力，访问到边缘 K3s 集群。如下图所示：



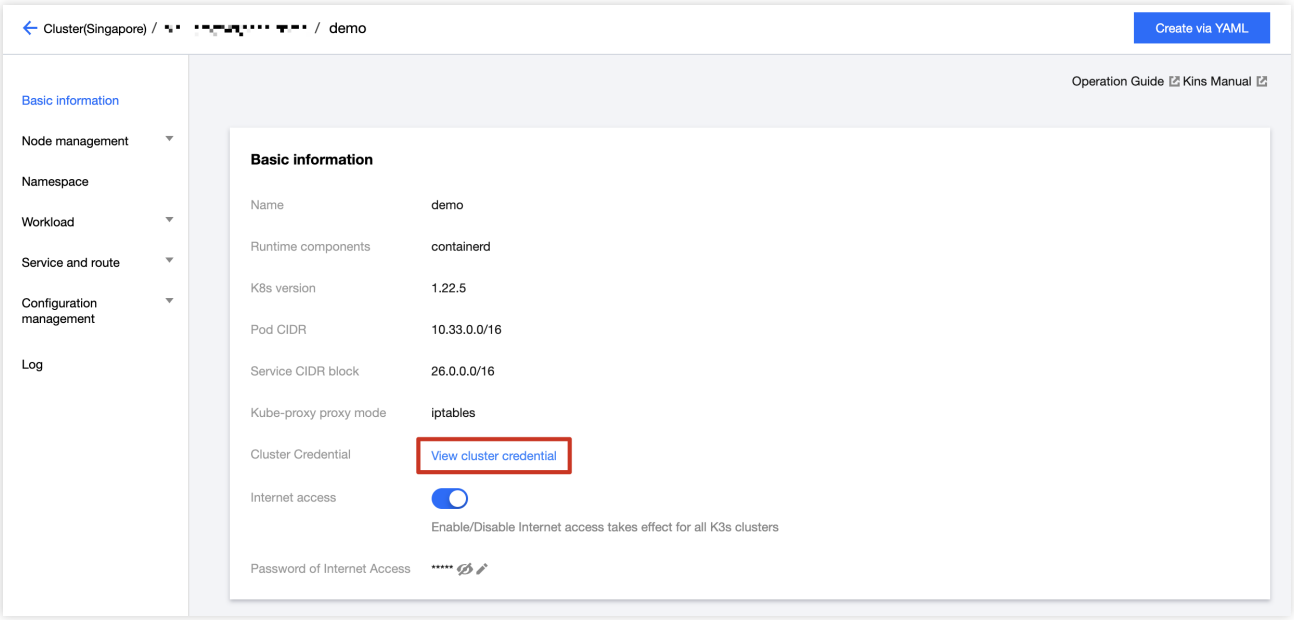
在**开启外网访问**中，请输入代理需要访问的密码，如下图所示：



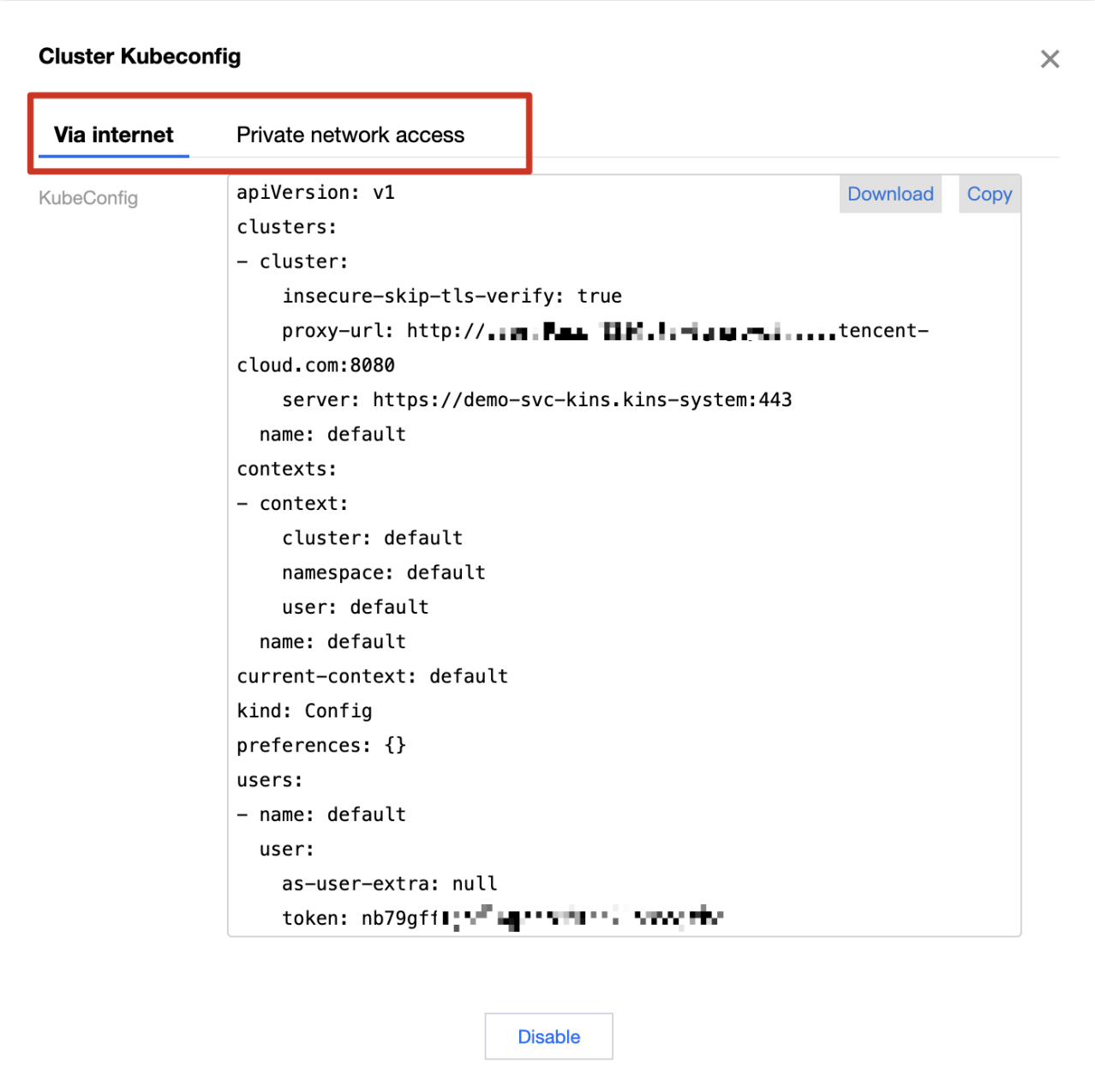
注意：

开启外网访问会通过云上 CLB 的能力对外提供 http/https 代理访问，因此这里会强制需要用户设置代理密码用于鉴权认证，保证代理安全性。

3. 您可以通过云上 kubeconfig 或者边缘侧 kubeconfig 两种方式登录边缘 K3s 集群。单击查看**集群凭证**，如下图所示：



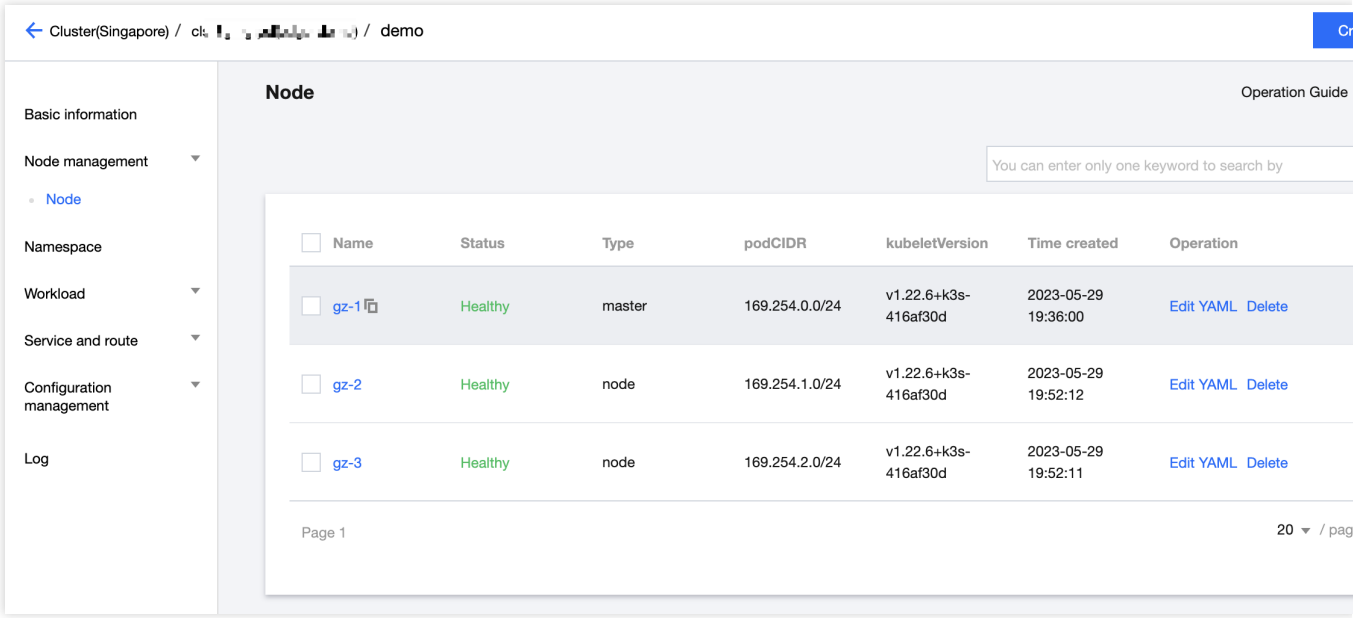
在集群访问凭证中查看，如下图所示：



