

# 什么是云边协同

近两年，全球预计有超过百亿数量的终端和传感器，其中我国约占全球数量的 7% 左右，其节点规模远远大于互联网，物联网接入节点数量呈爆发性增长趋势。

物联网的发展，为网络传输的实时性、网络带宽带来新挑战，也对数据的存储、分析、处理提出新要求。集中式的云计算模型已然无法满足万物互联下的海量数据的高效传输以及处理需求。伴随着 5G 的发展，边缘计算应运而生。

云边协同将云原生能力延伸至边缘，采用边缘节点模式，将数据处理、业务应用、AI 模型等下沉到边缘端执行，解决物联网落地时响应实时性、数据隐私性、维护便利性等问题，从而满足行业数字化在敏捷联接、实时业务、数据优化、应用智能、安全与隐私保护等方面的关键需求。

## 边缘地位

边缘地位

# 功能特性

云边协同将云原生能力延伸至边缘，采用“端-边-云”一体化协同架构，帮助企业快速拓展安全、灵活、高效、可靠的边缘云原生能力，广泛适用于工业互联网、物联网、智慧工厂、智慧园区、智慧零售等领域。云端实现边缘节点的注册、管理及应用和配置的下发，边缘侧负责运行边缘应用，实现边缘自治，设备侧支持多协议终端接入，并提供标准接口对接设备能力。

云边协同平台的功能架构如下图：

## 功能架构

### 功能架构

云边协同支持以下功能特性：

## 边缘节点管理

支持接入海量的边缘节点，在平台中可以自动生成边缘节点的配置信息，支持 token 和证书两种高效便捷的纳管边缘节点方式，边缘节点可以在云端统一管理、监控和运维。

## 边缘应用/模型管理

- 边缘计算平台支持将边缘应用或模型以容器的形式快速部署到边缘节点运行。用户可以将自己的边缘应用程序或模型打包成容器镜像，上传至镜像仓库，然后通过云边协同平台将容器镜像部署到边缘节点运行。
- 云边协同平台支持对已部署的实例的运行数据、事件和日志信息进行查看。
- 容器具有更繁荣的生态，能帮助用户的应用无缝切换到其他的运行环境中，具有更好的可移植性，而且容器具有更好的资源隔离性，并支持 CPU、GPU 调度。

## 边缘设备管理

边缘计算平台支持终端设备连接到边缘节点，终端设备支持通过多种协议接入。终端设备接入后，可以在云端对终端设备进行统一管理。

## 应用网络

支持服务发现与流量代理，从而实现边缘业务流量闭环，解决边缘场景下边边业务之间通信的问题。

## 数据管理

边缘计算平台提供消息路由功能，用户可以配置消息路由规则，平台根据配置的消息路由将边缘消息转发至对应的消息端点，让消息按照配置的路径转发，灵活控制数据路由，并提高了数据安全性。

## 权限管理

云边协同的权限管理采用分层架构设计，旨在提供灵活且安全的权限控制。通过平台/文件夹/工作空间三级权限管理，用户可以根据不同的业务需求进行精细化的权限配置，确保资源的安全性和共享性。

系统内预定义了多种角色，每个角色拥有不同的权限范围。每个角色的权限范围可以根据实际需求进行调整。

为用户分配角色和权限时，需要遵循一定的流程。首先，确定用户的业务需求，然后选择合适的角色，最后平台中进行角色和权限的配置。注意在配置过程中需要确保权限的最小化原则。

## 边缘控制

- 提供对边缘节点的资源调度和管理运维功能。
- 提供对边缘节点上运行的应用和模型进行全生命周期管理。
- 提供数据采集功能，并且支持设备协议转换，可采集与边缘节点连接的关联设备的数据。
- 提供消息和数据传输通道，支持将边缘消息以 EventBus 协议传输至云端 REST 协议，云端 REST 协议传输至边缘 EventBus / ServiceBus 协议三种转发路径，同时支持非结构化数据，如图片、视频、文件等的云边双向传输。

# 产品优势

云边协同提供六大核心优势能力，为用户解决不同类型的场景业务需求，更高效协同云边算力资源和数据。

## benefits

### benefits

- 边缘自治
  - 边缘侧自为一套完整的体系，具有不依赖云端的离线处理能力
  - 弱网/无网环境下，边缘节点具备自我管理能力，保障应用业务的连续性
- 智能运维
  - 边缘节点全生命周期监控
  - 边缘应用全生命周期监控
- 可插拔设备接入
  - 设备侧支持 Modbus 等多种协议终端接入，并提供标准接口 DMI 对接设备能力
  - 支持通过指定格式文件将设备批量导入
- 全流程安全保证
  - 提供平台、文件夹、工作空间多级权限管理，资源隔离
  - 支持云边双向数据加密传输，比如 TLS 加密
  - 提供审计日志功能
- 数据协同
  - 提供云边消息路由，支持时序消息的双向传输
  - 提供云边数据通道，支持非结构化数据的双向传输

- 边缘智能
  - 支持模型以容器模式在边缘节点运行
  - 支持模型的批量部署和升级

## 应用场景

云边协同适用于以下场景。

### 智慧工厂

帮助用户快速搭建靠近工厂物联网设备数据源头的云边协同平台，提供实时数据采集分析，建立工厂分析模型，感知并且降低环境和生产过程中的风险，提升生产的效率，降低生产的成本。

- 适配多种工业协议，实现生产设备统一接入管理，快速采集工业数据，并进行本地数据预处理和分析
- 边缘侧实时感知环境风险并监控生产过程，快速响应决策
- 实现统一应用、模型下发，节点状态统一监控

use case

use case

### 智慧门店

用边缘计算、人工智能、云原生技术重构人、货、场（消费场景）的管理，寻找新机会、带来新效率、提升用户体验。

- 计算和业务能力下沉，响应速度快，耦合度低
- 复杂网络环境下，门店营销视频自主播控，稳定性高

- 边缘节点和应用监控、日志云端统一管理、分析，故障快速定位，提升运维效率

use case

use case

## 智慧农业

云边协同平台能够对农业设备进行精确控制和实时数据传输，同时，边缘计算作为云计算的补充，与人工智能一起在网络边缘侧和更接近数据源的设备侧提供实时、高效的本地决策。

- 云端对设备进行管理、监控和控制
- 边缘测实时响应温湿度变化，智能调节设备运行

use case

use case

## 基本概念

### • 边缘单元

指容器运行所需计算资源的集合，包括云端 Master 和边端工作节点 Node。边缘单元跟 K8s 中的集群是同一概念，但组网方式有所不同。Master 节点部署在云端，一个云端 master (主备多台)对应一个边缘集群。

### • 边缘节点

这是容器集群组成的基本元素，既可以是云主机，也可以是物理机，用于运行容器化应用的载体，边缘应用将以 Pod 的形式在节点上运行。

### • 批量注册节点

相同类型的边缘节点能够预安装软件，在节点开机联网后能自动纳管到平台中。批量注册节点与边缘节点满足一对多的关系，提高管理效率，也节约了运维成本。

### • 边缘节点组

将节点按照特定属性抽象成节点组概念,以节点组的维度对不同边缘区域下的节点进行统一管理和运维。

- **终端设备**

终端设备可以小到传感器、控制器,大到智能摄像机或工控机床;终端设备可以连接到边缘节点,支持通过 Modbus 协议接入并进行统一管理。

- **工作负载**

是管理应用副本的一种 API 对象,具体表现为没有本地状态的 Pod, 这些 Pod 之间完全独立、功能相同,能够滚动更新,其实例的数量可以灵活扩缩。

- **批量工作负载**

将相同配置或差异化较小的工作负载定义部署到节点组,是一个任务或批量部署动作。

- **配置项**

是以键/值对形式保存的非机密性配置信息, Pod 可以将其用作环境变量、命令行参数或者数据卷中的配置文件。

- **密钥**

是以键/值对形式保存的、包含密码、令牌、密钥等少量敏感信息的对象,将敏感信息和容器镜像解耦,不需要在应用程序代码中包含机密数据。

- **消息端点**

发送或接收消息的规则端点。它包含 3 种类型: rest、eventbus、servicebus。

- **消息路由**

定义如何将消息从源消息端点传送到目标消息端点

- **应用网络**

提供非侵入式的微服务治理解决方案,实现完整的生命周期管理和流量治理,支持负载

均衡等多种治理能力。

## 云边协同 Release Notes

本页列出云边协同的 Release Notes，便于您了解各版本的演进路径和特性变化。

### 2024-12-31

#### v0.18.0

- **新增** 支持终端设备状态上报
- **优化** 边缘单元安装后状态获取到的时间展示

### 2024-11-30

#### v0.17.0

- **优化** 终端设备采集和上报间隔单位显示的问题
- **优化** 边端安装包下载提示问题

### 2024-10-31

#### v0.16.0

- **优化** 优化边缘节点安装命令 token 有效期提示



## 2024-09-30

### v0.15.0

- **优化** 支持在终端设备绑定节点状态下，编辑设备
- **优化** 优化了批量工作负载管理流程，交互更友好

## 2024-08-31

### v0.14.0

- **新增** 支持根据边缘节点规模，为边缘单元分配云端初始组件资源
- **新增** 边缘单元支持卸载策略设置
- **优化** 边缘节点接入命令支持设置云端 CloudCore 组件访问地址

## 2024-07-31

### v0.13.0

- **新增** 支持在网络状态良好的情况下提前将镜像在边缘节点上拉取好
- **优化** 新增新手引导，帮助用户快速上手边缘业务
- **优化** 终端设备属性值上报时间精确到秒，上报时间更精准

## 2024-06-30

### v0.12.0

- **新增** 支持创建和管理设备模型

- **优化** 设备参数配置，设备创建更灵活
- **优化** 支持边端安装包根据在离线环境动态更新下载链接

## 2024-05-30

### v0.11.0

- **优化** 边缘单元自定义仓库交互优化，创建更便捷
- **优化** 边缘节点接入支持配置云边通道通信协议，支持 WebSocket 和 QUIC 协议
- **优化** 边缘工作负载支持多实例配置

## 2024-05-06

### v0.10.0

- **新增** 支持用户自定义安装 MQTT 服务
- **新增** 支持工作负载升级策略配置
- **新增** 支持工作负载亲和性配置
- **新增** 支持添加边缘节点别名
- **优化** 优化了边缘节点接入流程，交互更友好

## 2024-04-07

### v0.9.0

- **新增** 支持用户自定义安装 MQTT 服务
- **优化** 边缘节点接入流程，交互更友好

- **修复** 客户端不同时区时间显示问题

## 2024-02-06

### v0.8.1

- **新增** 支持对工作负载进行版本回退操作
- **新增** 支持已部署工作负载的版本列表展示
- **新增** 支持对边缘节点状态，节点资源使用率、读写速率、接收发送速率等指标进行监控
- **新增** 支持对工作负载状态、资源使用率、读写速率、接收发送速率等指标进行监控
- **新增** 支持对 NVIDIA、华为昇腾、天数等 GPU 资源进行调度

## 2024-01-09

### v0.7.1

- **新增** 支持接入已安装在集群中的 KubeEdge
- **优化** 边缘资源批量删除交互
- **优化** 边缘单元镜像认证交互
- **优化** 自定义设备参数配置

## 2023-12-06

### v0.6.1

#### 新增

- **新增** 支持自定义终端设备接入，兼容多种设备协议
- **新增** 服务能力，支持边边通信
- **新增** 支持查看容器日志信息

#### 优化

- **优化** 创建边缘单元，组件 helm 仓库设置交互优化
- **优化** 创建批量任务，边端组件镜像仓库地址设置交互优化
- **优化** 边缘单元在“执行中”状态时，支持编辑、删除操作

#### 修复

- **修复** 通过安装器安装出来的云边协同，创建边缘单元时默认的镜像仓库为 release 问题
- **修复** 因边缘组件增加导致边缘单元状态判断异常的问题

## 2023-11-06

### v0.5.0

#### 新增

- **新增** 支持 x86\_64、arm64 架构边缘计算机以边缘节点的方式批量注册接入
- **新增** 支持暂停/恢复调度边缘节点
- **新增** 支持移除边缘节点
- **新增** 支持查看边缘节点概率信息，如 Kubelet 版本、操作系统、容器运行时、资源使用/分配状况等
- **新增** 支持查看边缘节点上的容器组列表、标签与注解、事件列表、状态
- **新增** 支持创建边缘节点组，通过指定节点或标签筛选方式
- **新增** 支持对边缘节点组进行编辑、删除操作
- **新增** 支持将应用以镜像的方式下发至边缘节点进行部署
- **新增** 支持通过 YAML 方式创建工作负载
- **新增** 支持多种镜像拉取策略以满足用户在应用重启时的不同操作需求
- **新增** 支持对 CPU/内存配额进行配置，如请求值和限制值
- **新增** 支持对工作负载的生命周期、健康检查、环境变量、数据存储、安全设置进行配置
- **新增** 支持添加工作负载标签与注解
- **新增** 工作负载支持多种网络访问，包含端口映射、主机网络、不可访问三种访问方式
- **新增** 支持编辑工作负载 YAML
- **新增** 支持更新工作负载，包含容器规格信息、访问配置、部署配置等

- **新增** 支持对工作负载进行重启/停止操作
- **新增** 支持查看工作负载的事件信息
- **新增** 支持将应用以镜像的方式下发至边缘节点组进行批量部署
- **新增** 支持创建消息路由，通过配置路由规则，让消息按照规定的路径转发至对应端点
- **新增** 支持 Modbus 协议终端设备接入平台并可视化管理
- **新增** 支持终端设备与已注册边缘节点绑定并支持解绑
- **新增** 支持添加、修改、删除设备孪生属性
- **新增** 通过编辑孪生属性，更改边缘节点上保存的设备孪生信息中的期待值，设备可以通过和孪生信息同步更改自身状态
- **新增** 支持为设备添加标签，可根据多个标签键值对查询符合条件的设备
- **新增** 支持设置设备访问配置，获取设备数据
- **新增** 支持删除终端设备

## 离线升级云边协同模块

本页说明[下载云边协同模块](#)后，应该如何安装或升级。

!!! info

下述命令或脚本内出现的 `__kant__` 字样是云边协同模块的内部开发代号。

### 从下载的安装包中加载镜像

您可以根据下面两种方式之一加载镜像，当环境中存在镜像仓库时，建议选择 `chart-syncer` 同步镜像到镜像仓库，该方法更加高效便捷。

## 使用 chart-syncer 同步镜像

使用 chart-syncer 可以将您下载的安装包中的 Chart 及其依赖的镜像包上传至安装器部署 DCE 时使用的镜像仓库和 Helm 仓库。

首先找到一台能够连接镜像仓库和 Helm 仓库的节点（如火种节点），在节点上创建 load-image.yaml 配置文件，填入镜像仓库和 Helm 仓库等配置信息。

### 1. 创建 load-image.yaml

!!! note

该 YAML 文件中的各项参数均为必填项。

=== “已添加 Helm repo”

若当前环境已安装 chart repo，chart-syncer 也支持将 chart 导出为 tgz 文件。

```
```yaml title="load-image.yaml"
source:
  intermediateBundlesPath: kant # (1)!
target:
  containerRegistry: 10.16.10.111 # (2)!
  containerRepository: release.daocloud.io/kant # (3)!
repo:
  kind: HARBOR # (4)!
  url: http://10.16.10.111/chartrepo/release.daocloud.io # (5)!
  auth:
    username: "admin" # (6)!
    password: "Harbor12345" # (7)!
containers:
  auth:
    username: "admin" # (8)!
    password: "Harbor12345" # (9)!
```
```

1. 使用 chart-syncer 之后 .tar.gz 包所在的路径
2. 镜像仓库地址
3. 镜像仓库路径
4. Helm Chart 仓库类别
5. Helm 仓库地址
6. 镜像仓库用户名
7. 镜像仓库密码

8. Helm 仓库用户名

9. Helm 仓库密码

### === “未添加 Helm repo”

若当前节点上未添加 helm repo，chart-syncer 也支持将 chart 导出为 tgz 文件，并存放在指定路径。

```
```yaml title="load-image.yaml"
source:
  intermediateBundlesPath: kant # (1)!
target:
  containerRegistry: 10.16.10.111 # (2)!
  containerRepository: release.daocloud.io/kant # (3)!
repo:
  kind: LOCAL
  path: ./local-repo # (4)!
containers:
  auth:
    username: "admin" # (5)!
    password: "Harbor12345" # (6)!
```
```

1. 使用 chart-syncer 之后 .tar.gz 包所在的路径
2. 镜像仓库 url
3. 镜像仓库路径
4. chart 本地路径
5. 镜像仓库用户名
6. 镜像仓库密码

### 2. 执行同步镜像命令。

```
charts-syncer sync --config load-image.yaml
```

## 使用 Docker 或 containerd 加载镜像

解压并加载镜像文件。

### 1. 解压 tar 压缩包。

```
tar xvf kant.bundle.tar
```

解压成功后会得到 3 个文件：

- hints.yaml
- images.tar
- original-chart



## 2. 从本地加载镜像到 Docker 或 containerd。

```
=== "Docker"
```shell
    docker load -i images.tar
```

=== "containerd"
```shell
    ctr -n k8s.io image import images.tar
```
```

### !!! note

每个 node 都需要做 Docker 或 containerd 加载镜像操作，加载完成后需要 tag 镜像，保持 Registry、Repository 与安装时一致。

## 升级

有两种升级方式。您可以根据前置操作，选择对应的升级方案：

### === “通过 helm repo 升级”

#### 1. 检查云边协同 Helm 仓库是否存在。

```
```shell
helm repo list | grep kant-release
```
```

若返回结果为空或如下提示，则进行下一步；反之则跳过下一步。

```
```none
Error: no repositories to show
```
```

#### 1. 添加云边协同的 Helm 仓库。

```
```shell
helm repo add kant-release http://{harbor url}/chartrepo/{project}
```
```

#### 1. 更新云边协同的 Helm 仓库。

```
```shell
helm repo update kant-release
```
```

1. 选择您想安装的云边协同版本（建议安装最新版本）。

```
```shell
helm search repo kant-release/kant --versions
```
```

输出类似于：

```
```none
NAME                                CHART VERSION  APP VERSION  DESCRIPTION
kant-release/kant  0.5.0          v0.5.0       A Helm chart for kant
...
```
```

1. 备份 `--set` 参数。

在升级云边协同版本之前，建议您执行如下命令，备份老版本的 `--set` 参数。

```
```shell
helm get values kant -n kant-system -o yaml > bak.yaml
```
```

1. 更新 kant crds

```
```shell
helm pull kant-release/kant --version 0.5.0 && tar -zxf kant-0.5.0.tgz
kubectl apply -f kant/crds
```
```

1. 执行 `helm upgrade`。

升级前建议您覆盖 bak.yaml 中的 `global.imageRegistry` 字段为当前使用的镜像仓库地址。

```
```shell
export imageRegistry={你的镜像仓库}
```

```shell
helm upgrade kant-release kant-release/kant \
  -n kant-system \
  -f ./bak.yaml \
  --set global.imageRegistry=$imageRegistry \
  --version 0.5.0
```
```

### === “通过 Chart 包升级”

#### 1. 备份 `--set` 参数。

在升级云边协同版本之前，建议您执行如下命令，备份老版本的 `--set` 参数。

```
```shell
helm get values kant -n kant-system -o yaml > bak.yaml
```
```

#### 1. 更新 kant crds

```
```shell
kubectrl apply -f ./crds
```
```

#### 1. 执行 `helm upgrade`。

升级前建议您覆盖 bak.yaml 中的 `global.imageRegistry` 为当前使用的镜像仓库地址。

```
```shell
export imageRegistry={你的镜像仓库}
```

```shell
helm upgrade kant-release . \
  -n kant-system \
  -f ./bak.yaml \
  --set global.imageRegistry=$imageRegistry
```
```

## 常见问题

本页面列出了一些在云边协同模块中可能遇到的问题，为您提供便利的故障排除解决办法。

#### 1. 容器下载镜像发生 TLS 证书认证相关问题

如果遇到上述问题，需要过滤私有 https 服务证书认证，不同运行时参考方案如下：

##### **containerd:**

配置 `/etc/containerd/config.toml` 文件，下列内容为参考配置，具体可以按需求定制化

设置。

```
...
[plugins."io.containerd.grpc.v1.cri".registry.configs]
  [plugins."io.containerd.grpc.v1.cri".registry.configs."reg.xxx.cn".tls]
    insecure_skip_verify = true
```

#### Docker:

配置 `/etc/docker/daemon.json` 文件，下列内容为参考配置，具体可以按需求定制化设置。

置。

```
{
  ...
  "insecure-registries": [
    "0.0.0.0/0"
  ],
  ...
}
```

## 2. CNI 无法正确使用

解决方案的操作步骤如下：

### 1. 下载 CNI 插件包并解压

访问 [containernetworking\\_\\_release](#) 页面，下载

`cri-plugins-{os}-{arch}-{version}.tar.gz` 的包，此包会包含 CNI 工具，使用命令

将压缩包中的二进制文件解压到 `/opt/cni/bin` 目录。

```
mkdir -p /opt/cni/bin
tar Cxzf /opt/cni/bin cni-plugins-linux-amd64-v1.1.1.tgz
```

### 2. 创建默认 CNI 配置文件

创建默认 CNI 配置文件到 `/etc/cni/net.d/` 目录下，文件名如：`10-mynet.conf`，

具体配置参考如下：

```
{
  "cniVersion": "1.0.0",
  "name": "mynet",
  "plugins": [
    {
      "type": "bridge",
```

```

    "bridge": "cni0",
    "isGateway": true,
    "ipMasq": true,
    "promiscMode": true,
    "ipam": {
      "type": "host-local",
      "ranges": [
        [
          {
            "subnet": "10.88.0.0/16"
          }
        ],
        [
          {
            "subnet": "2001:db8:4860::/64"
          }
        ]
      ],
      "routes": [
        { "dst": "0.0.0.0/0" },
        { "dst": "::/0" }
      ]
    }
  },
  {
    "type": "portmap",
    "capabilities": { "portMappings": true }
  }
]
}

```

## 创建专有边缘单元

边缘单元定义：指容器运行所需计算资源的集合，包括云端 Master 和边端工作节点 Node。

边缘单元跟 K8S 中的集群是同一概念，但组网方式有所不同，Master 节点部署在云端，一个云端 master (主备多台)对应一个边缘集群。

边缘单元业务：给指定工作集群安装 KubeEdge 云端套件 (CloudCore、ControllerManager)，并且对其进行全生命周期管理。KubeEdge 是一个开源系统，将原生的容器化应用程序编排功能扩展到边缘节点。

- CloudCore：KubeEdge 云端核心组件。

- ControllerManager：KubeEdge CRD 扩展，目前应用于边缘应用和边缘节点组。

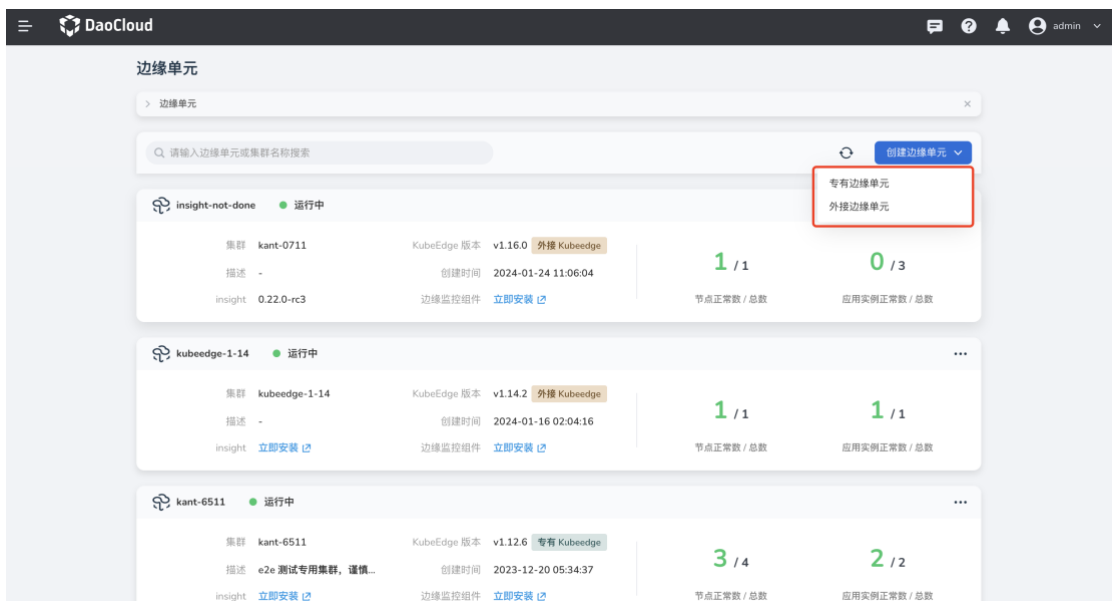
DCE 5.0 云边协同支持两种边缘单元：

- **专有边缘单元** 给指定工作集群安装 KubeEdge 云端套件（CloudCore、ControllerManager），并且对其进行全生命周期管理。
- **外接边缘单元** 指的是将企业系统中已安装的 KubeEdge 接入到 DCE 5.0 云边协同中进行统一管理。参见 [创建外接边缘单元](#)

## 操作步骤

下文说明创建专有边缘单元的步骤：

1. 选择左侧导航栏的 **云边协同**，进入边缘单元列表页面，点击页面右上角的 **创建边缘单元** 按钮，在下拉列表中选择 **创建专有边缘单元**；



### 创建专有边缘单元

2. 填写基础信息；

- 边缘单元名称：小写字母、数字、中划线 (-)、点 (.) 的组合，不能有连续符号；以字母或数字为开头、结尾；最多包含 253 个字符。

- 集群：运行边缘单元控制面的集群。
- KubeEdge 版本：KubeEdge 开源系统发布的某一个版本，用于将容器化应用程序编排功能扩展到边缘的主机，它基于 kubernetes 构建，并为网络应用程序提供基础架构支持。
- MQTT 服务：是否开启边缘节点中默认的 KubeEdge mqtt 服务 (mosquitto)，主要作用于消息路由和设备 mapper 通信。
- 描述：边缘单元描述信息。

← 创建边缘单元

1

2

基本信息 高级设置

边缘单元名称 \*

edgeunit-001

最长 253 个字符，只能是小写字母、数字、中划线(-)、点(.)的组合，且不能有连续符号，以字母或数字为开头、结尾。

集群 \*

test-cluster-hou

创建集群

KubeEdge 版本 \*

v1.16.1

节点规模

请选择

MQTT 服务

启用

描述

0~253 个字符

取消 下一步

## 基本信息

### 3. 填写高级设置

!!! note

如果您是使用在线安装方式，则只需要填写访问配置，如果使用的是离线安装，则还需设置组件仓库信息。

## 访问设置

KubeEdge 云端组件的访问设置，边缘节点通过此设置与云端建立连接。

- 通讯协议：云边信令通道通信协议，云边网络经常不稳定时，推荐使用 QUIC 协议。

- 访问地址：KubeEdge 云端组件 CloudCore 的访问地址，需要能被边缘节点访问，边缘节点通过该地址与云端建立连接。
- 端口：云端 CloudCore 默认给边端开放 NodePort 端口，如有冲突，请修改。
  - WebSocketPort：访问协议 WebSocket 端口，默认 30000。
  - QUICPort：访问协议 QUIC 端口，默认 30001。
  - HTTPServerPort：HTTP 服务端口，默认 30002。
  - CloudStreamPort：云端流处理接口端口，默认 30003。
  - TunnelPort：边缘节点业务数据通道端口，默认 30004。

### 访问配置

4. 完成以上信息配置后，点击 **确定** 按钮，完成边缘单元创建，自动返回边缘单元列表。

下一步：[管理边缘单元](#)

## 创建外接边缘单元

**外接边缘单元** 指的是将企业系统中已安装的 KubeEdge 接入到 DCE 5.0 云边协同中进行



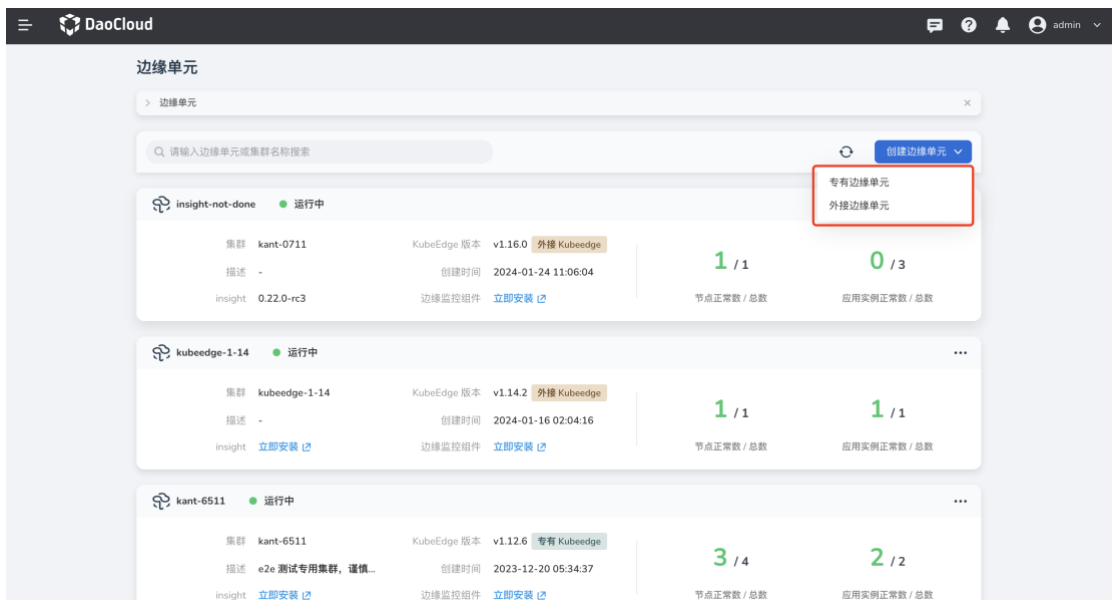
统一管理。

KubeEdge：是一个开源系统，将原生的容器化应用程序编排功能扩展到边缘节点。

- CloudCore：KubeEdge 云端核心组件。
- ControllerManager：KubeEdge CRD 扩展，目前应用于边缘应用和边缘节点组。