如果您想要从公有云侧访问边缘 K3s 集群，可以下载/拷贝**公网访问** kubeconfig。

如果您想要从边缘侧节点上直接访问本地 K3s 集群，可以下载/拷贝**内网访问** kubeconfig。

4. 您也可以直接使用容器服务控制台对边缘 K3s 集群进行一些基本操作，如下图所示：

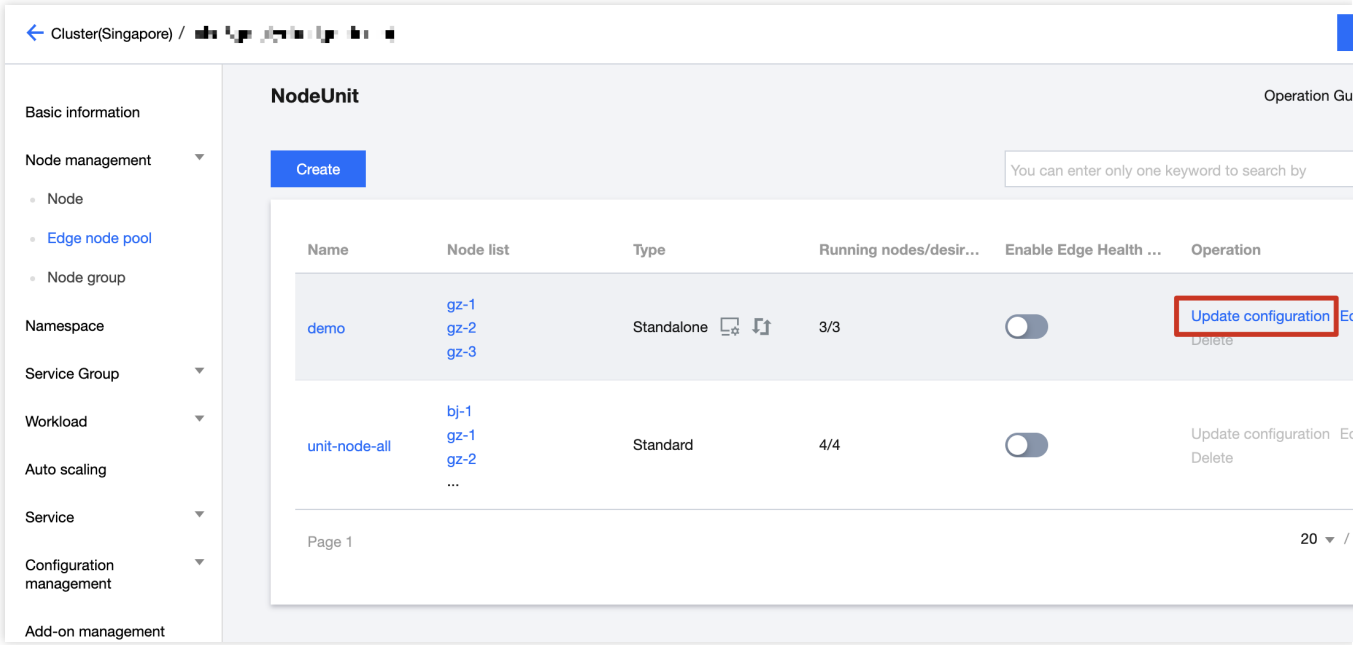


边缘 K3s 集群添加节点

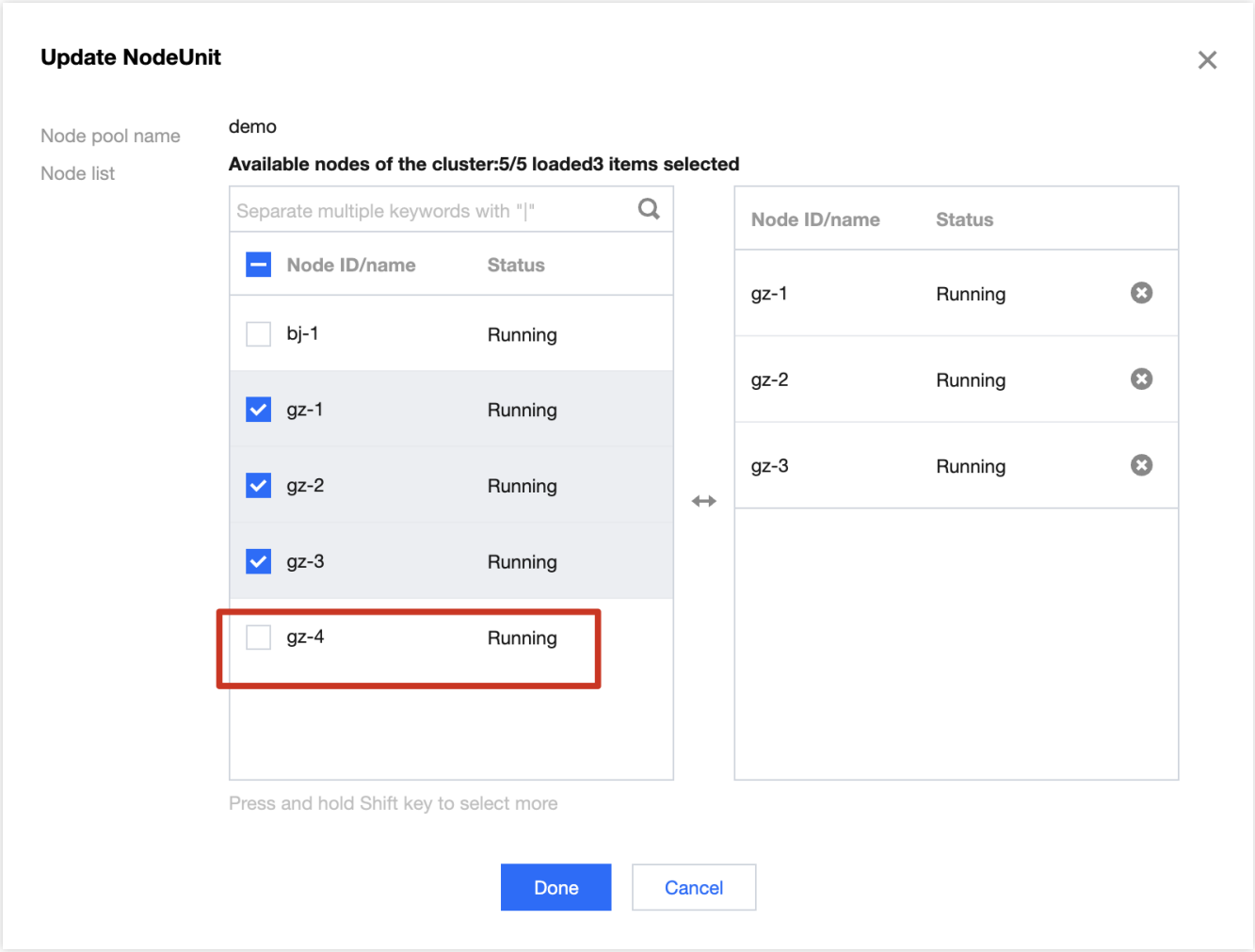
注意：

当前产品形态暂时只支持用户在云上控制台添加边缘 K3s 集群的新节点，暂时不支持 K3s 集群离线后独立添加节点。用户可以直接通过云端在 NodeUnit 中添加节点，添加的新节点会自动加入到边缘 K3s 集群中。