下文说明创建外接边缘单元的步骤：

1. 选择左侧导航栏的 **云边协同**，进入边缘单元列表页面，点击页面右上角的 **创建边缘单元** 按钮，在下拉列表中选择 **创建外接边缘单元**；



### 创建专有边缘单元

2. 填写基础信息；
  - 边缘单元名称：小写字母、数字、中划线 (-)、点 (.) 的组合，不能有连续符号；以字母或数字为开头、结尾；最多包含 253 个字符。
  - 集群：运行边缘单元控制面的集群。
  - KubeEdge 版本：KubeEdge 开源系统发布的某一个版本，用于将容器化应用程序编排功能扩展到边缘的主机，它基于 kubernetes 构建，并为网络应用程序提供基础架构支持。

- 描述：边缘单元描述信息。

### !!! note

如果您选择的集群是一个单节点集群，请确保 master 节点支持调度，即已删除污点 NoSchedule，避免系统组件安装失败。

← 创建边缘单元

1 基本信息

2 高级设置

边缘单元名称 \*

edgeunit-001

最长 253 个字符，只能是小写字母、数字、中划线(-)、点(.)的组合，且不能有连续符号，以字母或数字为开头、结尾。

集群 \*

test-cluster-hou

创建集群

KubeEdge 版本 \*

v1.16.1

e.g v1.0.0

节点规模

请选择

描述

0~253 个字符

取消

下一步

## 基本信息

### 3. 填写高级设置

### !!! note

如果您是使用在线安装方式，则只需要填写访问配置，如果使用的是离线安装，则还需设置组件仓库信息。

## 访问设置

KubeEdge 云端组件的访问设置，边缘节点通过此设置与云端建立连接。

- 通讯协议：云边信令通道通信协议，云边网络经常不稳定时，推荐使用 QUIC 协议。
- 访问地址：KubeEdge 云端组件 CloudCore 的访问地址，需要能被边缘节点访问，边缘节点通过该地址与云端建立连接。
- 端口：云端 CloudCore 默认给边端开放 NodePort 端口，如有冲突，请修改。
  - WebSocketPort：访问协议 WebSocket 端口，默认 10000。
  - QUICPort：访问协议 QUIC 端口，默认 10001。

- HTTPServerPort: HTTP 服务端口, 默认 10002。
- CloudStreamPort: 云端流处理接口端口, 默认 10003。
- TunnelPort: 边缘节点业务数据通道端口, 默认 10004。

#### 访问配置

4. 完成以上信息配置后, 点击 **确定** 按钮, 完成边缘单元创建, 自动返回边缘单元列表。

下一步: [管理边缘单元](#)

## 管理边缘单元

### 边缘单元状态查看

边缘单元有如下状态:

- 执行中: 边缘单元正在创建, 或编辑边缘单元信息更新中, 此时边缘单元名称不可点击, 不可进入边缘单元操作。
- 运行中: 边缘单元正常运行。
- 失败: 边缘单元创建失败或运行异常, 鼠标悬浮右侧图标可以查看异常信息

- 删除中：删除边缘单元进行中，此时边缘单元名称、编辑、删除按钮不可点击。
- 等待中：边缘单元正在等待创建或更新操作的执行。


### 边缘单元资源统计：

- 节点正常数/总数：节点状态为健康的节点数量/总数
- 应用实例正常数/总数；工作负载运行正常数量/总数

### 边缘单元列表

#### 边缘单元列表


## 编辑边缘单元

在边缘单元列表的右侧，点击  按钮，在弹出菜单中选择 **编辑**，可以对边缘单元的基础信息、组件仓库设置、访问设置进行编辑操作。

### 编辑边缘单元

#### 编辑边缘单元

## 删除边缘单元

在边缘单元列表的右侧，点击  按钮，在弹出菜单中选择 **删除**。

!!! note

- 删除边缘单元，系统会自动删除边缘单元下的终端设备、消息端点和消息路由资源。
- 如果边缘单元下有创建边缘节点和工作负载资源，需要用户先手动删除。
- 点击边缘节点和工作负载，用户可以快速跳转到边缘节点列表和工作负载列表页。

### 删除边缘单元

#### 删除边缘单元

## 边缘单元概览

点击列表中边缘单元名称，进入边缘单元概览页面，可以查看基本信息和资源状态信息。更

多管理操作点击左侧菜单栏进入对应菜单进行管理操作。

## 边缘单元概览

边缘单元概览

# 边缘资源监控

本文介绍如何在边缘单元开启监控能力,以实时查看边缘节点和边缘工作负载的状态及其资源使用情况。

边缘监控依赖 insight-agent 监控组件和边缘监控组件,下文介绍如何安装对应组件。

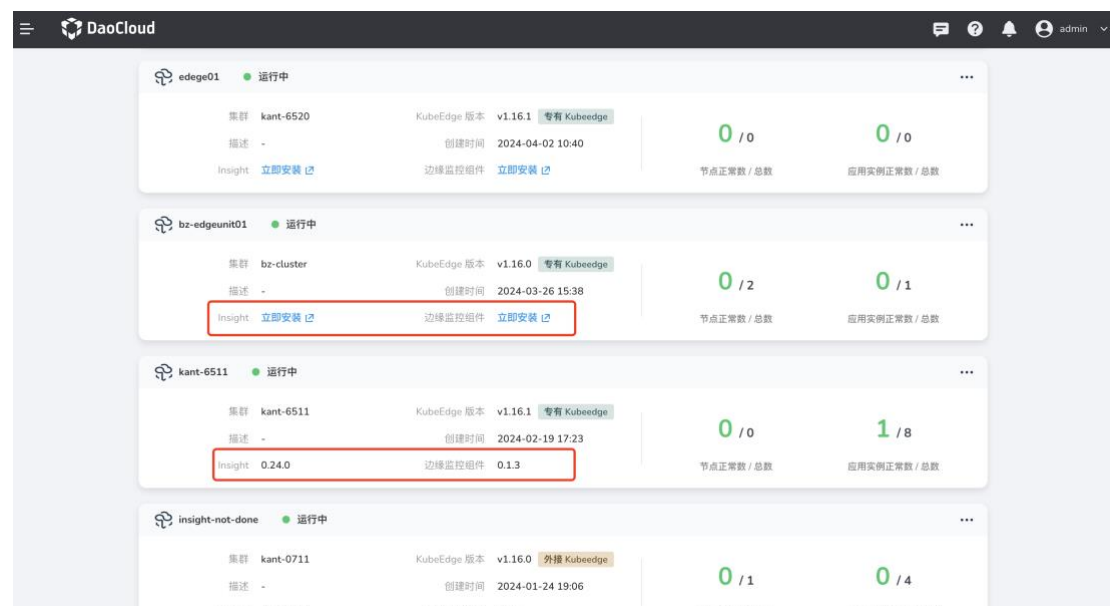
!!! note

请先安装 insight-agent 监控组件,边缘监控组件依赖 Insight 相关插件。

## 检查是否安装监控组件

选择左侧导航栏的 **云边协同**,进入边缘单元列表页面,查看边缘单元条目中 Insight 和边缘组件显示信息:

- 显示版本号,代表监控组件已安装;
- 显示立即安装,代表监控组件未安装,点击 **立即安装** 按钮,跳转到对应的 Helm 应用详情页。



监控组件状态

## 安装 insight-agent

insight-agent 是集群观测数据采集的插件，支持对指标、链路、日志数据的统一观测，安装流程参考 [在线安装 insight-agent](#)。

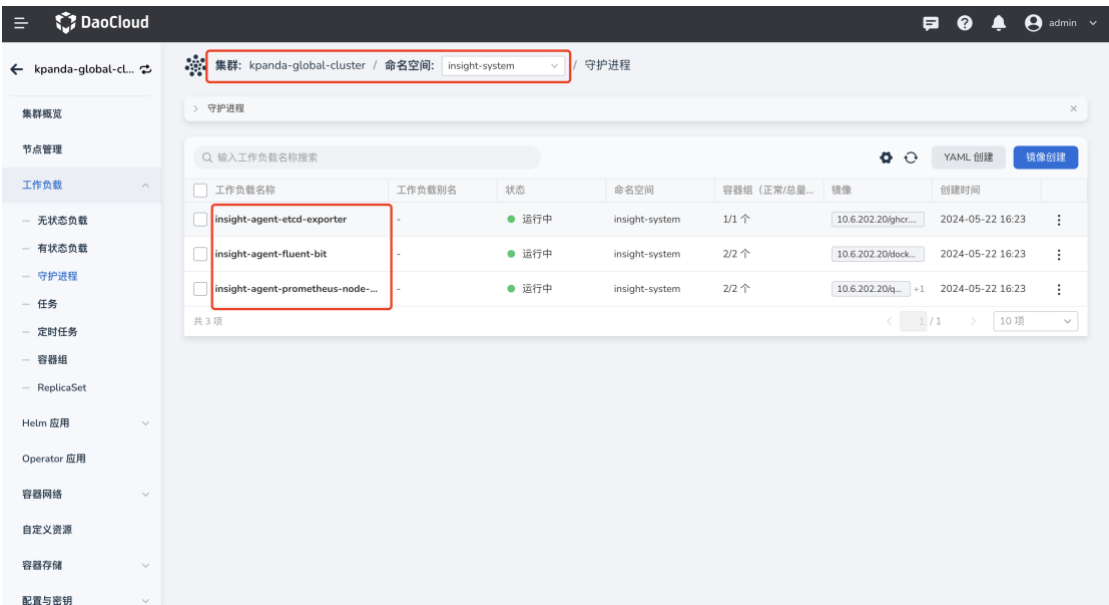
为避免 insight daemonset 组件被调度到边缘节点，需要给 **全局服务集群** 的 insight-system 命名空间下各 DaemonSet 组件加上以下亲和性设置。

nodeAffinity:

requiredDuringSchedulingIgnoredDuringExecution:

nodeSelectorTerms:

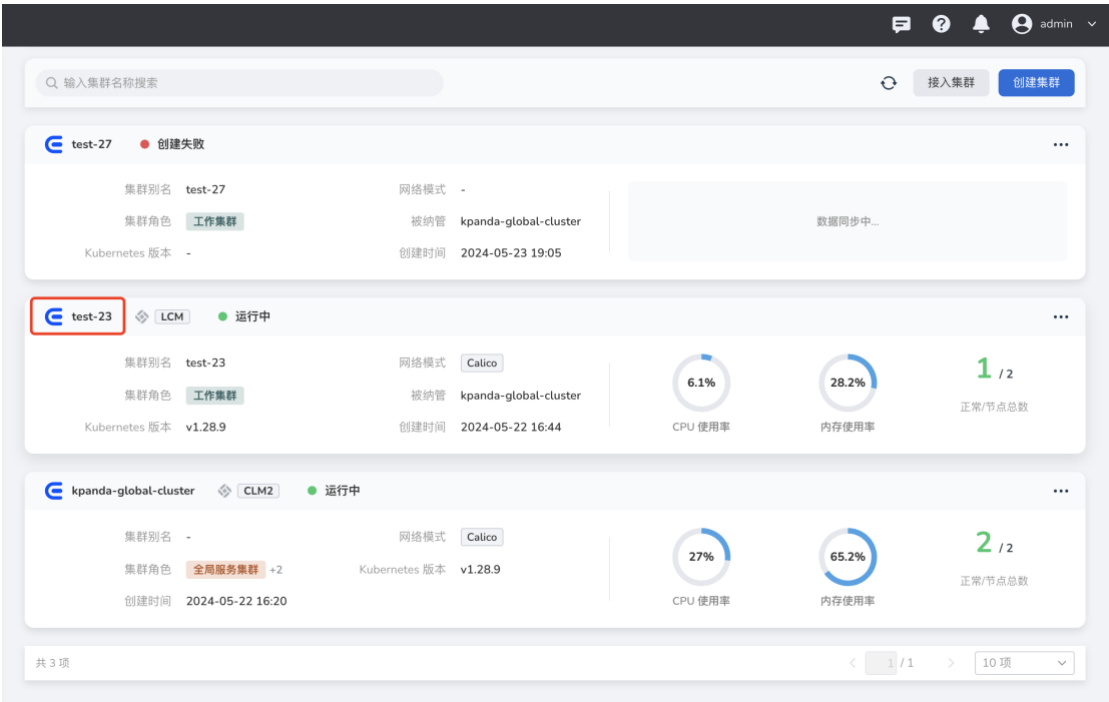
- matchExpressions:
  - key: node-role.kubernetes.io/edge
    - operator: DoesNotExist



insight daemonset 组件

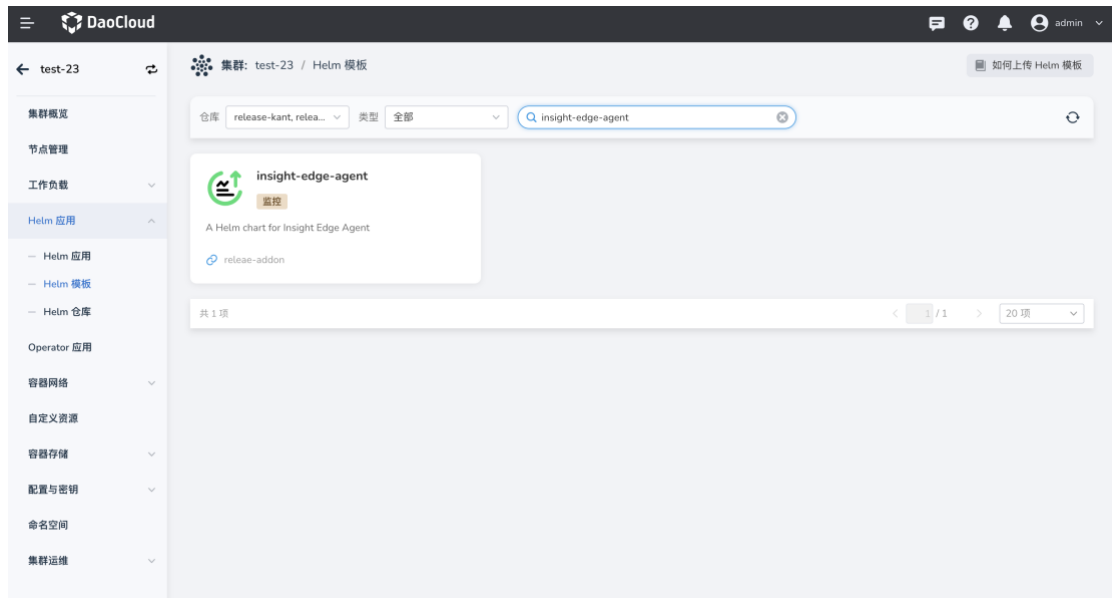
## 安装边缘监控组件

1. 选择左侧导航栏的 **容器管理** -> **集群列表**，进入集群列表页面，点击集群名称，进入集群详情页。



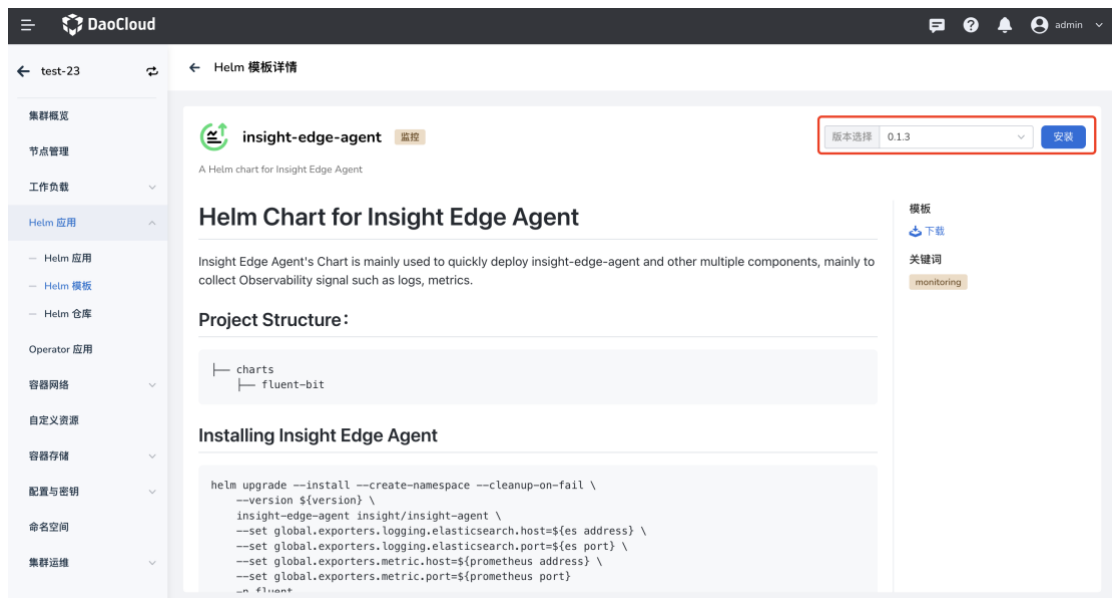
集群列表

2. 选择左侧菜单 **Helm 应用** -> **Helm 模板** , 在 addon 仓库下找到 insight-edge-agent 插件, 点击进入模版详情页。