1. 在边缘节点池列表页面，选择指定的 NodeUnit，单击**更新配置**。如下图所示：

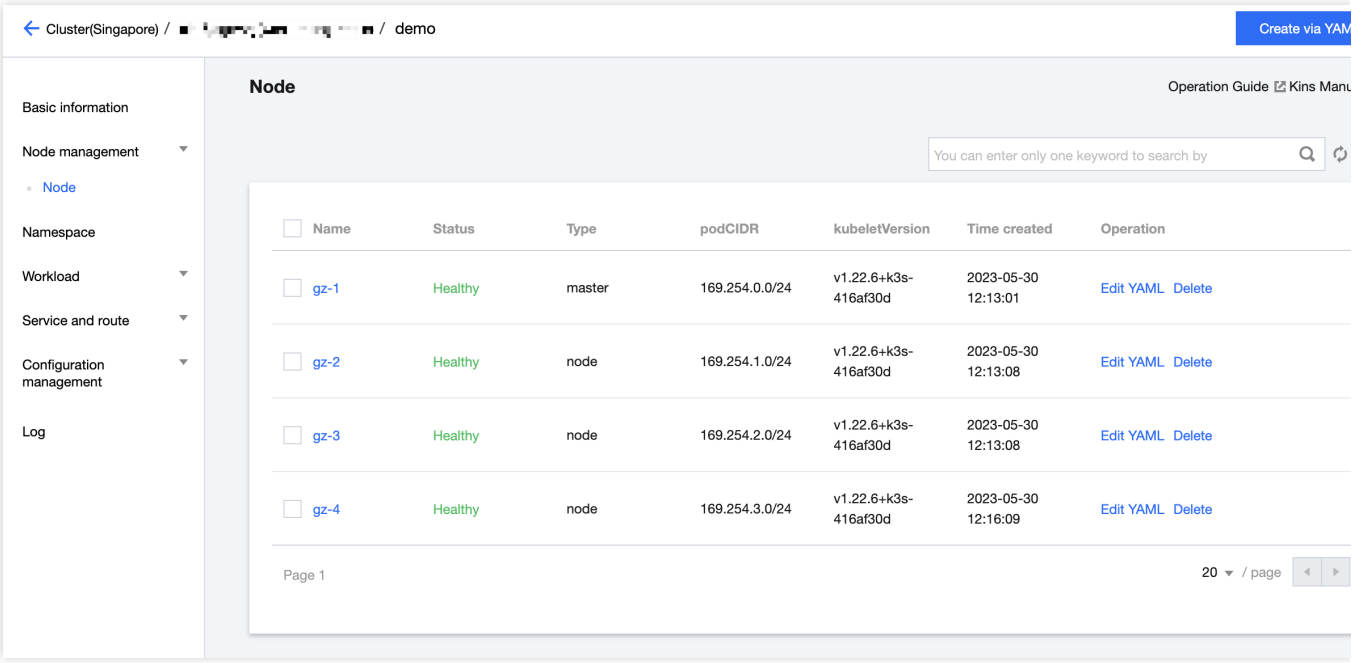


2. 在**更新节点池**中，添加新节点 **gz-4**，如下图所示：



3. 单击**完成**，可以观察到此节点已经加入到 demo 的 NodeUnit 中。

4. 稍等一段时间后，通过访问边缘节点池链接，进入边缘节点池控制台页面，观察集群节点信息，会发现添加的节点已经加入边缘 K3s 集群，如下图所示：



5. 通过云上的 NodeUnit 交互操作，您可以在边缘节点池中任意添加计算节点了。

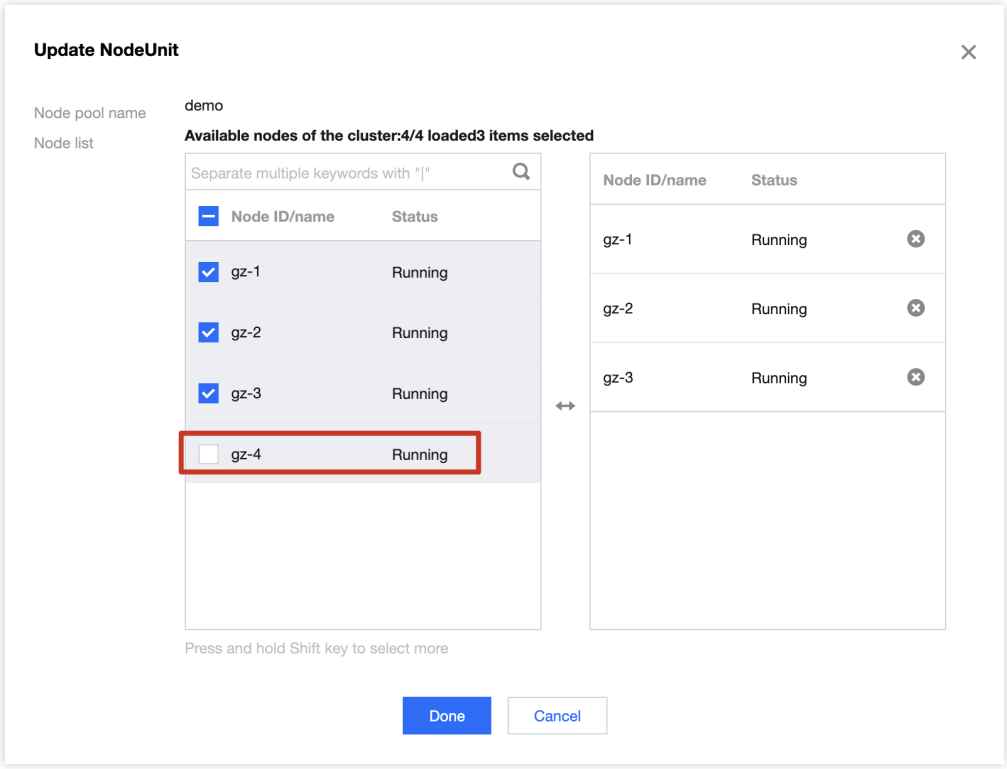
边缘 K3s 集群删除节点

当前产品暂时仅支持用户在云上删除边缘 K3s 集群的节点。用户有两种方式从边缘 K3s 集群中删除节点。

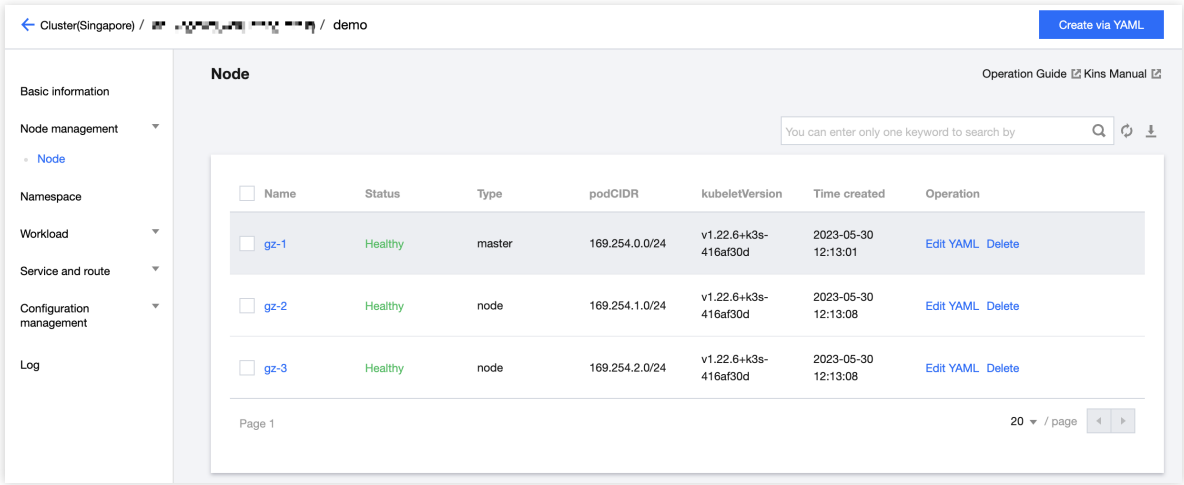
方式1

如果用户仅希望节点从 NodeUnit 中清除相应节点，而不是需要从 TKE-Edge 集群中移除，那么只需要修改 NodeUnit 的节点配置即可，如下图取消选择相应的集群即可：

1. 在边缘节点池列表页面，选择指定的 NodeUnit，单击**更新配置**。
2. 在**更新节点池**中，取消选择指定的节点，如下图所示：



3. 单击**完成**，可以观察边缘 K3s 集群节点状态，相应的节点会被清除，更新如下图所示：



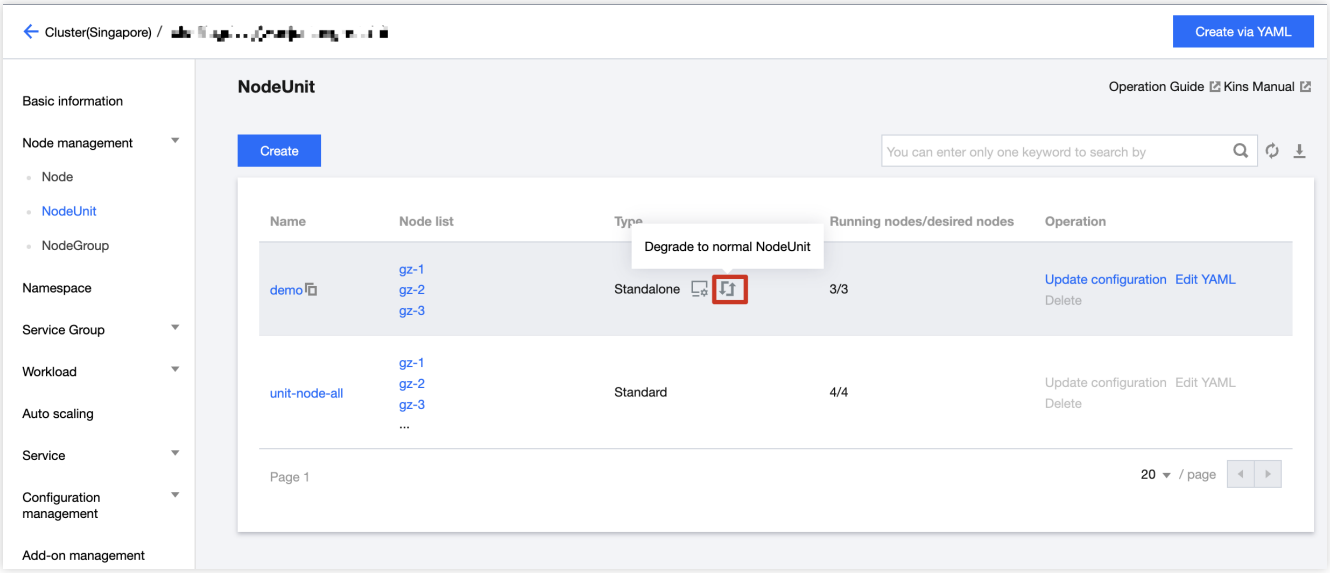
方式2

如果用户想要删除某个 TKE-Edge 上的节点，不再使用，可以直接在 TKE-Edge 节点页面移除相应节点，对应边缘 K3s 集群中的节点也会被对应清理。

边缘 K3s 集群降级

如果用户不再需要使用边缘 K3s 独立集群，期望恢复到边缘 TKE-Edge 标准边缘节点池方式使用。操作步骤如下：

1. 在边缘节点池页面，选择需要降级的 NodeUnit，单击**降级为普通节点池**。如下图所示：



2. 降级为标准节点池后，默认行为和标准 NodeUnit 行为一致。

边缘 K3s 集群删除

注意：

TKE-Edge 产品默认只能从界面删除标准节点池，独立节点池的删除按钮默认置灰，不允许用户直接删除独立节点池。

若您要删除独立节点池，需要首先根据边缘 K3s 集群降级操作，将独立节点池降级为标准节点池，然后才可以从节点池列表页中进行删除。

边缘容器支撑能力
边缘容器监控

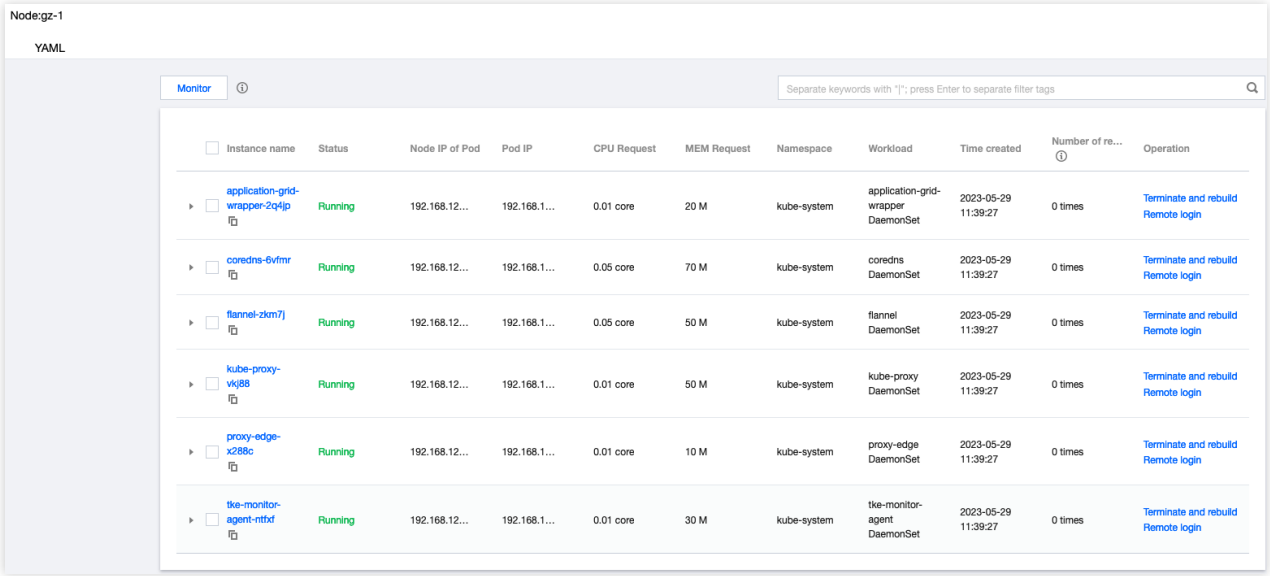
最近更新时间：2023-06-01 11:22:54

监控组件能力是一个平台系统的基础支撑组件，边缘容器复用了腾讯云的云监控能力，可以将边缘侧的监控指标通过边缘容器的 “Tunnel 云边通道”上传到云端，提供基础的监控能力。您可根据以下操作查看集群监控信息：

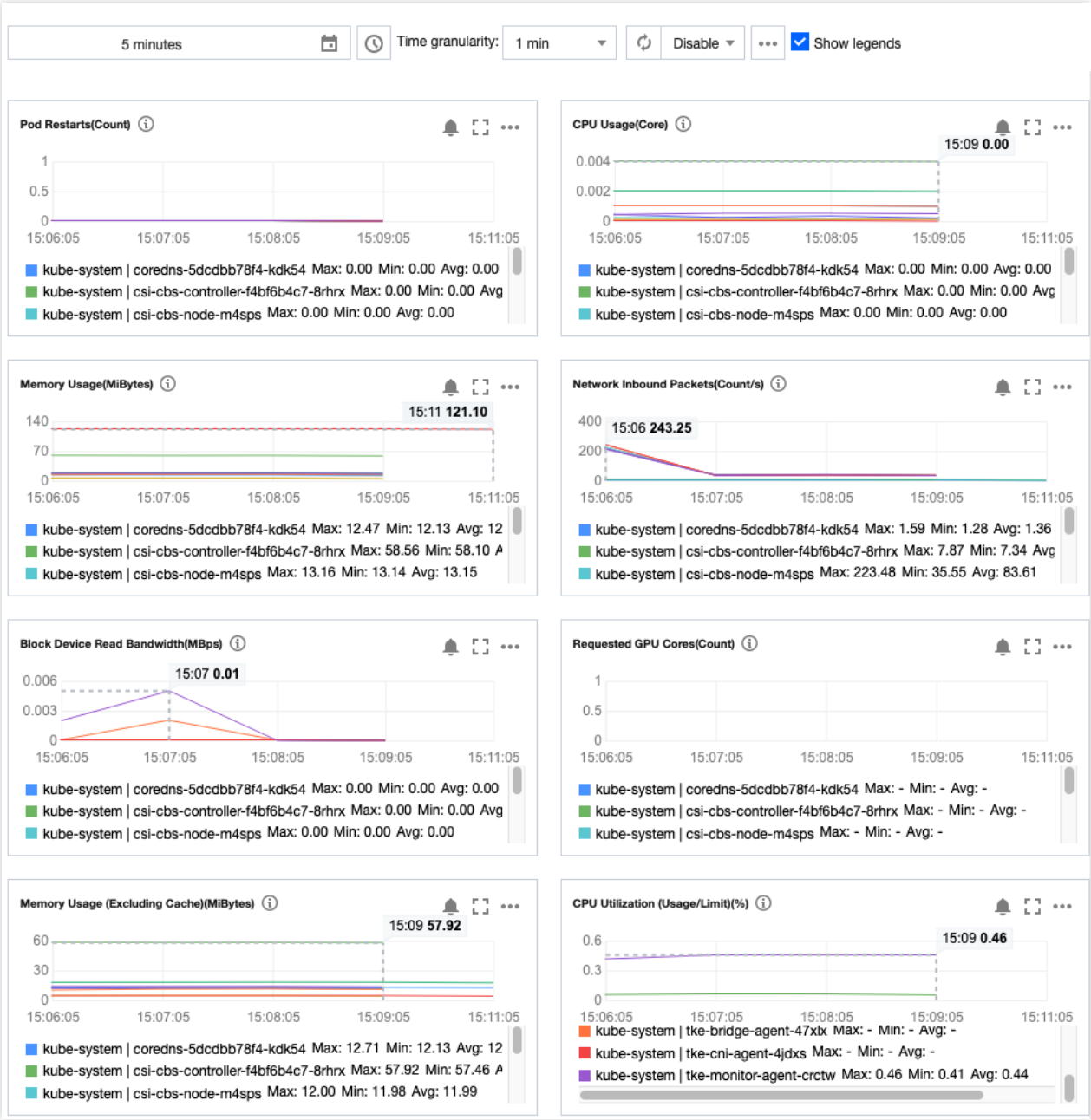
操作步骤

基础监控信息

1. 登录 容器服务控制台，选择左侧导航栏中的**集群**。
2. 在集群管理页，单击边缘集群 ID，进入该集群详情页。
3. 选择页面左侧**节点管理 > 节点**，进入节点列表页面，单击**监控**查看具体监控信息，如下图所示：



4. 查看监控信息，如下图所示：



具体监控的 Metric 包括：

事件：Pod 重启次数、Pod Ready 状态。

CPU：CPU使用量（核）、CPU 利用率（占 limit）、CPU 利用率（占节点）、CPU 利用率（占 request）。

内存：内存使用量(MBytes)、内存使用量（不包含cache）、内存利用率（占节点，不包含cache）、内存利用率（占request）、内存利用率（占request，不包含cache）、内存利用率（占limit）、内存利用率（占limit，不包含cache）、内存利用率（占节点）、内存使用量_working_set、内存利用率_working_set占request、内存利用率_working_set占limit、内存利用率_working_set占节点。

网络：网络入包量(个/s)、网络出流量(MBytes)、网络入流量(MBytes)、网络出带宽(Mbps)、网络入带宽(Mbps)、网络出包量(个/s)。

GPU(如果包含)：GPU显存使用率(%)、GPU编码资源使用率(%)、GPU解码资源使用率(%)、GPU流处理器使用率(%)。

存储：rootfs 空间大小(MiB)。

说明

边缘容器提供的上述“基础监控”能力现阶段免费提供给用户使用。

云原生监控

平台自带监控信息存在一些不足之处：

Metrics 指标较少，同时不能够自定义指标。

同时缺乏云原生相关 Metrics 的监控能力。

因此边缘容器服务也同时支持接入 **Prometheus 云原生监控**的能力。通过该产品，边缘容器平台可以支持自定义指标，且部署后自带免费节点指标 node-exporter、kubelet 指标，以及集群维度 kube-state-metrics 信息，能够为边缘集群提供完整的云原生监控能力。具体可参考 Prometheus 监控。

说明

此产品为付费产品，请您了解详细特性后按需购买使用。

边缘容器日志

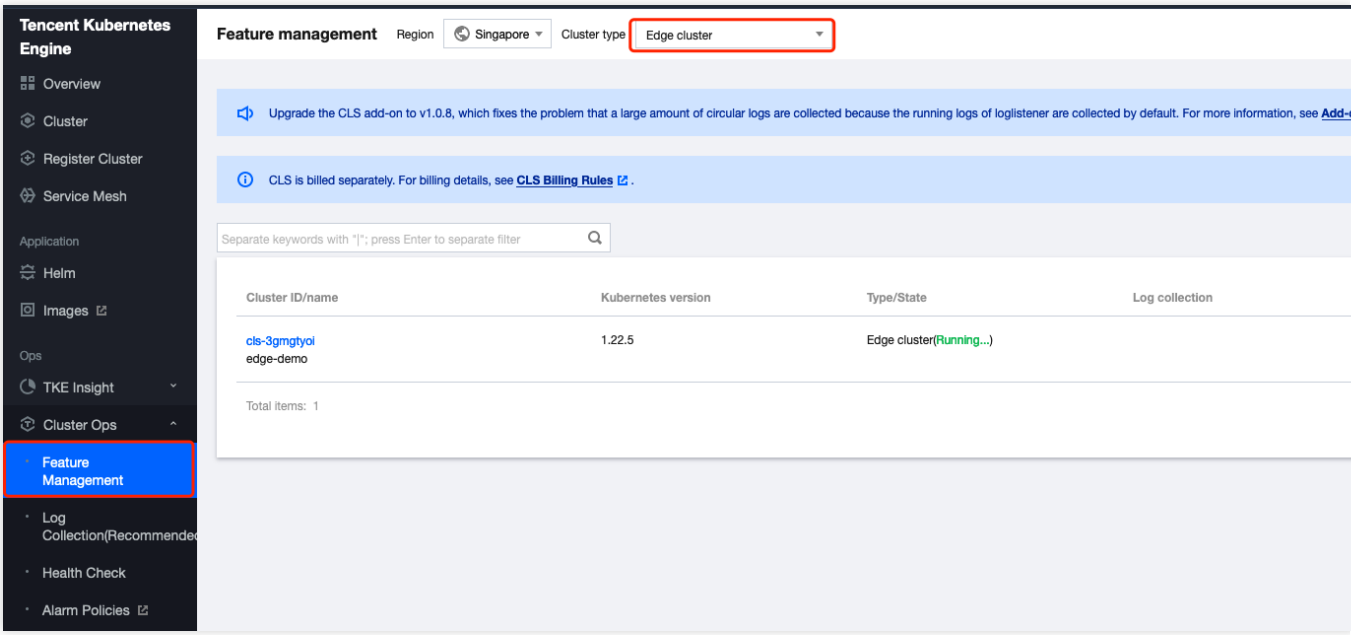
最近更新时间：2023-06-01 11:22:54

日志组件能力是一个平台系统的基础支撑组件，边缘容器复用了腾讯云的日志服务，可以将边缘侧的日志数据上传到云端，提供完整的日志采集以及检索能力。您可根据以下操作开启日志采集功能：