### insight-edge-agent 模版

3. 在页面右上角选择 insight-edge-agent 版本, 点击 **安装** 按钮, 进入 insight-edge-agent 安装页面。



### insight-edge-agent 安装

4. 在表单中分别填写全局服务集群和边缘工作集群中对应数据存储组件的地址, 确认填写



的信息无误后，点击 **确定**。

为实现对多集群观测数据的统一存储、查询，边缘集群需要将采集的观测数据上报给全局服务集群进行统一存储。

- 在 **边缘工作集群** 获取 Prometheus 服务地址和端口，目前仅支持 NodePort 访问方式，故地址对应的是工作集群控制节点地址

在 insight-system 命名空间下，输入如下命令，找到 9090 端口映射的服务端口

```
kubectl get svc -n insight-system | grep prometheus
```

!!! note

如果 Prometheus 服务默认访问方式为 ClusterIP，请修改为 NodePort 访问方式。

```
[root@controller-node-1 ~]# kubectl get svc -n insight-system | grep prometheus
insight-agent-kube-prometheus-prometheus      NodePort    10.233.57.31    <none>    9090:31774/TCP    5d2h
insight-agent-prometheus-usackoo-exporter      ClusterIP   10.233.49.243   <none>    9125/TCP          5d2h
insight-agent-prometheus-node-exporter         ClusterIP   10.233.3.23     <none>    9188/TCP          5d2h
prometheus-operated                            ClusterIP   None            <none>    9090/TCP          5d2h
[root@controller-node-1 ~]#
```

获取 Prometheus 服务端口

- 在 **全局服务集群** 获取 Elasticsearch 服务地址和端口，目前仅支持 NodePort 访问方式，故地址对应的是集群控制节点地址。

在 insight-system 或 mcamel-system 命名空间下，输入如下命令，找到 9200 端口映射的服务端口：

```
kubectl get service -n mcamel-system | grep es
```

```
[root@master1 ~]# kubectl get service -n mcamel-system | grep es
mcamel-common-es-cluster-masters-es-data      ClusterIP   None            <none>    9288/TCP          5d1h
mcamel-common-es-cluster-masters-es-http      NodePort    10.233.48.288   <none>    9288:31884/TCP   5d1h
mcamel-common-es-cluster-masters-es-inverted-index-http ClusterIP   10.233.24.239   <none>    9288/TCP          5d1h
mcamel-common-es-cluster-masters-es-transport ClusterIP   None            <none>    9388/TCP          5d1h
mcamel-common-es-cluster-masters-kb-http      ClusterIP   10.233.2.189    <none>    5681/TCP          5d1h
mcamel-common-es-cluster-masters-prometheus-exporter ClusterIP   10.233.53.26    <none>    9114/TCP          5d1h
[root@master1 ~]#
```

获取 Elasticsearch 服务端口

!!! note

其他参数默认不用修改。

test-23

安装 - insight-edge-agent

insight-edge-agent 监控

A Helm chart for Insight Edge Agent

名称 insight-edge-agent

命名空间 fluent

版本 0.1.3

失败删除 关闭

就绪等待 关闭

详细日志 关闭

参数配置

表单 YAML 变化

Configure following parameters to upload metric/log/tracing data to global cluster. Please make sure all insight components are running healthily in global cluster first!

Global

Please make sure below addresses are reachable in this cluster. Need detail about how to get those parameters, please refer to: <https://docs.daocloud.io/insight/DEUserGuide/01quickstart/gethosturl>.

Metric Settings

Configure to upload metric data, use global cluster vminsert service address by default, like http://10.0.0.1:8480/insert...

scheme http

host

host不能为空

port 8480

path /insert/0/prometheus

Logging Settings

Configure to upload log use elasticsearch or kafka as the data output type

output elasticsearch

Elasticsearch Settings

Configure to Elasticsearch exporter, use by Fluentbit

host

host不能为空

port 9200

user elastic

password dangerous

Audit Log Settings

Configure to upload cluster audit log, use global cluster opentelemetry-collector service with port 8006 by default, lik...

Kubernetes Audit

enabled Off

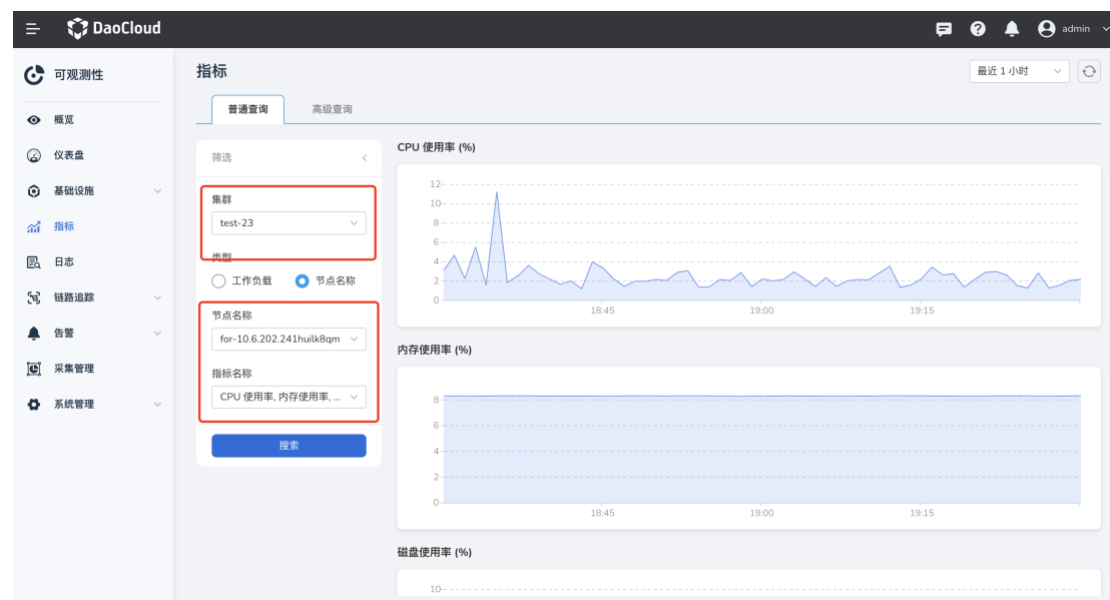
取消 确定

## 安装 insight-edge-agent

- 系统将自动返回 Helm 应用列表，当应用 **insight-edge-agent** 的状态从 **未就绪** 变为 **已部署**，且所有的组件状态为 **运行中** 时，则表示安装成功。

## 查看边缘监控数据

边缘组件安装部署成功后，等待一段时间，可从左侧导航栏进入 **可观测性** 模块，查看边缘资源的数据。



查看边缘资源监控数据

## 边缘节点概述

边缘节点是容器集群组成的基本元素，既可以是云主机，也可以是物理机，用于运行容器化应用的载体，边缘应用将以 Pod 的形式在节点上运行。

管理边缘节点，需要完成如下操作步骤：

1. 准备边缘节点并完成节点环境配置，边缘节点需要满足一定的规格要求，具体请参见[边缘节点接入要求](#)。
2. 在云边协同模块创建接入指南，获取边缘节点配置文件和安装程序，具体请参见[创建接入配置](#)。
3. 根据安装指南，完成边缘节点的接入纳管，具体请参见[接入边缘节点](#)。

使用流程如下：

graph TB

```
start1([开始]) --> require[了解边缘节点接入要求] --> config[配置边缘节点] --> oam[运维边缘节点] --> end1([结束])
```

```
classDef plain fill: #ddd,stroke:#fff,stroke-width:1px,color:#000;
classDef k8s fill: #326ce5,stroke:#fff,stroke-width:1px,color:#fff;
classDef cluster fill: #fff,stroke:#bbb,stroke-width:1px,color:#326ce5;
```

```
class start1,end1 k8s;
class require,config,oam cluster
```

```
click require "https://docs.daocloud.io/kant/user-guide/node/join-rqmt.html"
click config "https://docs.daocloud.io/kant/user-guide/node/access-guide.html"
click oam "https://docs.daocloud.io/kant/user-guide/node/manage-node.html"
```

## 边缘节点接入要求

边缘节点需要满足下表的规格要求。

### 操作系统 OS

- x86\_64 架构

Ubuntu LTS (Xenial Xerus / Bionic Beaver / Jammy Jellyfish / Focal Fossa)、CentOS、EulerOS、

RHEL、银河麒麟、中兴新支点、中标麒麟、openEuler、Rocky Linux

- aarch64 (arm64) 架构

Ubuntu LTS (Bionic Beaver)、CentOS、EulerOS、openEuler、Rocky Linux

### 内存

边缘软件常规开销约 150MB，具体视边缘管理压力适当增加。

为保证业务的正常运行，建议边缘节点的内存大于 256MB。

## CPU

不小于 1 核

## 硬盘

不小于 1GB

## 容器运行时

容器运行时支持 Docker 或 containerd。

- Docker 版本必须高于 v19.0，推荐使用 v19.03.6 版本。

须知：如果您安装的 KubeEdge 版本高于或等于 v1.14 且云端 Kubernetes 版本高于 v1.24，

除 Docker 之外，还需要安装 CRI-Dockerd 。

- 如果您安装的 KubeEdge 版本高于 v1.12.0，推荐安装 containerd，安装流程参见[边缘](#)

[节点安装容器引擎](#)。

## glibc

版本必须高于 2.17。

ubuntu 查看版本示例：

```
ldd --version
```

## 端口

边缘节点需要使用如下端口，请确保这些端口能够正常使用。

- 1883: 边缘节点内置 MQTT broker 监听端口, 并需要开放该端口。

## 时间同步

边缘节点时间需要与云端服务器时间保持一致, 建议 UTC 标准时间。

## 组件调度

边缘不支持运行 Calico、KubeProxy、Insight 等组件, 为避免云上 Daemonset 应用调度到

边缘, 导致节点状态异常, 需添加如下注解进行屏蔽。

```
nodeAffinity:
  requiredDuringSchedulingIgnoredDuringExecution:
    nodeSelectorTerms:
      - matchExpressions:
          - key: node-role.kubernetes.io/edge
            operator: DoesNotExist
```

## 边缘节点安装容器引擎

边缘节点在接入系统之前, 需要先配置节点环境, 其中就包括安装容器引擎。本文介绍如何

安装容器运行时组件 containerd。

!!! note

- 如果您安装的 KubeEdge 版本高于 v1.12.0, 推荐安装 containerd。
- KubeEdge v1.15.0 以及以上版本, 请安装 v1.6.0 或更高版本的 containerd。

## 安装 containerd

边缘节点接入需要依赖 CNI 插件, 所以建议直接安装带有 CNI 插件的 containerd, 操作步骤如下:

1. 下载 containerd 安装包并上传到边缘节点, 前往[下载地址](#), 根据边缘节点操作系统以及 CPU 架构选择对应版本安装包。

## 2. 将安装包解压到根目录。

```
tar xvfz {安装包名称} -C /
```

## 3. 生成 containerd 配置文件。

!!! note

注意修改配置文件的 sandbox 镜像，国内可能拉不到 k8s 仓库的镜像，可以换成 DaoCloud 的镜像仓库：m.daocloud.io/k8s.gcr.io/pause:3.8。

```
mkdir /etc/containerd
```

```
containerd config default > /etc/containerd/config.toml
```

## 4. 启动 containerd。

```
systemctl start containerd && systemctl enable containerd
```

## 5. 验证 containerd 是否安装成功并且成功运行。

```
systemctl status containerd
```

# 安装 nerdctl 工具（可选）

建议安装 nerdctl 命令行工具，方便在节点上对容器进行运维调试，操作步骤如下：

## 1. 下载 nerdctl 安装包并上传到边缘节点，前往[下载](#)。

!!! note

请根据实际操作系统以及 CPU 架构选择对应安装包，请安装 v1.7.0 或更高版本。

## 2. 解压安装包，并将二进制文件拷贝至 /usr/local/bin 目录下。

```
tar -zxvf nerdctl-1.7.6-linux-amd64.tar.gz
```

```
cd nerdctl-1.7.6-linux-amd64
```

```
cp nerdctl /usr/local/bin
```