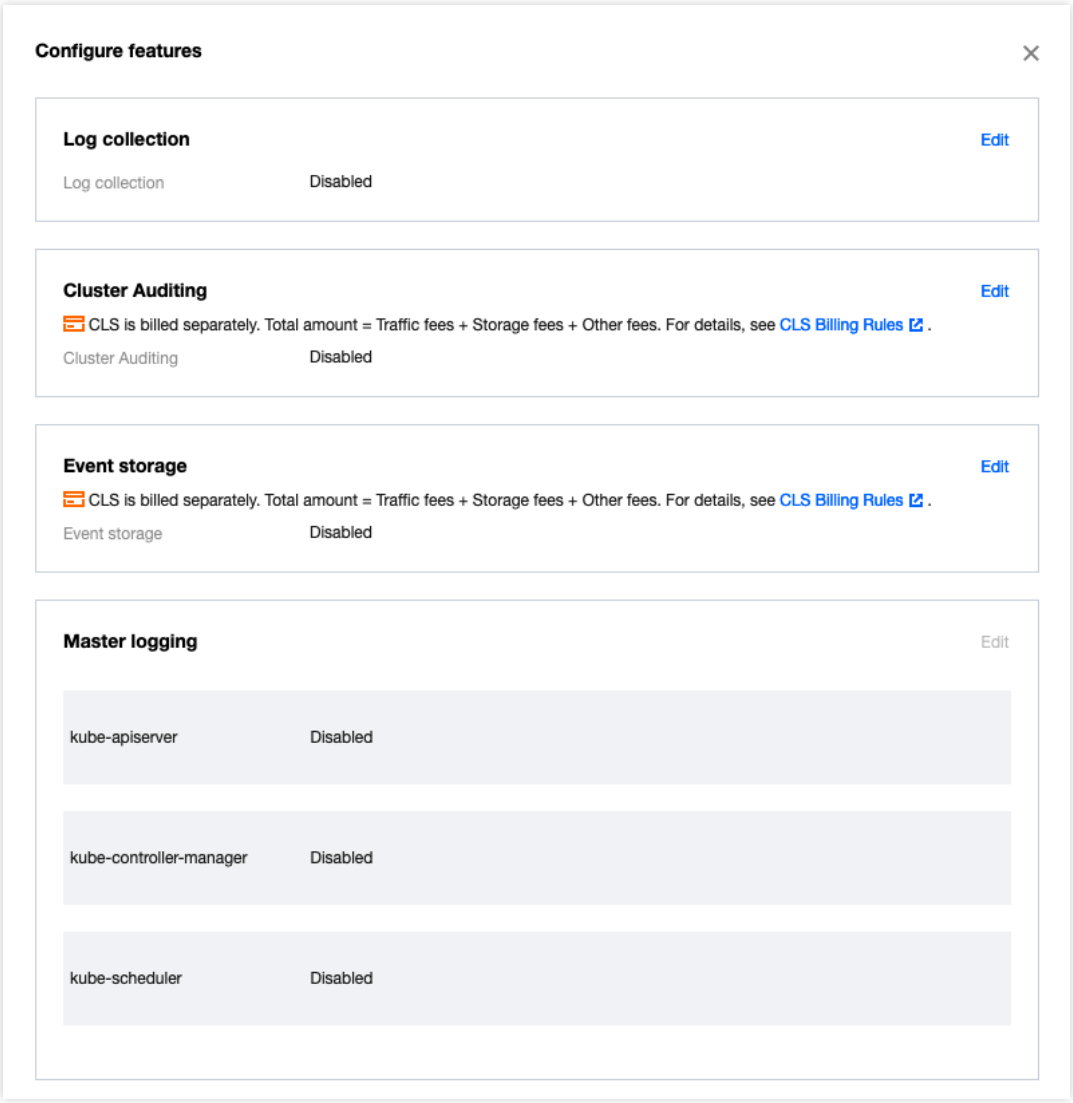
操作步骤

开启边缘容器日志能力

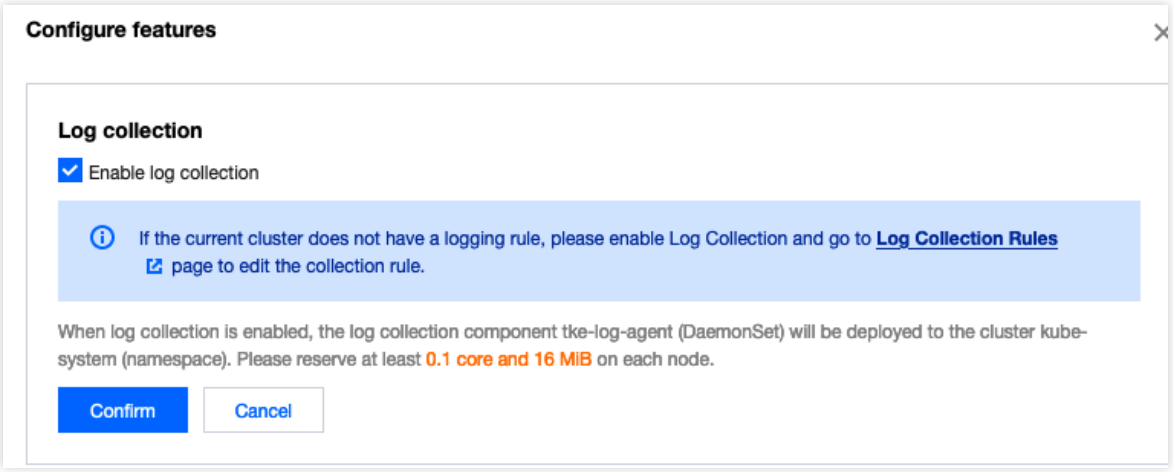
1. 登录 容器服务控制台，选择左侧导航栏中的**运维功能管理 > 功能管理**。
2. 在“功能管理”页面上方选择地域和集群类型。如下图所示：



3. 单击需要开启日志采集的集群右侧的**设置**。
4. 在“设置功能”页面，单击日志采集右侧的**编辑**。如下图所示：



勾选“开启日志采集”后单击**确定**。如下图所示：



5. 开启后，您可以在日志服务控制台查看或者检索日志，具体请参考 日志服务。

运维管理

edgectl 边缘节点管理工具说明

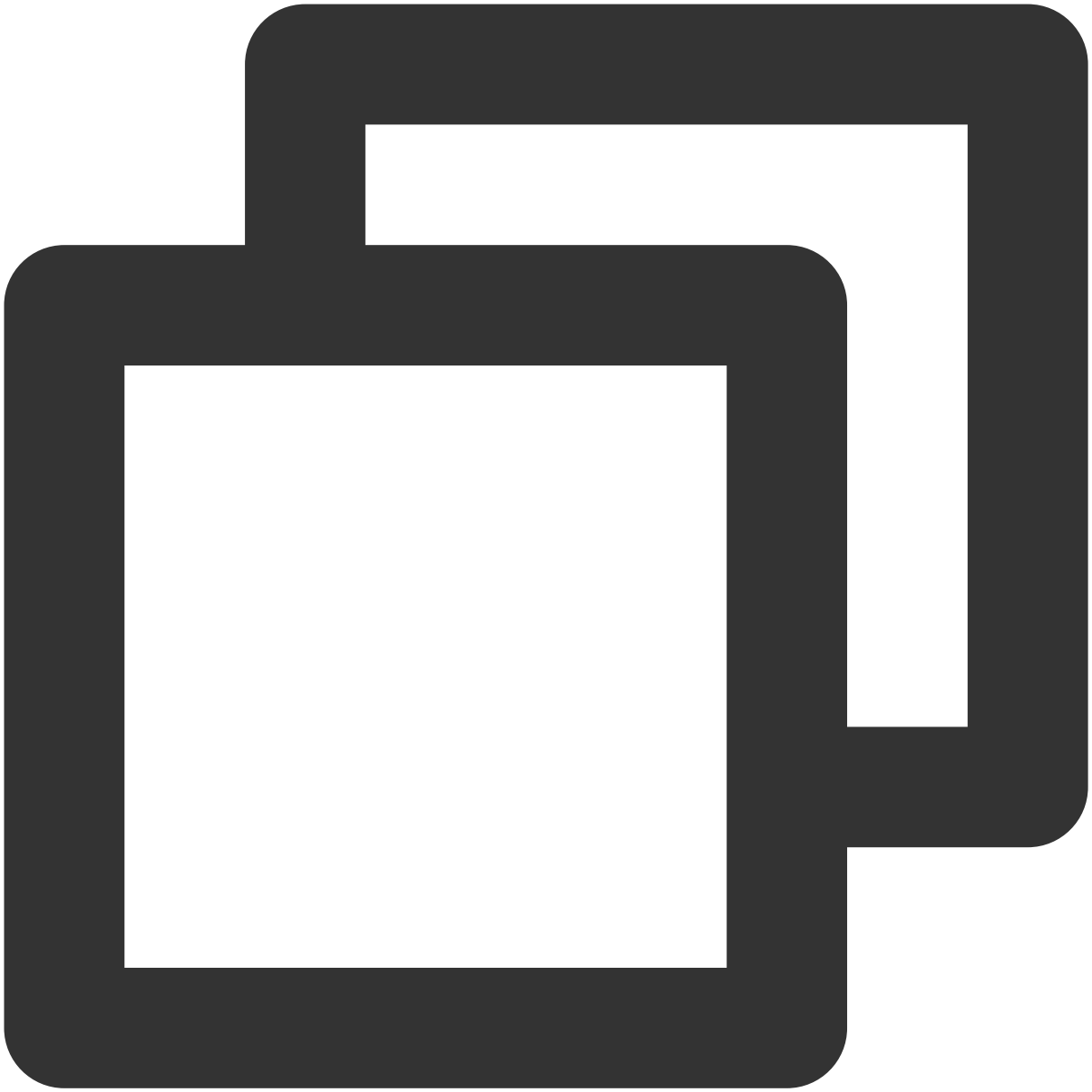
最近更新时间：2023-06-01 11:22:54

edgectl 是边缘节点管理工具，用于远程添加边缘节点。您可通过**节点管理** > 脚本添加节点操作获取该工具。

edgectl 提供了以下三个命令：

命令	描述
edgectl check	检查节点是否满足安装边缘节点的条件。
edgectl clear	清理边缘节点。
edgectl install	安装边缘节点。

示例代码如下所示：



```
# ./edgectl -h
Usage:
   edgectl command [flags]
Available Commands:
   check        Check the edge node if to be add to clusters
   install      Install components to edge node
   clear        Clear edge node and recovery as usual

Flags:
   -h, --help   Help for edgectl
```

edgectl check 命令

含义

检查节点是否满足安装边缘节点的条件。check 项目包含内容：

check 是否 root 用户。

check 系统是否在支持范围内。

check 交换区是否关闭。

check 防火墙是否关闭。

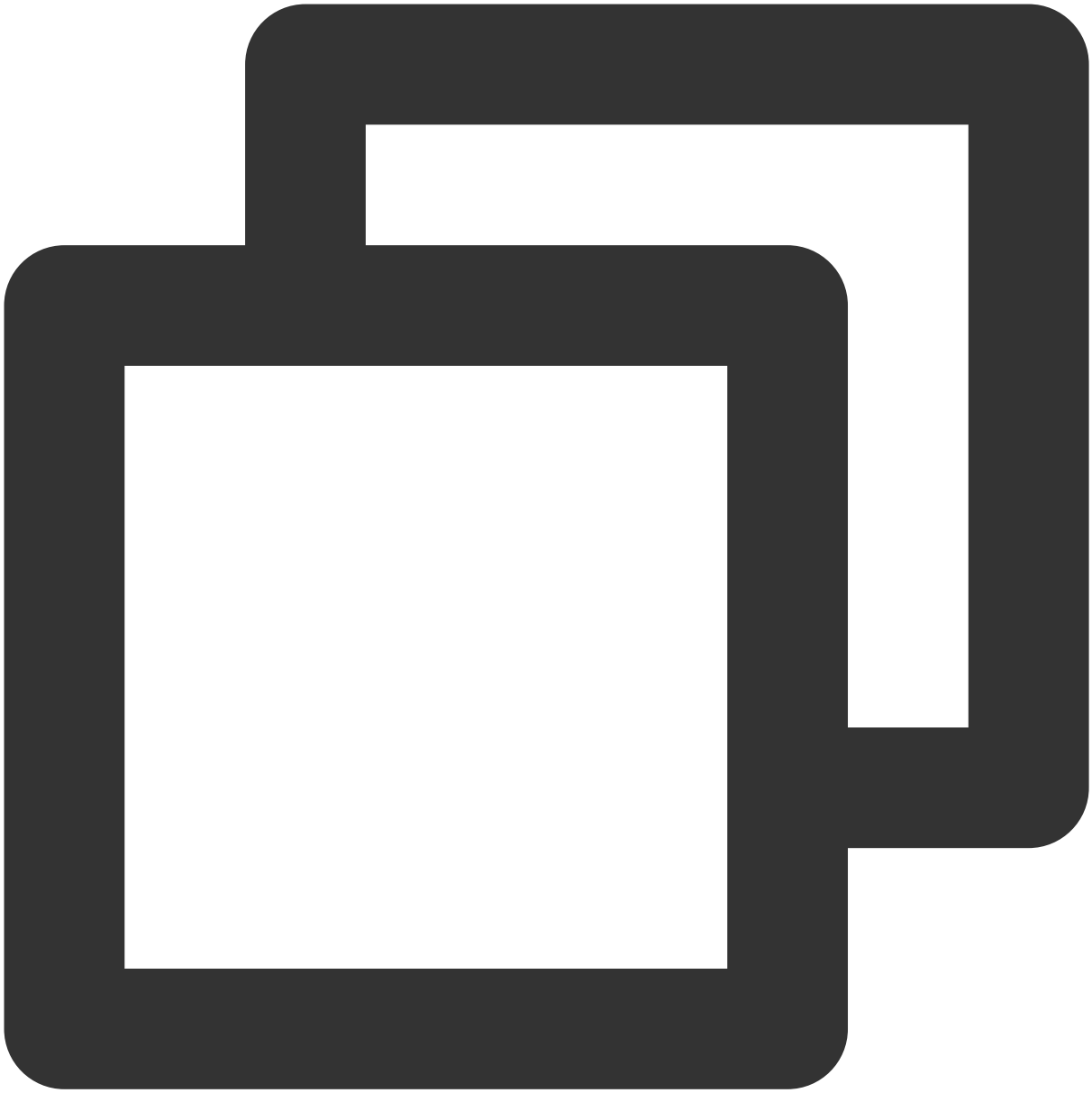
check ufw 是否关闭。

check 端口是否被占用（check 的端口有 1443 {10249..10259} {51000..51020}）。

check 是否开启 cgroup memory。

check 节点之前是否安装过 kubeadm docker kubelet kubectl。

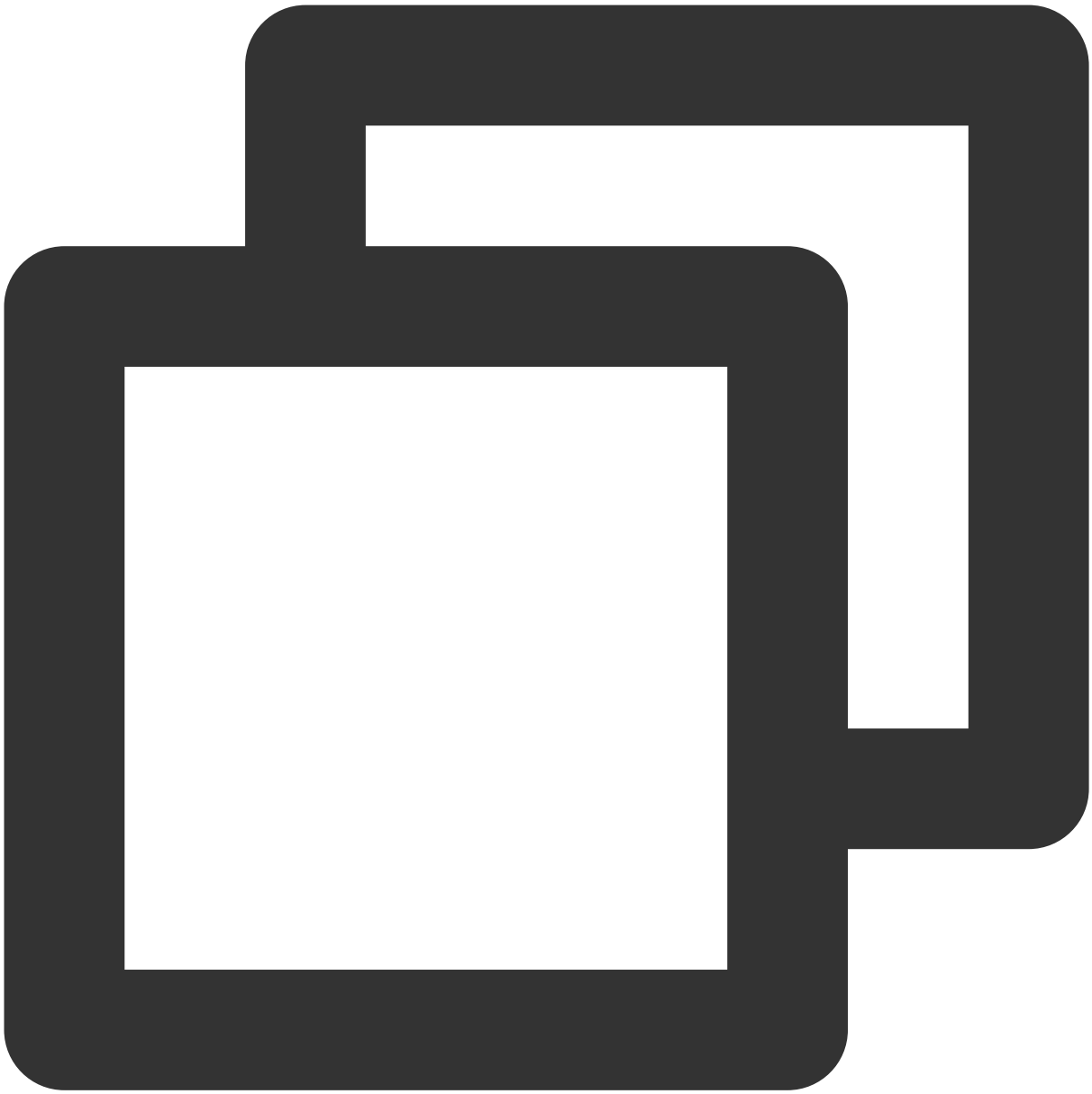
使用示例



```
# ./edgectl check
Unit firewalld.service could not be found.
 WARN >> Port: 1443 occupied. Please turn off port service.
...
 WARN >> The machine is not clean. Please reinstall the system.
/usr/bin/kubelet
...
 >> Check Environment Finish! <<
```