## 3. 验证 containerd 是否安装成功。

使用如下命令查看 Server 版本号，如果能正常显示，说明 nerdctl 已经成功安装。

```
nerdctl version
```

# 创建接入配置

相同类型的边缘节点能够设定相同的边缘节点配置，通过创建接入配置，获取边缘节点配置

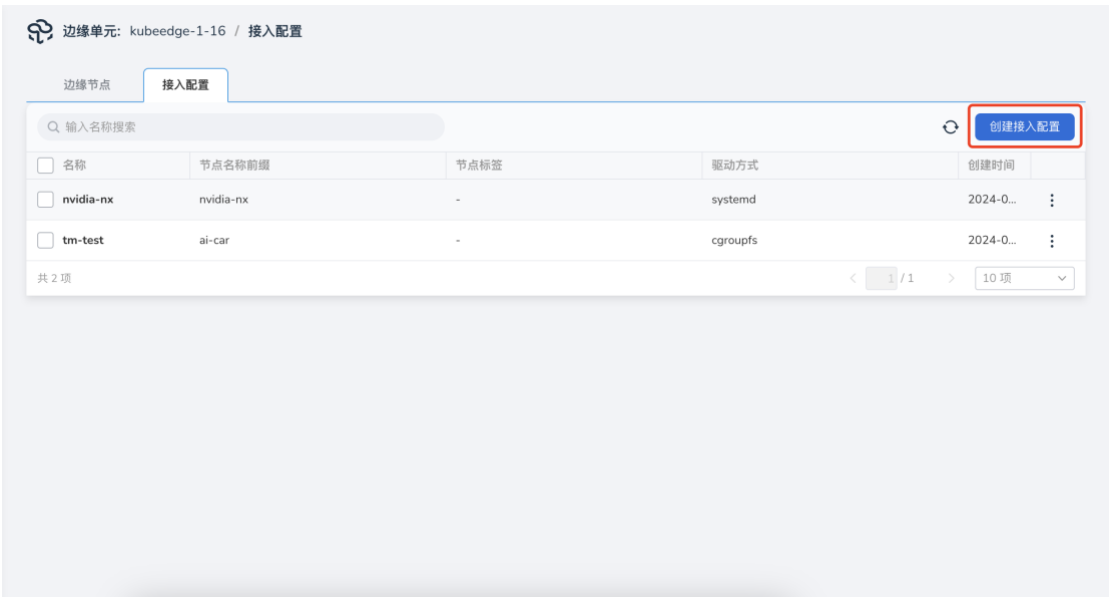
文件和安装程序。接入配置与边缘节点满足一对多的关系，提高管理效率的同时，节约了

运维成本。

下文说明创建接入配置的步骤和接入配置管理。

## 操作步骤

1. 左侧导航栏点击 **边缘节点** 菜单，进入页面，选择 **接入配置**，进入配置管理列表页，点击右上角 **创建接入配置** 按钮。



### 指南管理

2. 填写注册信息。
- 指南名称：指南名称不能为空，长度限制为 253 位。
  - 节点前缀：节点名称由“节点前缀-随机码”组成。
  - 驱动方式：控制组（CGroup）的驱动，用于对 Pod 和容器进行资源管理和资源配置，如 CPU 和内存资源的请求和限制。
  - CRI 服务地址：CRI Client 和 CRI Server 在本地进行通信的 socket 文件或者 TCP 地址， 例如 unix:///run/containerd/containerd.sock
  - KubeEdge 边端镜像仓库：存储 KubeEdge 组件 (Mosquitto、installation-package、



pause) 镜像仓库地址，如果边端镜像和云端镜像在一个镜像仓库，您可以

点击 **引用云端地址** 按钮，快速填写。

- 描述：接入指南描述信息。
- 标签：接入指南标签信息。

← 创建接入配置

配置名称 \*

最长 253 个字符，只能是小写字母、数字、中划线(-)、点(.)的组合，且不能有连续符号，以字母或数字为开头、结尾。

节点前缀 \*

最长 20 个字符，只能是小写字母、数字、中划线(-)、点(.)的组合，且不能有连续符号，以字母或数字为开头、结尾。

驱动方式 \*

cgroupfs

CRI 服务地址 \*

镜像仓库 \*

ⓘ 为保证边端组件安装包能正常拉取，须知：

1. 请确认相关组件存储在对应的镜像仓库；

2. 如果使用的是私有仓库，请到边缘节点手动登录。

引用云端地址

KubeEdge 边缘镜像仓库 \*

示例：docker.m.daocloud.io/kubeedge

取消 确定

## 创建接入配置

3. 完成信息填写后，点击 **确定** 按钮，完成接入配置创建。

## 后续操作

完成接入配置创建后，您可以进详情页查看 **接入配置**，并根据接入流程提示，完成对边缘节点的接入操作，具体请参见[节点接入指南](#)。

## 接入边缘节点

根据节点接入配置，获取安装文件和接入命令，在节点上安装边缘核心软件 EdgeCore，这样边缘节点就能与 DCE 5.0 云边协同建立连接，纳入平台管理。

上海道客网络科技有限公司

4

边缘节点初次接入时，自动安装最新版本的边缘核心软件 EdgeCore。

### !!! note

接入配置与实际的边缘节点机器是一对多的关系，一个接入配置的安装文件和接入命令可以在多台实际的边缘节点上使用。

## 前提条件

- 已经按要求准备好节点，并配置好节点环境，具体请参见[边缘节点接入要求](#)
- 已经创建好边缘节点接入指南，具体请参见[创建接入配置](#)

### !!! note

如果您是离线环境下接入异构节点，请先进行 Helm 应用多架构融合操作，操作流程参考[Helm 应用多架构和升级导入步骤](../../kpanda/user-guide/helm/multi-archi-helm.md)。



## 接入节点

## 操作步骤

1. 边缘节点列表 页面，点击 **接入节点** 按钮，右侧会弹出接入节点弹框。
2. 根据节点环境配置，选择对应的接入配置。
3. 点击 **下载文件** 按钮，跳转到下载中心，在下载列表中选择对应版本和架构的边端安装包：kantadm\_{版本}\_{架构}.tar.gz。建议选择最新版本。

下载中心 > DaoCloud Enterprise 5.0 > 下载子模块

云边协同

本页可下载云边协同各版本的离线安装包。

Tip

kant 是云边协同的内部开发代号。

下载

| 版本    | 架构     | 文件大小     | 安装包  | 校验文件  | 更新日期       | 备注           |
|-------|--------|----------|--|---|------------|--------------|
| 0.8.1 | AMD 64 | 106.3 MB | <a href="#">kant_0.8.1_amd64.tar</a>                 | <a href="#">kant_0.8.1_amd64_checksum.sha512sum</a>                 | 2024-02-02 | 云端 amd 安装包   |
| 0.8.1 | AMD 64 | 43.3 MB  | <a href="#">kantadm_installation_0.8.1_amd64.tar</a> | <a href="#">kantadm_installation_0.8.1_amd64_checksum.sha512sum</a> | 2024-02-02 | 边缘 amd 安装包   |
| 0.8.1 | ARM 64 | 41.9 MB  | <a href="#">kantadm_installation_0.8.1_arm64.tar</a> | <a href="#">kantadm_installation_0.8.1_arm64_checksum.sha512sum</a> | 2024-02-02 | 边缘 arm64 安装包 |
| 0.7.1 | AMD 64 | 105.2 MB | <a href="#">kant_0.7.1_amd64.tar</a>                 | <a href="#">kant_0.7.1_amd64_checksum.sha512sum</a>                 | 2024-01-04 | 云端 amd 安装包   |
| 0.7.1 | AMD 64 | 44.1 MB  | <a href="#">kantadm_installation_0.7.1_amd64.tar</a> | <a href="#">kantadm_installation_0.7.1_amd64_checksum.sha512sum</a> | 2024-01-04 | 边缘 amd 安装包   |

下载边端安装包

4. 将边端安装包拷贝到要接入的边缘节点，并解压。

解压命令：

```
tar -zxf kantadm_{version}_{tarch}.tar.gz
```

!!! note

将解压后的 kantadm 二进制文件放到 `/usr/local/bin` 目录下。

5. 通过 token 或证书方式，执行命令，接入节点。

token 安装

1. 接入指南界面，第三步点击 **token 安装** 页签，显示 token 安装步骤。

!!! note

安装命令的 token 24 小时内有效，如需长期有效的安装方式，请使用证书安装。

2. 接入节点，执行如下命令。

```
kantadm join --cloudcore-host=10.31.226.14 --websocket-port=30000 --node-prefix=
edge --token=b2d6bb5d9312c39ffac08ecfd5030bed006b8b67d0799d632d381f19fca
9e765.eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJleHAiOjE2OTQ2NTk3NDV9.
0sdaWbYSTURmAYmQwDn_zF7P9TwcRTSMhwPw6l87U7E --cgroup-driver=cg
roupfs --remote-runtime-endpoint= --version=v1.12.2 --batch-name=edge --edge-r
egistry=docker.m.daocloud.io/kubeedge --quic-port=30001 --http-port=30002 --stre
am-port=30003 --tunnel-port=30004 --labels=test=1,test1=1
```

证书安装

1. 接入节点界面，第三步点击 **证书安装** 页签，显示证书安装步骤。

2. 点击 **下载证书** 按钮，将证书下载到本地。

3. 保存证书，执行如下命令。

```
mkdir -p /etc/kant && mv ./cert.tar /etc/kant/cert.tar
```

4. 接入节点，执行如下命令。

```
kantadm join --cloudcore-host=10.2.129.13 --websocket-port=30000 --node-prefix=s
h --remote-runtime-endpoint=unix:///run/containerd/containerd.sock --cgroup-driver
=cgroupfs --version=v1.12.6 --batch-name=guide-test --edge-registry=docker.m.da
ocloud.io/kubeedge --quic-port=30001 --http-port=30002 --stream-port=30003 --tu
nnel-port=30004
```

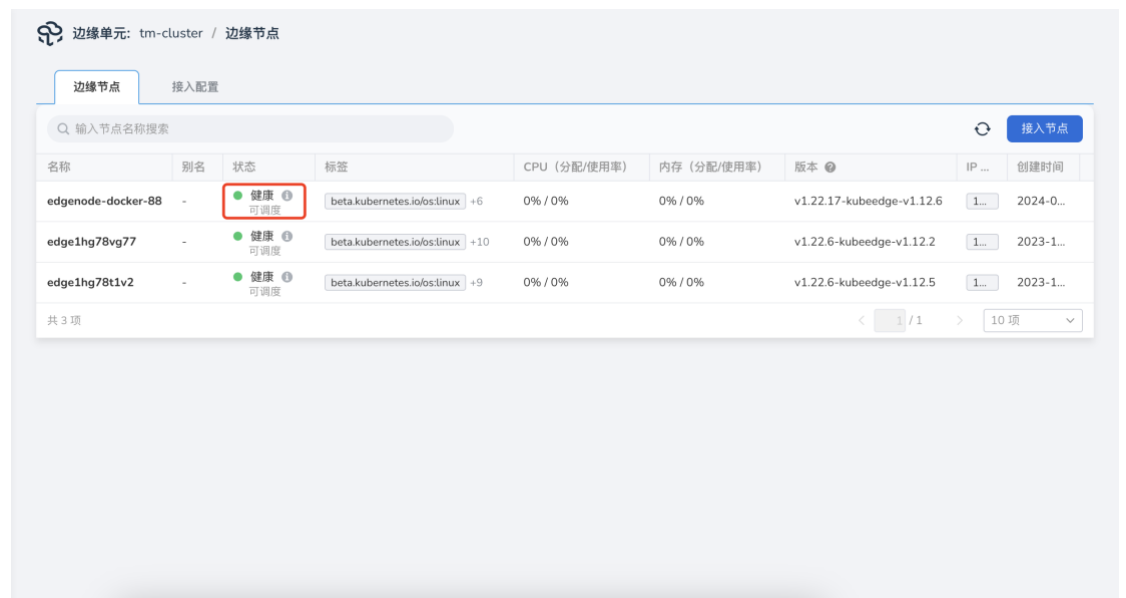
6. 验证边缘节点是否纳管成功。

1. 选择左侧导航栏的 **边缘计算** -> **云边协同**，进入边缘单元列表页面。

2. 点击边缘单元名称，进入边缘单元详情页。

3. 选择左侧导航栏的 **边缘资源** -> **边缘节点**，进入边缘节点列表页面。

4. 查看边缘节点的状态，当前状态为 **健康** 表示纳管成功。



| 名称                 | 别名 | 状态        | 标签                              | CPU (分配/使用率) | 内存 (分配/使用率) | 版本                        | IP ... | 创建时间      |
|--------------------|----|-----------|---------------------------------|--------------|-------------|---------------------------|--------|-----------|
| edgenode-docker-88 | -  | 健康<br>可调度 | beta.kubernetes.io/os:linux +6  | 0% / 0%      | 0% / 0%     | v1.22.17-kubeedge-v1.12.6 | 1...   | 2024-0... |
| edge1hg78vg77      | -  | 健康<br>可调度 | beta.kubernetes.io/os:linux +10 | 0% / 0%      | 0% / 0%     | v1.22.6-kubeedge-v1.12.2  | 1...   | 2023-1... |
| edge1hg78t1v2      | -  | 健康<br>可调度 | beta.kubernetes.io/os:linux +9  | 0% / 0%      | 0% / 0%     | v1.22.6-kubeedge-v1.12.5  | 1...   | 2023-1... |

共 3 项

边缘节点纳管成果

# 运维边缘节点

边缘节点纳管成功后，DCE 5.0 云边协同可以对边缘节点执行暂停调度、移除节点操作。

## 暂停调度

应用场景：当需要对节点进行运维操作，用户可以选择暂停调度节点，那么工作负载在节点暂停调度期间，将无法下发到此节点。

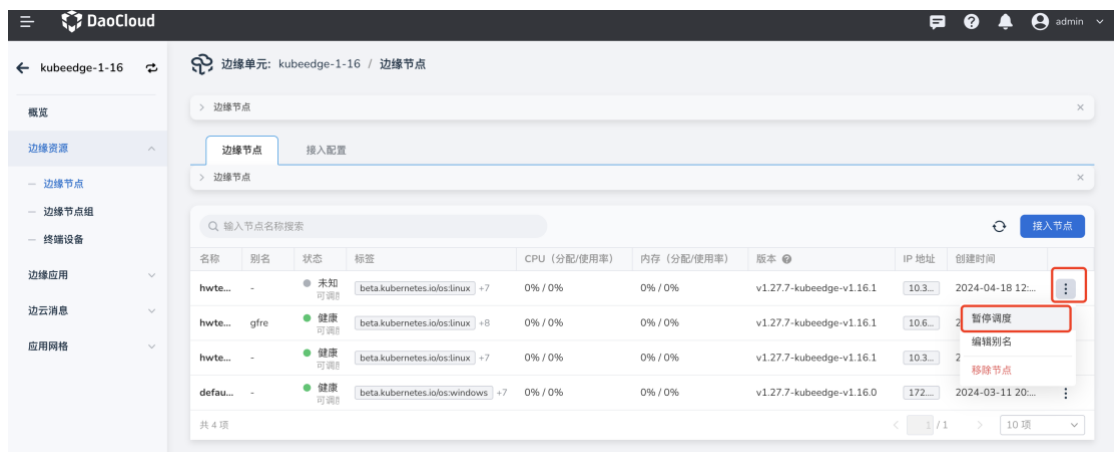
!!! note

已经部署到节点的工作负载不影响其正常运行。

操作步骤如下：

1. 进入边缘单元详情页，选择左侧菜单 **边缘资源** -> **边缘节点** 。
2. 在节点列表右侧，点击三个点 **⋮** -> **暂停调度** ，提示暂停调度任务下发成功，稍后点击

**刷新** 图标，节点状态变更为“健康/暂停调度”。



暂停调度

3. 如果需要恢复调度，则点击列表右侧 **恢复调度** 按钮，提示恢复调度任务下发成功，稍后点击 **刷新** 图标，节点状态将变更为 **健康/可调度** 。

## 移除节点

操作步骤如下：

1. 进入边缘单元详情页，选择左侧菜单 **边缘资源** -> **边缘节点** 。
2. 在节点列表右侧，点击三个点 **！** -> **移除节点** ，弹出确认移除弹框。



### 暂停调度

3. 手动完成工作负载和终端设备资源解绑操作。如果节点上的资源已清除，忽略此步骤。

1. 点击弹框中 **工作负载** 按钮，跳转到工作负载列表页面。
2. 找到节点上的工作负载，逐一执行删除操作。
3. 点击弹框中 **终端设备** 按钮，跳转到终端设备列表页面。
4. 找到与节点绑定的终端设备，执行解绑操作。

4. 输入框中输入节点名称，点击 **删除** 按钮，完成节点移除。

## 批量升级边缘节点

边缘节点上安装的边缘软件，像 EdgeCore、MQTT 等软件支持升级，DCE 5.0 云边协同会不定期发布新版本，用户可以根据需求升级边缘节点。

## 升级说明

- 升级任务中选中的节点无差别更新升级到与当前云端 CloudCore 版本一致。例如，当前云端 CloudCore 版本为 1.13，选中的节点版本有 1.12、1.14，那么升级版本统一为 1.13。
- 只有健康状态的节点可升级，包含可调度和暂停调度状态。
- 为了让您的边缘节点应用更稳定可靠的运行，需要由您在业务影响最小的时间窗内进行节点升级，以减轻对您业务的影响。
- 升级期间，边缘节点上的应用业务不会中断，如果您有使用消息路由功能，可能会有短暂影响。
- 请勿在节点升级过程中变更节点配置，比如重启 Docker、变更网络配置等，这些操作会增大节点升级失败风险。

## 操作步骤

1. 在边缘节点列表页面，点击 **批量管理** 按钮，选择 **批量升级** 页签，点击右上角 **创建升级任务** 按钮。

批量升级

批量升级

2. 填写相关参数。
  - 任务名称：批量升级任务名称不能为空，长度限制为 63 位。
  - 描述：批量升级任务描述信息。
  - 升级版本：升级版本与云端 CloudCore 版本一致。
  - 升级对象：需要升级的节点，可以是指定节点或通过标签匹配。

说明：升级仅对与升级版本有差异的节点进行操作，与升级版本一致的节点将被自动忽略。

### 创建升级任务

创建升级任务

## 节点升级状态说明

- 升级中：节点正在进行升级操作。
- 升级成功：节点已经完成升级操作，并成功升级到与云端 CloudCore 版本一致。
- 升级失败回滚失败：节点升级失败，系统自动回滚到升级前版本失败。此时需要用户排查失败原因，查看 EdgeCore 等组件日志，并手动处理。
- 升级失败回滚成功：节点升级失败，系统自动回滚到升级前版本，回滚成功。可以查看失败原因，排查解决问题后再次尝试升级操作。

## 创建边缘节点组

在边缘计算场景中，边缘节点通常分布在不同的地理区域，这些区域中的节点计算资源、网络结构和硬件平台存在诸多差异。根据边缘节点的地理分布特点，可以把同一区域的边缘节点分为一组，将边缘节点以节点组的形式组织起来，同一节点同时只能属于一个节点组。将边缘应用部署到节点组，提升跨地域应用部署的运维效率。

下文说明创建边缘节点组的步骤。

1. 进入边缘单元详情页，选择左侧菜单 **边缘资源** -> **边缘节点组** 。
2. 点击节点组列表右上角 **创建边缘节点组** 按钮。



## 创建边缘节点组

### 创建边缘节点组

#### 3. 填写相关参数。

- 节点组名称：小写字母、数字、中划线（-）、点（.）的组合，不能有连续符号；以字母或数字为开头、结尾；最多包含 253 个字符。
- 描述：节点组描述信息。
- 节点选择方式：边缘节点组中的节点可以指定多个节点或通过标签选择器匹配，两种方式可以同时使用并生效。

## 创建边缘节点组

### 创建边缘节点组

#### 4. 点击 **确定** 按钮，即创建边缘节点组成功，返回到边缘节点组列表。

下一步：[管理边缘节点组](#)

# 管理边缘节点组

本文说明编辑、删除边缘节点组的操作步骤。

## 编辑节点组

操作步骤如下：

1. 进入边缘单元详情页，选择左侧菜单 **边缘资源** -> **边缘节点组** 。
2. 点击边缘节点组列表中的节点组名称。
3. 节点组详情页，点击右上角 **编辑节点组** 按钮。

编辑节点组

编辑节点组

4.编辑页面，修改节点组配置信息。

编辑节点组

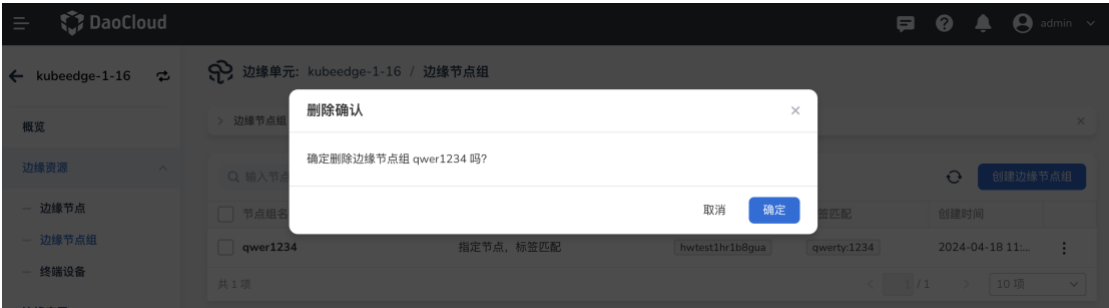
编辑节点组

5.点击 **确定** ，完成节点组配置修改，系统自动返回到节点组详情页。

删除节点组

操作步骤如下：

- 1.进入边缘单元详情页，选择左侧菜单 **边缘资源** -> **边缘节点组** 。
- 2.在节点组列表的右侧，点击三个点 **⋮** ，在弹出菜单中选择 **删除** 。
- 3.输入节点组名称，点击 **删除** ，完成删除操作。



删除节点组

终端设备概述

终端设备

终端设备可以小到传感器、控制器，大到智能摄像机或工控机床。

终端设备可以连接到边缘节点，支持通过 Modbus 协议接入并进行统一管理。

## 设备模型

设备模型是终端设备在云端的数字化表示，将设备共同特征抽象成数据模型，用属性来定义设备是什么，可以对外提供哪些信息。

## 设备孪生

设备孪生指终端设备的动态数据，包括专有实时数据，例如灯的开、关状态。这种数据也可以称为终端设备的孪生属性。

设备孪生具有与物理设备相同的特性，便于终端设备与应用之间进行更好地通信。应用发送的命令首先到达设备孪生，设备孪生根据应用设置的期望状态进行状态更新，此外终端设备实时反馈自身的实际状态，设备孪生同时记录终端设备的期望值和实际值。终端设备在离线状况下再次上线时，终端设备的状态也能得到同步。

### 设备孪生

#### 设备孪生

在云边协同模块中可以创建终端设备，将终端设备与边缘节点关联，关联后会在边缘节点上保存被关联设备的孪生属性信息。边缘节点上的应用程序可在边缘节点获取设备孪生属性实际值，同时可以修改终端设备孪生期望值以修改设备状态。

## 使用流程

管理和控制终端设备，通常使用的步骤如下：

1. [创建设备模型](#)
2. [创建终端设备](#)

- 3.将终端设备关联到边缘节点
- 4.[管理和控制终端设备](#)，监测终端设备状态

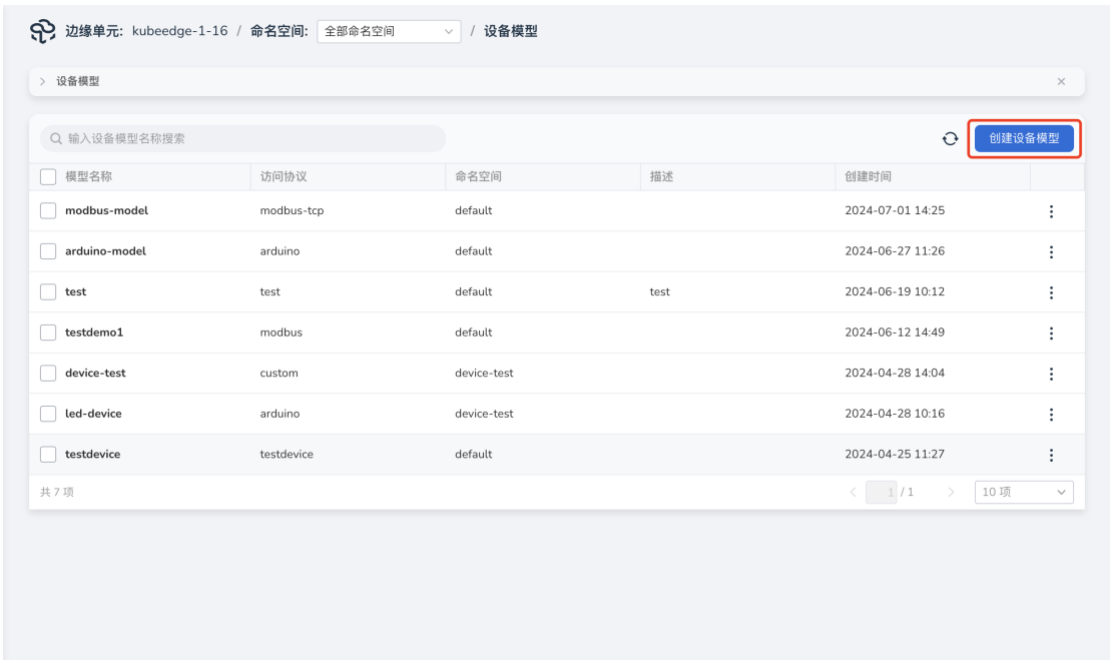
下一步：[创建设备模型](#)

## 创建设备模型

设备模型是终端设备在云端的数字化表示，将设备共同特征抽象成数据模型，用属性来定义设备是什么，可以对外提供哪些信息。

本文介绍创建设备模型的操作步骤，流程如下：

- 1.进入边缘单元详情页，选择左侧菜单 **边缘资源 -> 设备模型** 。
- 2.点击终端设备列表右上角 **创建设备模型** 按钮。



### 创建设备模型

- 3.填写基础信息。
- **模型名称** ：小写字母、数字、中划线 (-)、点 (.) 的组合，不能有连续符

号；以字母或数字为开头、结尾；最多包含 253 个字符。

- **访问协议**：DCE 5.0 云边协同支持 Modbus 等多种协议设备接入。
- **命名空间**：设备所在命名空间，命名空间的资源相互隔离。
- **描述**：设备模型描述信息。

← 创建设备模型

1

2

基础信息

设备配置

模型名称 \*

最长 253 个字符，只能是小写字母、数字、中划线(-)、点(.)的组合，且不能有连续符号，以字母或数字为开头、结尾。

访问协议 \*

请输入协议名称

命名空间 \*

default

↻ 创建命名空间

描述

内容无限制，长度限制为 63 个字符

取消 下一步

## 创建设备模型

4. 填写设备配置, 可以添加设备孪生属性和标签, 关联的设备实例可以直接引用模型配置。

- **孪生属性**：选填，指终端设备的动态数据，包括专有实时数据，例如灯的开、关状态，温湿度传感器的温度、适度等。
- **标签**：选填，通过给设备打上标签，将不同设备进行分类管理。

← 创建设备模型

✓

基础信息

2

设备配置

孪生属性

1 孪生属性会明文展示所输入的信息。请不要填入敏感数据，如涉及敏感信息，请先加密。

新增孪生属性

| 属性名  | 属性类型 | 访问权限 | 属性单位 | 属性值区间 |
|------|------|------|------|-------|
| 暂无数据 |      |      |      |       |

标签

+ 添加

取消

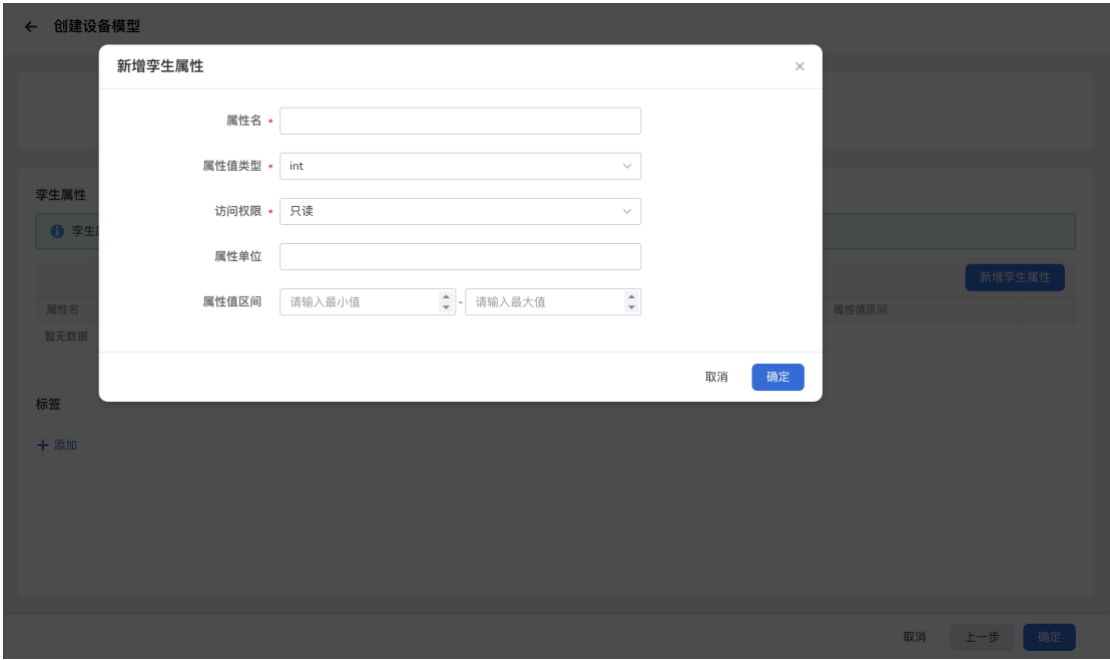
上一步

确定

设备配置

新增孪生属性，用户可以根据设备类型选择对应的寄存器类型，并填写对应的参数。参数说明如下：

- **属性名** ：必填项，设备属性名称。
- **属性值类型** ：必填项，属性值类型，选项包含 string、int、float、boolean。
- **访问权限** ：设备孪生属性访问权限，选项包含 读/写、只读。
- **属性值单位** ：选填，属性值单位。
- **属性值区间** ：对获取的原始数据进行范围限定。



新增孪生属性

下一步：[创建终端设备](#)

# 创建终端设备

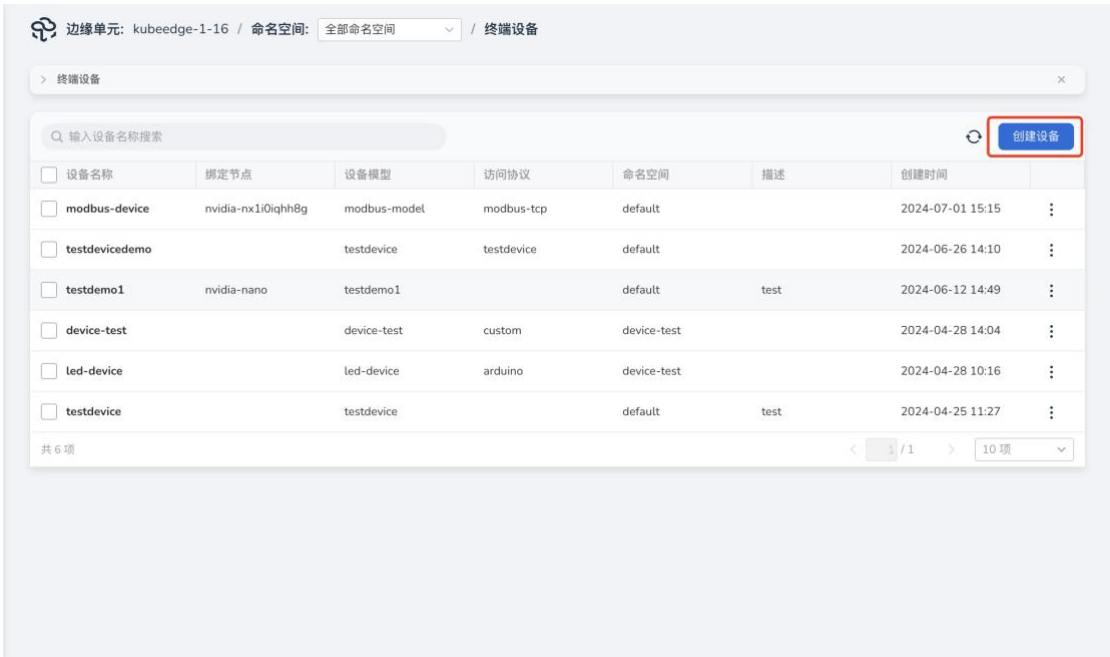
终端设备可以连接到边缘节点，支持通过 Modbus 协议或自定义协议接入。终端设备接入后，可以在云端管理平面对设备进行统一管理。