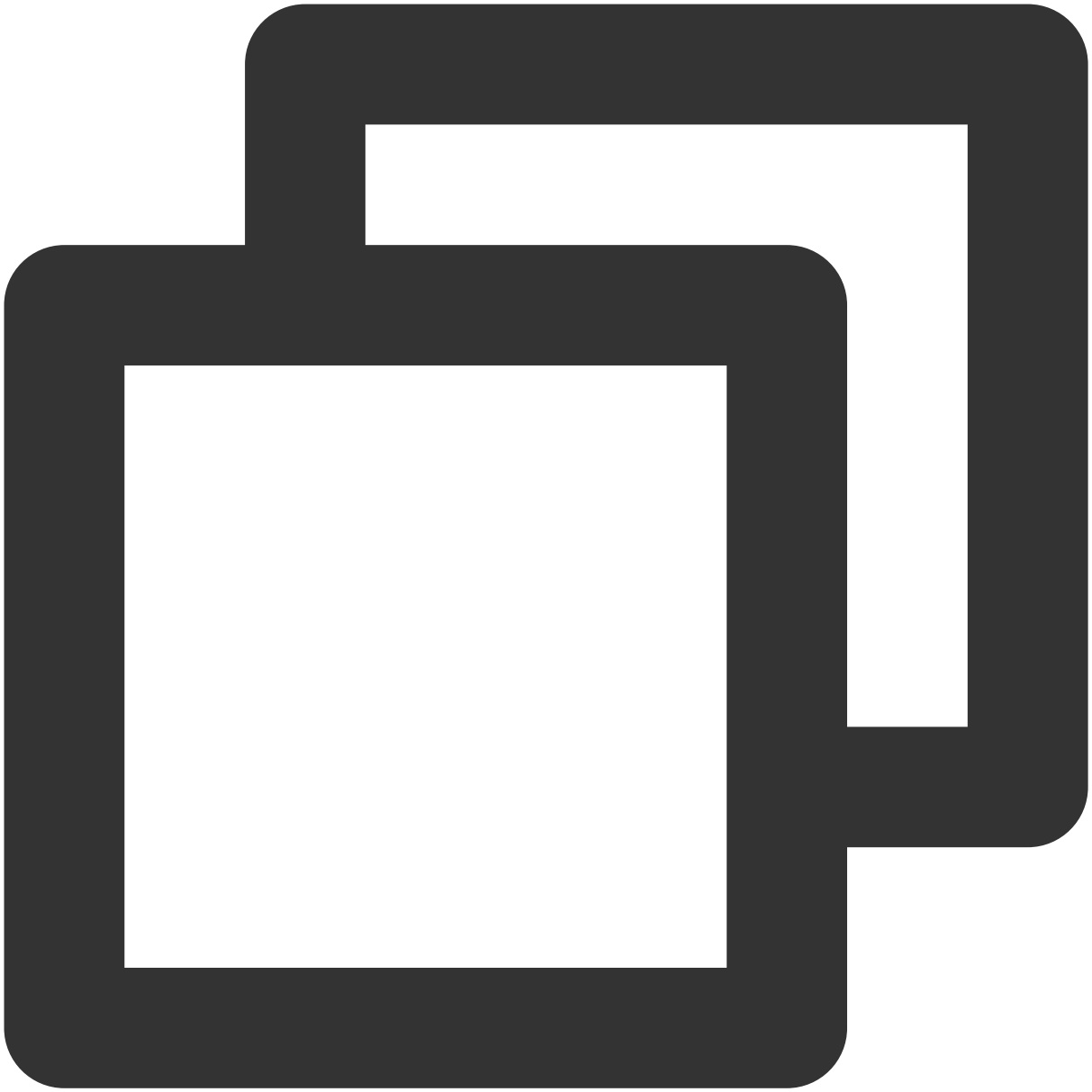
例如本次执行示例代码后，提示了用户包含以下2个风险项目：

该边缘节点的端口1443已经被占用，请关闭其服务。



```
WARN >> Port: 1443 occupied. Please turn off port service.
```

节点有 kubelet 残留，建议重装节点系统，或执行 `edgectl clear` 命令，进行清理操作。



```
WARN >> The machine is not clean. Please reinstall the system.
/usr/bin/kubelet ## 之前安装过kubelet
```

edgectl clear 命令

含义

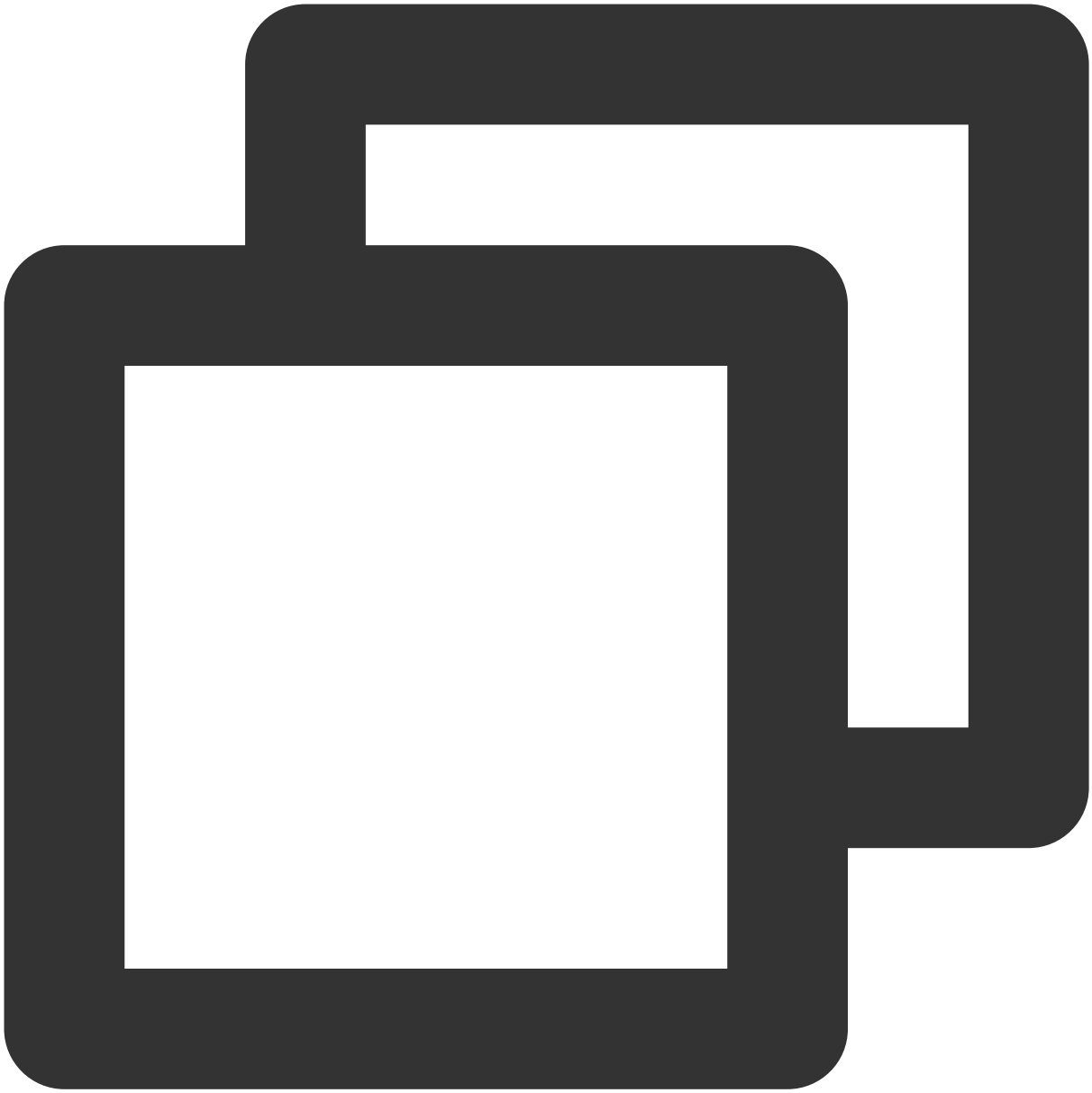
清理边缘节点。clear 命令会清除如下信息：

删除边缘节点上运行的所有容器和 Pod。
停止 kubelet、lite-apiserver、docker。
删除创建的网络信息和路由信息。
删除如下文件夹或文件：
/etc/kubernetes
/etc/docker
/root/.kube/config
/var/lib/kubelet >/dev/null 2>&1
/var/lib/cni
/etc/cni
/etc/sysconfig/kubelet/
/etc/sysconfig/lite-apiserver
/data/lite-apiserver >/dev/null 2>&1
/usr/lib/systemd/system/{kubelet,docker,lite-apiserver}.service

注意

`edgectl clear` 会删除节点上的所有容器和 Pod，请谨慎执行。
`edgectl clear` 会删除相关的文件夹或文件，请提前备份重要的资料。
`edgectl install` 命令默认会执行 `edgectl clear`，执行 `edgectl install` 前请考虑 `edgectl clear` 删除的风险项。

使用示例

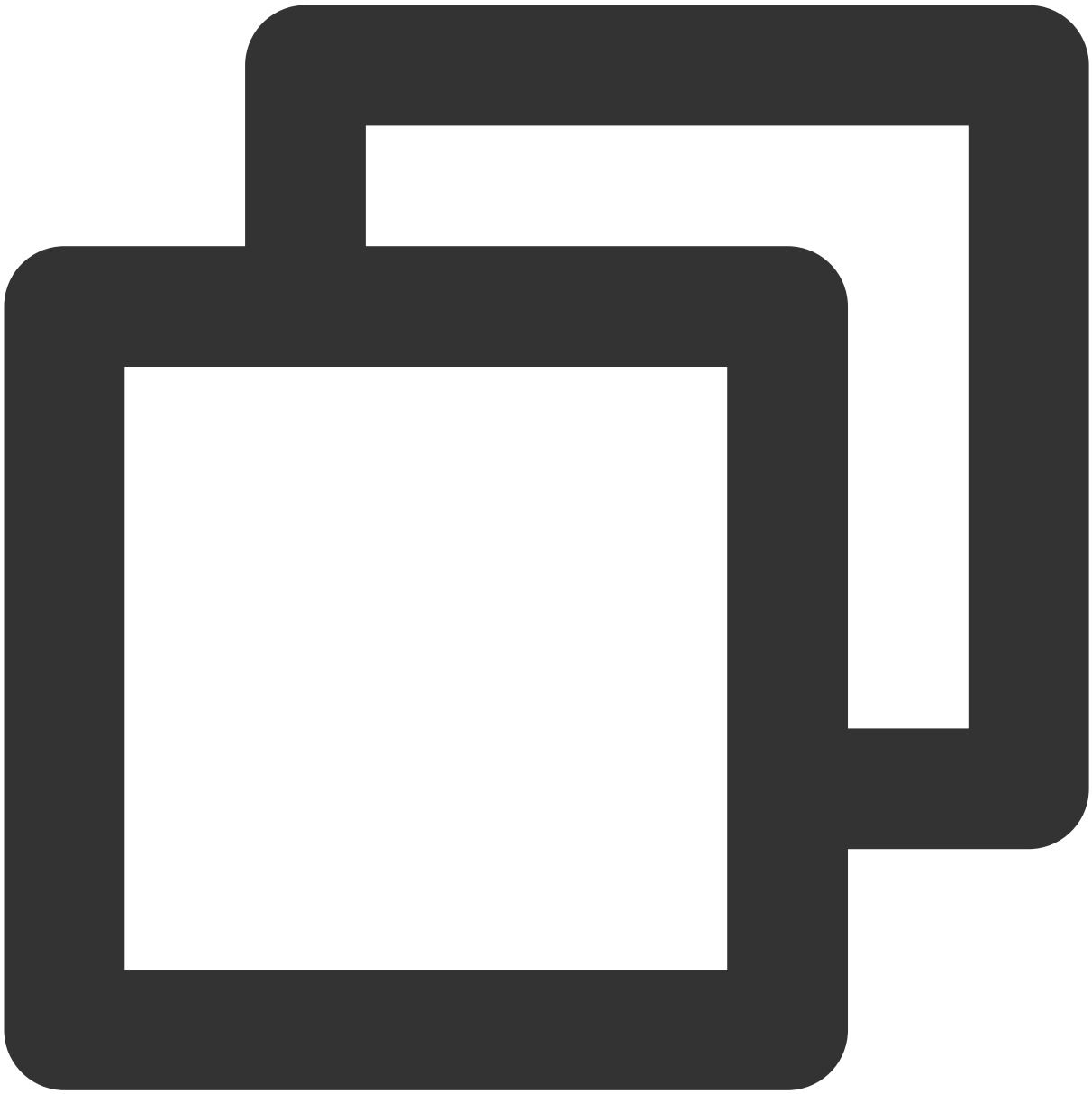


```
# ./edgectl clear
removed '/etc/kubernetes/cluster-ca.crt'
...
  >> Clear Node Complete! <<
```

edgectl install 命令

含义

安装边缘节点。执行如下命令，安装边缘节点。



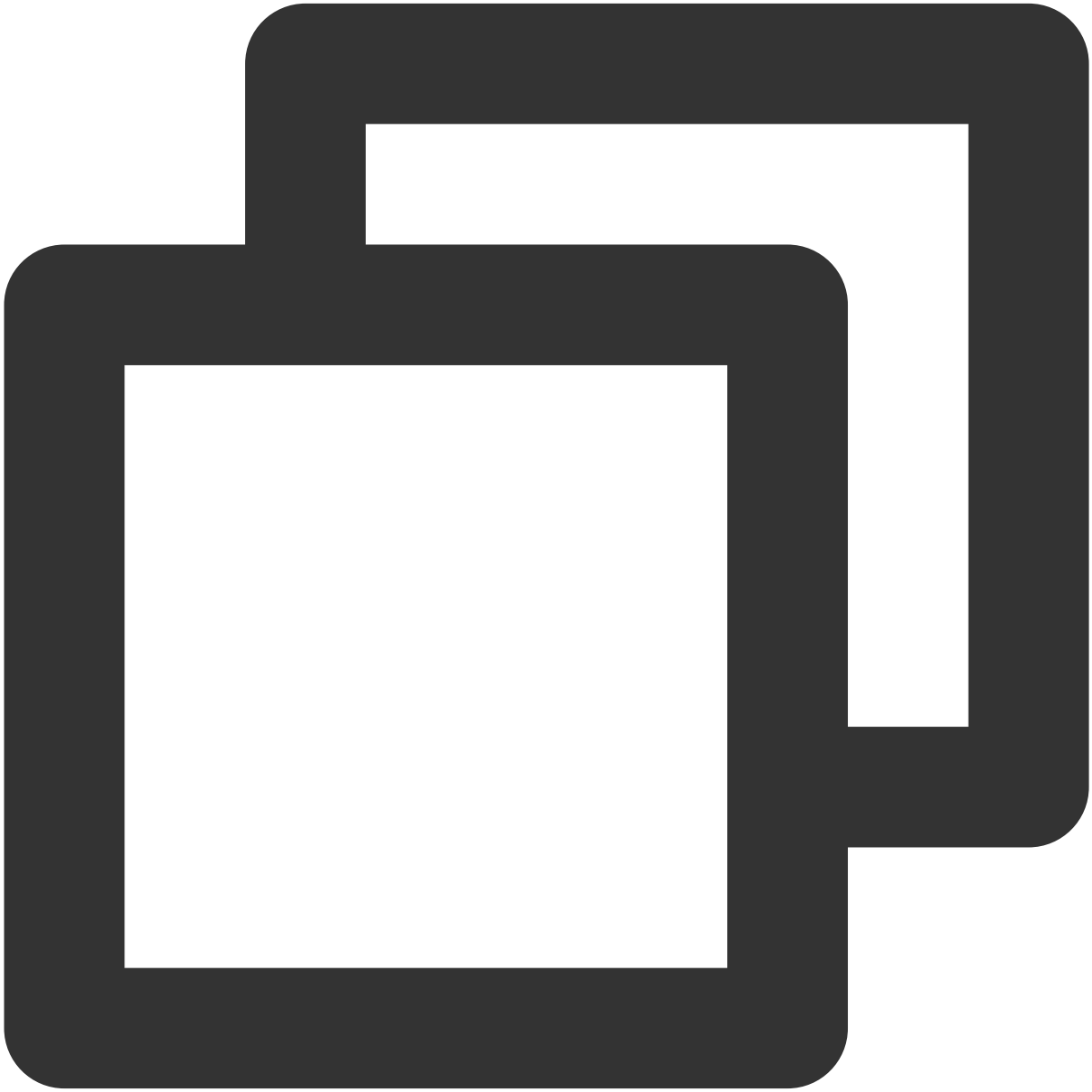
```
# ./edgectl install -h
Usage:
   edgectl install [flags]
Flags:
   -n,  --node-name      Node name in edge cluster. Must！## 节点的名字，必填
   -i,  --interface      Default network interface name. ## 节点的默认网卡名，可选项
```

--node-name：边缘节点的名字，简写 `-n` 。

名字是必传选项，必须严格遵守 kubernetes 的 node 节点的命名规范。

请保证节点名在所加的集群里面唯一，否则会产生同一节点名所对应节点不断切换的问题。
节点名不支持直接用 IP，否则会引起 `kubectl log` 和 `kubectl exec` 命令失效。
--interface：边缘节点的默认网卡名，简写 `-i` 。
可选项，边缘节点的默认网卡名，填写错误会引起 flannel 和 coredns 组件异常。
默认值为从腾讯云页面填的指定网卡名称，指定 `--interface` 将覆盖原默认值。

使用示例



```
# ./edgectl install  --node-name node-192.168.67.91 --interface eth0
NOTE:
```

```
        input: [ edgectl install --node-name node-192.168.67.91 --interface eth0 ] ##
        logPath: /tmp/tke-edge-install.log ## 本次安装的输出的日志位置
        success-message: Install Edge Node: node-192.168.67.91 Success! ## 安装成功后的

  Start Install Edge Node node-192.168.67.91, Please Waiting...
  Waiting Running of the base service
  Dockerd kubelet lite-apiserver has Running! ## Dockerd kubelet lite-apiserver 安装成
    Install Edge Node: node-192.168.67.91 Success! ## 表示此边缘节点安装成功
```

此边缘节点安装执行完成后，可执行命令 `kubectl -n kube-system get pod` 查看所有 Pod 是否 Running。

`edgectl install` 命令**一小时有效**，可在有效期内多次添加不同的节点，如过期失效请重新进行 脚本添加节点 操作请求新的 edgectl。

要是安装过程中无提示退出，请参见 NOTE 中提示的安装日志位置 `logPath: /tmp/tke-edge-install.log` 查看具体的错误。

执行完没有 `Install Edge Node: node-192.168.67.91 Success!` 则认为安装失败，请查看安装日志排错。