本文介绍创建终端设备和终端设备绑定边缘节点的操作步骤。

## 创建终端设备

操作步骤如下：

- 1. 进入边缘单元详情页，选择左侧菜单 **边缘资源 -> 终端设备** 。
- 2. 点击终端设备列表右上角 **创建设备** 按钮。



创建设备

3.填写 基础信息 。

- **设备名称** ：小写字母、数字、中划线 (-)、点 (.) 的组合，不能有连续符号；以字母或数字为开头、结尾；最多包含 253 个字符。
- **命名空间** ：设备所在命名空间，命名空间的资源相互隔离。
- **设备模型** ：设备关联的设备模型。
- **访问协议** ：设备关联的设备模型访问协议会自动填入，此处不支持修改。
- **描述** ：设备描述信息。



← 创建设备

1

2

3

4

基础信息

设备配置

访问配置

信息确认

设备名称 \*

最长 253 个字符，只能是小写字母、数字、中划线(-)、点(.)的组合，且不能有连续符号，以字母或数字为开头、结尾。

命名空间 \*

default

创建命名空间

设备模型 \*

modbus-model

访问协议

modbus-tcp

描述

取消

下一步

创建设备

4. 填写设备配置，可以添加设备孪生属性和标签。

- **孪生属性**：选填，指终端设备的动态数据，包括专有实时数据，例如灯的开、关状态，温湿度传感器的温度、适度等。
- **标签**：选填，通过给设备打上标签，将不同设备进行分类管理。

创建设备

创建设备

新增孪生属性，用户可以根据设备类型选择对应的寄存器类型，并填写对应的参数。参数说明如下：

- **属性名**：必填项，设备属性名称。
- **期望值**：选填，属性的期望值，当属性访问权限为 读/写 时可填写。
- **采集间隔**：选填，对设备进行指定间隔的数据采集。
- **上报间隔**：选填，对设备进行指定间隔的数据上报。
- **访问方式**：平台连接到设备后，访问设备属性的方式，确保跟 mapper 属性

访问方式一致。

新增孪生属性

基本信息

属性名

采集间隔

上报间隔

属性访问方式

YAML 格式参数

YAML 示例

取消

确定

新增孪生属性

5.填写设备 访问配置 。

平台连接到设备的访问参数，YAML 格式填写。

← 创建设备

基础信息

设备配置

访问配置

信息确认

参数配置

YAML 格式参数

YAML 示例

取消

上一步

下一步

访问配置

6.信息确认，确认所配置的信息无误，点击 **确定** ，完成设备创建。

← 创建设备

✓

基础信息

✓

设备配置

✓

访问配置

4

信息确认

基础信息

设备名称

test

访问协议

modbus-tcp

命名空间

default

描述

-

设备配置

孪生属性

| 属性名称 | 期望值 | 采样间隔 | 上报间隔 | 属性访问方式 |
|------|-----|------|------|--------|
| 暂无数据 |     |      |      |        |

标签

| 键 | 值 |
|---|---|
|---|---|

取消

上一步

确定

创建设备

## 终端设备绑定边缘节点

一个终端设备只能绑定一个边缘节点，设备绑定节点后，部署在节点上的应用可以通过云端创建的设备孪生获取到设备实时数据。

操作步骤如下：

- 1.进入边缘单元详情页，选择左侧菜单 **边缘资源 -> 终端设备** 。
- 2.在终端设备列表的右侧，点击 **！** 按钮，在弹出菜单中选择 **绑定节点** 。
- 3.在弹框中选择要绑定的节点，点击 **确定** ，完成边缘节点的绑定。

创建设备

创建设备

下一步：[管理终端设备](#)

# 终端设备管理

DCE 5.0 云边协同支持对终端设备的基本信息、孪生属性、标签、访问配置进行编辑修改操作。

!!1 note

仅在终端设备和边缘节点解除绑定状态下，才能对终端设备进行编辑操作。

## 编辑基本信息

操作步骤如下：

- 1.进入边缘单元详情页，选择左侧菜单 **边缘资源** -> **终端设备** 。
- 2.点击终端设备列表中的设备名称。
- 3.设备详情页，点击右上角 **编辑** 按钮。
- 4.编辑弹框，修改设备描述信息。
- 5.点击 **确定** ，完成基本信息修改，系统自动返回到设备详情页。

编辑基本信息

编辑基本信息

## 编辑孪生属性

操作步骤如下：

- 1.在设备详情页，选择 **孪生属性** 页签。
- 2.点击属性列表右上角 **新增孪生属性** 或在列表的右侧点击三个点 **⋮** ，在弹出菜单中选择 **编辑** 、 **删除** 。
- 3.在对应操作的弹框页，点击 **确定** 按钮，完成孪生属性的新增、编辑、删除操作。

编辑孪生属性

编辑孪生属性

编辑孪生属性

编辑孪生属性

## 修改标签

操作步骤如下：

1. 在设备详情页，选择 **标签** 页签。
2. 点击标签列表右上角 **修改标签** 按钮。
3. 在修改标签弹框，可以对便签进行添加、删除操作。
4. 点击 **确定** 按钮，完成设备标签修改操作。

修改标签

修改标签

## 编辑访问配置


操作步骤如下：

1. 在设备详情页，选择 **访问配置** 页签。
2. 点击标签列表右上角 **编辑** 按钮。
3. 在编辑弹框，修改访问配置内容。
4. 点击 **确定** 按钮，完成设备访问配置修改操作。

编辑访问配置

编辑访问配置

# 删除终端设备

在终端设备列表的右侧，点击三个点 ，在弹出菜单中选择 **删除**。



删除终端设备

# 边缘应用概述

DCE 5.0 云边协同支持下发容器应用到边缘节点。容器应用可以直接从镜像仓库中拉取镜像。

容器镜像的架构必须与边缘节点架构一致。

# 工作负载

工作负载 (Deployment) 是管理应用副本的一种 API 对象, 具体表现为没有本地状态的 Pod, 这些 Pod 之间完全独立、功能相同, 能够滚动更新, 其实例的数量可以灵活扩缩。

# 批量工作负载

将相同配置或差异化较小的工作负载定义部署到节点组, 是一个任务或批量部署动作。

- 通过标签, 将同一种类的资源过滤出来, 针对某一类或某一地域的资源, 提供容器应用批量部署操作
- 每一个节点组部署对应一个批量部署任务, 一个工作负载定义可以部署到多个节点组
- 避免由于网络抖动或高并发流量控制等导致的任务失败问题, DCE 5.0 云边协同提供可

视化的部署视图，将批量部署的状态和结果及时同步

批量工作负载

批量工作负载

## 应用故障迁移

- 当节点组中应用实例出现异常或节点故障时，DCE 5.0 云边协同离线自愈功能能够快速将应用实例调度到节点组中其它可用的节点上运行，确保应用的自主运行，保障业务的持续性
- 即使在节点处于离线状态（边缘节点与云端断开连接）时仍然能够自动调度

应用故障迁移

应用故障迁移

## 配置项与密钥

配置项和密钥是用来存储和管理工作负载所需的认证信息、证书、密钥等重要、敏感信息的资源类型，内容由用户决定，资源创建完成后，可在容器应用中加载使用。

云边协同模块的配置项和密钥的创建和使用流程和容器管理模块的流程一致，在边缘单元详情页，进入到 **配置项和密钥** 菜单。创建和使用流程参考[创建配置项](#)。

## 创建工作负载

本文介绍如何通过镜像和 YAML 文件两种方式创建工作负载。这里的工作负载指无状态负载。

工作负载 (Deployment) 是管理应用副本的一种 API 对象，具体表现为没有本地状态的 Pod，

这些 Pod 之间完全独立、功能相同，能够滚动更新，其实例的数量可以灵活扩缩。

Deployment 是无状态的，不支持数据持久化，适用于部署无状态的、不需要保存数据、随时可以重启回滚的应用。

## 镜像创建

参考以下步骤，使用镜像创建一个无状态负载。

1. 进入边缘单元详情页，选择左侧菜单 **边缘应用** -> **工作负载**。
2. 点击工作负载列表右上角 **镜像创建** 按钮。

### 工作负载列表

#### 工作负载列表

3. 填写基本信息

- 负载名称：最多包含 63 个字符，只能包含小写字母、数字及分隔符（“-”），且必须以小写字母或数字开头及结尾，例如 deployment-01。同一命名空间内同一类型工作负载的名称不得重复，而且负载名称在工作负载创建好后不可更改
- 命名空间：选择将新建的负载部署在哪个命名空间，默认使用 default 命名空间。找不到所需的命名空间时可以根据页面提示去创建新的命名空间。
- 实例数：负载的 Pod 实例数量，默认创建 1 个 Pod 实例，不可修改
- 描述：输入负载的描述信息，内容自定义

### 填写基本信息

#### 填写基本信息


4. 填写容器配置



容器配置分为基本信息、生命周期、健康检查、环境变量、数据存储、安全设置六部分，

点击下方的相应页签可查看各部分的配置要求。

### !!! note

容器配置仅针对单个容器进行配置，如需在一个容器组中添加多个容器，可点击右侧的  添加多个容器

### === “基本信息（必填）”

在配置容器相关参数时，必须正确填写容器的名称、镜像参数，否则将无法进入下一步。参考以下要求填写配置后，点击确认。

- 容器名称：最多包含 63 个字符，支持小写字母、数字及分隔符（“-”）。必须以小写字母或数字开头及结尾，例如 nginx-01。
- 容器镜像：输入镜像地址或名称。输入镜像名称时，默认从官方的 [DockerHub](<https://hub.docker.com/>) 拉取镜像。

接入 DCE 5.0 的[镜像仓库]([../kangaroo/intro/index.md](https://docs.daocloud.io/daocloud-docs-images/docs/zh/docs/kant/images/create-app-03.png))模块后，可以点击右侧的选择镜像来选择镜像。

- 更新策略：勾选总是拉取镜像后，负载每次重启/升级时都会从仓库重新拉取镜像。如果不勾选，则只拉取本地镜像，

只有当镜像在本地不存在时才从镜像仓库重新拉取。更多详情可参考[镜像拉取策略](<https://kubernetes.io/zh-cn/docs/concepts/containers/images/#image-pull-policy>)。

- 特权容器：默认情况下，容器不可以访问宿主机上的任何设备，开启特权容器后，容器即可访问宿主机上的所有设备，享有宿主机上的运行进程的所有权限。
- CPU/内存配额：CPU/内存资源的请求值（需要使用的最小资源）和限制值（允许使用的最大资源）。

请根据需要为容器配置资源，避免资源浪费和因容器资源超额导致系统故障。默认值如图所示。

- GPU 配额：为容器配置 GPU 用量，仅支持输入正整数。GPU 配额设置支持为容器设置独享整张 GPU 卡或部分 vGPU。例如，对于一张 8 核心的 GPU 卡，输入数字 8 表示让容器独享整张卡，输入数字 1 表示为容器配置 1 核心的 vGPU。

> 设置 GPU 配额之前，需要管理员预先在集群节点上安装 GPU 卡及驱动插件，并在[集群设置]([../kpanada/user-guide/clusterops/cluster-settings.md](https://docs.daocloud.io/daocloud-docs-images/docs/zh/docs/kant/images/create-app-03.png))中开启 GPU 特性。

![填写容器配置](<https://docs.daocloud.io/daocloud-docs-images/docs/zh/docs/kant/images/create-app-03.png>)

### === “生命周期（选填）”

设置容器启动时、启动后、停止前需要执行的命令。详情可参考容[容器生命周期配置]([../kpanada/user-guide/workloads/pod-config/lifecycle.md](https://docs.daocloud.io/daocloud-docs-images/docs/zh/docs/kant/images/create-app-04.png))。

![生命周期](<https://docs.daocloud.io/daocloud-docs-images/docs/zh/docs/kant/images/create-app-04.png>)

### === “健康检查（选填）”

用于判断容器和应用的健康状态，有助于提高应用的可用性。详情可参考[容器健康检查配置](../../kpanda/user-guide/workloads/pod-config/health-check.md)。

![健康检查](https://docs.daocloud.io/daocloud-docs-images/docs/zh/docs/kant/images/create-app-05.png)

### === “环境变量（选填）”

配置 Pod 内的容器参数，为 Pod 添加环境变量或传递配置等。详情可参考[容器环境变量配置](../../kpanda/user-guide/workloads/pod-config/env-variables.md)。

![环境变量](https://docs.daocloud.io/daocloud-docs-images/docs/zh/docs/kant/images/create-app-06.png)

### === “数据存储（选填）”

配置容器挂载数据卷和数据持久化的设置。详情可参考[容器数据存储配置](../../kpanda/user-guide/workloads/pod-config/env-variables.md)。

![数据存储](https://docs.daocloud.io/daocloud-docs-images/docs/zh/docs/kant/images/create-app-07.png)

### === “安全设置（选填）”

通过 Linux 内置的账号权限隔离机制来对容器进行安全隔离。您可以通过使用不同权限的账号 UID（数字身份标记）来限制容器的权限。

例如，输入 0 表示使用 root 账号的权限。

![安全设置](https://docs.daocloud.io/daocloud-docs-images/docs/zh/docs/kant/images/create-app-08.png)

## 5. 填写高级配置

高级配置包括负载的节点调度、标签与注解、访问配置三部分，可点击下方的页签查看各部分的配置要求。

### === “节点调度”

点击 选择边缘节点 按钮，在弹出的选择弹框中，可以选择指定的某一节点部署工作负载。

![节点调度](https://docs.daocloud.io/daocloud-docs-images/docs/zh/docs/kant/images/create-app-09.png)

![选择边缘节点](https://docs.daocloud.io/daocloud-docs-images/docs/zh/docs/kant/images/create-app-10.png)

### === “标签与注解”

点击 **\_\_添加\_\_** 按钮，可以为工作负载和容器组添加标签和注解。

![标签与注解](https://docs.daocloud.io/daocloud-docs-images/docs/zh/docs/kant/images/create-app-11.png)

### === “访问配置”

容器访问支持不可访问、端口映射和主机网络三种配置方式。

- 不可访问：工作负载不可被外部访问。
- 端口映射：容器网络虚拟化隔离，容器拥有单独的虚拟网络，容器与外部通信需要与主机做端口映射。

配置端口映射后，流向主机端口的流量会映射到对应的容器端口。例如容器端口 80 与主机端口 8080 映射，

那主机 8080 端口的流量会流向容器的 80 端口。

- 主机网络：使用宿主机（边缘节点）的网络，即容器与主机间不做网络隔离，使用同一个 IP。

![访问配置](https://docs.daocloud.io/daocloud-docs-images/docs/zh/docs/kant/images/create-app-12.png)

6. 点击 **确定** 按钮，完成工作负载的创建。

## YAML 创建

除了通过镜像方式创建外，还可以通过 YAML 文件更快速地创建无状态负载。

操作步骤如下：

1. 进入边缘单元详情页，选择左侧菜单 **边缘应用** -> **工作负载**。
2. 点击终端设备列表右上角 **YAML 创建** 按钮。

工作负载列表

工作负载列表

3. 输入或粘贴事先准备好的 YAML 文件，点击 **确定** 即可完成创建。

!!! note

在使用 YAML 创建工作负载时，建议增加以下限制。

- 增加标签（labels）用于标识边缘应用
- 增加节点亲和性（affinity）用于将 Pod 分配到指定边缘节点

```
```yaml
labels:
  kant.io/app: "
affinity:
  nodeAffinity:
    requiredDuringSchedulingIgnoredDuringExecution:
      nodeSelectorTerms:
        - matchExpressions:
            - key: kubernetes.io/hostname
              operator: In
              values:
                - edge1h6382jnk
```
```

影响：

- 如未加上标签（labels）配置，云边协同模块的工作负载列表将不会显示该工作负载，此时用户可以到容器管理模块查看工作负载详情。
- 如未加上节点亲和性（affinity）配置，工作负载可能会随机调度，无法部署到指定边缘节点。

## YAML 创建

### YAML 创建

# 创建批量工作负载

将边缘节点按地区划或特性可以划分为不同的节点组，并将应用所需资源打包成一个整体在节点组上进行部署，降低了边缘应用生命周期管理的复杂度，有效减少运维成本。

本文介绍如何通过镜像和 YAML 文件两种方式创建批量工作负载。

## 镜像创建

参考以下步骤，使用镜像创建一个无状态负载。

1. 进入边缘单元详情页，选择左侧菜单 **边缘应用** -> **批量工作负载**。
2. 点击批量工作负载列表右上角 **镜像创建** 按钮。

### 批量工作负载列表

#### 批量工作负载列表

#### 3. 填写基本信息

- 负载名称：最多包含 63 个字符，只能包含小写字母、数字及分隔符（“-”），且必须以小写字母或数字开头及结尾，例如 deployment-01。同一命名空间内同一类型工作负载的名称不得重复，而且负载名称在工作负载创建好后不可更改
- 命名空间：选择将新建的负载部署在哪个命名空间，默认使用 default 命名空间。找不到所需的命名空间时可以根据页面提示去创建新的命名空间
- 实例数：负载的 Pod 实例数量，默认创建 1 个 Pod 实例，可以修改
- 描述：输入负载的描述信息，内容自定义

### 填写基本信息

#### 填写基本信息

#### 4. 填写容器配置

容器配置分为基本信息、生命周期、健康检查、环境变量、数据存储、安全设置六部分，

点击下方的相应页签可查看各部分的配置要求。

#### !!! note

容器配置仅针对单个容器进行配置，如需在一个容器组中添加多个容器，可点击右侧的 **\_+\_** 添加多个容器

### === “基本信息（必填）”

在配置容器相关参数时，必须正确填写容器的名称、镜像参数，否则将无法进入下一步。  
参考以下要求填写配置后，点击确认。

- 容器名称：最多包含 63 个字符，支持小写字母、数字及分隔符（“-”）。必须以小写字母或数字开头及结尾，例如 nginx-01。
- 容器镜像：输入镜像地址或名称。输入镜像名称时，默认从官方的 [DockerHub](<https://hub.docker.com/>) 拉取镜像。

接入 DCE 5.0 的[镜像仓库]([../kangaroo/intro/index.md](#))模块后，可以点击右侧的选择镜像来选择镜像。

- 更新策略：勾选总是拉取镜像后，负载每次重启/升级时都会从仓库重新拉取镜像。如果不勾选，则只拉取本地镜像，

只有当镜像在本地不存在时才从镜像仓库重新拉取。更多详情可参考[镜像拉取策略](<https://kubernetes.io/zh-cn/docs/concepts/containers/images/#image-pull-policy>)。

- 特权容器：默认情况下，容器不可以访问宿主机上的任何设备，开启特权容器后，容器即可访问宿主机上的所有设备，

享有宿主机上的运行进程的所有权限。

- CPU/内存配额：CPU/内存资源的请求值（需要使用的最小资源）和限制值（允许使用的最大资源）。

请根据需要为容器配置资源，避免资源浪费和因容器资源超额导致系统故障。默认值如图所示。

- GPU 配额：为容器配置 GPU 用量，仅支持输入正整数。GPU 配额设置支持为容器设置独享整张 GPU 卡或部分 vGPU。例如，对于一张 8 核心的 GPU 卡，输入数字 8 表示让容器独享整张卡，输入数字 1 表示为容器配置 1 核心的 vGPU。

> 设置 GPU 配额之前，需要管理员预先在集群节点上安装 GPU 卡及驱动插件，并在[集群设置]([../kpanda/user-guide/clusterops/cluster-settings.md](#))中开启 GPU 特性。

### === “生命周期（选填）”

设置容器启动时、启动后、停止前需要执行的命令。

详情可参考容[容器生命周期配置]([../kpanda/user-guide/workloads/pod-config/lifecycle.md](#))。

### === “健康检查（选填）”

用于判断容器和应用的健康状态，有助于提高应用的可用性。

详情可参考[容器健康检查配置]([../kpanda/user-guide/workloads/pod-config/health-check.md](#))。

### === “环境变量（选填）”

配置 Pod 内的容器参数，为 Pod 添加环境变量或传递配置等。

详情可参考[容器环境变量配置]([../kpanda/user-guide/workloads/pod-config/env-variables.md](#))。

### === “数据存储（选填）”

配置容器挂载数据卷和数据持久化的设置。

详情可参考[容器数据存储配置](../../kpanda/user-guide/workloads/pod-config/env-variables.md)。

### === “安全设置（选填）”

通过 Linux 内置的账号权限隔离机制来对容器进行安全隔离。您可以通过使用不同权限的账号 UID（数字身份标记）来限制容器的权限。

例如，输入 0 表示使用 root 账号的权限。

## 5. 填写高级配置

高级配置包括负载的标签与注解、访问配置两部分，可点击下方的页签查看各部分的配置要求。

### === “标签与注解”

点击 **\_\_添加\_\_** 按钮，可以为工作负载和容器组添加标签和注解。

![标签与注解](https://docs.daocloud.io/daocloud-docs-images/docs/zh/docs/kant/images/create-batch-app-03.png)

### === “访问配置”

容器访问支持不可访问、端口映射和主机网络三种配置方式。

- 不可访问：工作负载不可被外部访问。
- 端口映射：容器网络虚拟化隔离，容器拥有单独的虚拟网络，容器与外部通信需要与主机做端口映射。

配置端口映射后，流向主机端口的流量会映射到对应的容器端口。例如容器端口 80 与主机端口 8080 映射，

那主机 8080 端口的流量会流向容器的 80 端口。

- 主机网络：使用宿主机（边缘节点）的网络，即容器与主机间不做网络隔离，使用同一个 IP。

![访问配置](https://docs.daocloud.io/daocloud-docs-images/docs/zh/docs/kant/images/create-batch-app-04.png)

## 6. 点击 **确定** 按钮，完成批量工作负载的创建。

## YAML 创建

除了通过镜像方式外，还可以通过 YAML 文件更快速地创建创建批量工作负载。操作步骤如下：

1. 进入边缘单元详情页，选择左侧菜单 **边缘应用** -> **批量工作负载**。
2. 点击终端设备列表右上角 **YAML 创建** 按钮。

### 批量工作负载列表

#### 批量工作负载列表

3. 输入或粘贴事先准备好的 YAML 文件，点击 **确定** 即可完成创建。

#### !!! note

在使用 YAML 创建批量工作负载的时候，建议加上以下限制。

标签（labels）用于标识边缘应用，workloadScope 字段用于对部署到节点组的边缘应用进行差异化配置，

当前差异化支持镜像（imageOverrides）和副本数（replicas）设置。

参考示例如下。

```
```yaml
apiVersion: apps.kubeedge.io/v1alpha1
kind: EdgeApplication
metadata:
  name: nginx-app2
  namespace: default
spec:
  workloadScope:
    targetNodeGroups:
      - name: test
      overrides:
        imageOverrides:
          - component: Repository
            operator: replace
            predicate:
              path: /spec/template/spec/containers/0/image
              value: nginx1
          - component: Tag
            operator: replace
            predicate:
              path: /spec/template/spec/containers/0/image
              value: v1
        replicas: 1
  workloadTemplate:
    manifests:
```



```
- apiVersion: apps/v1
  kind: Deployment
  metadata:
    labels:
      kant.io/app: "
    name: nginx-test2
    namespace: default
  spec:
    replicas: 10
    selector:
      matchLabels:
        app: nginx-test2
    template:
      metadata:
        labels:
          app: nginx-test2
      spec:
        containers:
          - image: nginx
            imagePullPolicy: IfNotPresent
            name: nginx
  ...
```

## 部署批量工作负载

本文介绍将批量工作负载定义部署到节点组的操作步骤。


### 前提条件

- 创建一个批量工作负载。
- 创建一个节点组。

### 部署

操作步骤如下：

1. 进入边缘单元详情页，选择左侧菜单 **边缘应用** -> **批量工作负载** 。

2. 在批量工作负载列表的右侧，点击  按钮，在弹出菜单中选择 **部署**。

#### 批量工作负载列表

#### 批量工作负载列表

3. 填写相关参数。

- 部署对象：需要部署批量工作负载的边缘节点组。

点击 **选择边缘节点组**，勾选需要选择的边缘节点组。页面左上角的搜索框输入节点组名称，可以快速检索想要部署的边缘节点组。

- 差异化配置：支持对不同节点组的工作负载实例数和容器镜像进行差异化配置。

!!! note

1. 默认实例数为批量工作负载定义的实例数，随着定义的修改而修改。修改后的实例数不受批量负载定义的更新影响。
2. 支持对容器镜像的镜像、镜像仓、版本进行分段配置，配置后的分段对象不受批量负载定义的更新影响。

#### 部署

#### 部署

#### 选择边缘节点组

#### 选择边缘节点组


## 查看定义

将应用资源打包，抽象成一个应用定义，按照节点组部署，产生工作负载实例。支持对应用定义的基本信息、容器配置、标签与注解、访问配置进行编辑操作。

说明：更新工作负载定义配置后，除差异化配置外，所有已部署实例的原有配置将同步被替换。

操作步骤如下：

1. 进入边缘单元详情页，选择左侧菜单 **边缘应用** -> **批量工作负载**。

2. 在批量工作负载列表的右侧，点击  按钮，在弹出菜单中选择 **查看定义**。
3. 在定义详情页，切换容器配置、标签与注解、访问配置 Tab，点击 **编辑** 按钮，进入对应编辑界面，完成定义配置修改操作。

查看定义

查看定义

## 边云消息概述

DCE 5.0 云边协同平台提供了边云消息路由功能，用户可以配置消息路由，平台根据消息路由将消息转发至对应端点，让消息按照规定的路径转发，灵活控制数据路由，并提高数据安全性。

- 消息端点：发送或接收消息的一方，可以是边缘应用、云服务。
- 消息路由：消息转发的路径。

## 消息端点

DCE 5.0 云边协同提供以下三种类型消息端点：

- Rest：云端端点，向边缘发送消息请求的源端点。或者作为目标端点，从边缘接收消息。
- Event Bus：边端端点，可作为源端点，向云端发送数据。或者作为目标端点，从云端接收消息。
- Service Bus：边端端点，可作为目标端点，用于接收从云端传递的消息。

## 消息路由

对应不同类型的消息端点，DCE 5.0 云边协同提供以下消息转发路径：

- Rest -> EventBus：用户应用调用云端的 REST API 发送消息，最终消息发送到边缘中的 MQTT broker。
- EventBus -> Rest：用户向边缘中的 MQTT broker 发布消息，最终将消息发送到云端的 REST API。
- Rest -> ServiceBus：用户应用调用云端 REST API 发送消息，最终消息发送到边缘应用。

路径

路径

后续操作：[创建消息路由](#)

## 创建消息路由

本文介绍创建消息端点、消息路由以及删除消息路由的步骤。

## 创建消息端点

操作步骤如下：

1. 进入 **边缘单元** 详情页，选择左侧菜单 **边云消息 -> 消息端点** 。
2. 点击消息端点列表右上角 **创建消息端点** 按钮。

创建消息端点

创建消息端点

3. 填写相关参数。
  - 消息端点类型：选择类型，当前支持 Rest 云上端点、Event Bus 边缘端点、Service Bus 边缘端点。
  - 命名空间：消息端点所在命名空间。

- 消息端点名称：输入消息端点名称。
  - 服务端口：只有类型为 Service Bus 的边缘端点需要填写，范围为 1-65535。
4. 点击 **确定**，即创建消息端点成功，返回到消息端点列表页面。

创建消息端点

创建消息端点

## 创建消息路由

操作步骤如下：

1. 进入 **边缘单元** 详情页，选择左侧菜单 **边云消息** -> **消息路由**。
2. 点击消息路由列表右上角 **创建消息路由** 按钮。

创建消息路由

创建消息路由

3. 填写相关参数。
  - 消息路由名称：输入消息路由名称。
  - 命名空间：消息路由所在的命名空间。
  - 源端点：选择源端点，选项来源于创建的消息端点。
  - 源端点资源：
    - 当源端点为云端 Rest 类型，输入 Rest 路径，如 /abc/bc。
    - 当源端点为边端 Event Bus 类型，输入 Topic，由字母、数字、下划线（\_）、中划线（-）、斜杠（/）组成。
  - 目标端点：选择目的端点，选项来源于创建的消息端点。
  - 目标端点资源：

- 当目的端点为云端 Rest 类型，输入 URL，如  
http://127.0.0.1:8080/hello。
- 当目的端点为边端 Service Bus 类型，输入 Service Bus 路径，如  
/abc/bc。

### 创建消息路由

#### 创建消息路由

4. 点击 **确定**，即创建路由规则成功，返回到消息路由列表页面。

路由规则创建完成后，系统将按照相应规则将发送到源端点指定资源的消息转发到目的端点的指定资源上。

## 删除消息路由

操作步骤如下：

1. 进入 **边缘单元** 详情页，选择左侧菜单 **边云消息** -> **消息路由**。
2. 点击指定消息路由右侧 **删除** 按钮。
3. 点击 **确定**，即删除成功，返回到消息路由列表页。

### 删除消息路由

#### 删除消息路由

## 应用网格概述

边缘计算场景下，云端会纳管海量分散的边缘节点，而边缘节点往往处于弱网、网络抖动等极差网络环境，网络拓扑较为复杂，不同区域的边缘节点往往网络不互通。为了解决边缘场景下边边业务之间通信的问题，云边协同提供应用网格的能力，支持服务发现与流量代

理，从而实现边缘业务流量闭环。

应用网格中有如下概念：

服务：服务定义了工作负载实例及访问实例的途径。使用服务名称可以代替 IP 地址，从而实现节点上应用间的相互访问。

如下图所示，边缘节点上的应用实例通过访问对应的服务，可以实现不同边缘节点上的应用实例的通信。

graph TD

```
A[服务访问端口 28099] -- B[Service]
B -->|容器端口 8099| C[应用实例 1]
B -->|容器端口 8099| D[应用实例 2]
B -->|容器端口 8099| E[应用实例 3]
```

```
classDef plain fill: #ddd,stroke:#fff,stroke-width:1px,color:#000;
```

```
classDef k8s fill: #326ce5,stroke:#fff,stroke-width:1px,color:#fff;
```

```
classDef cluster fill: #fff,stroke:#bbb,stroke-width:1px,color:#326ce5;
```

```
class B k8s
```

应用网格能力依赖 EdgeMesh 应用，用户在使用网格能力前，需要先部署 EdgeMesh 应用。

部署流程参考[部署 edgemesn 应用](#)

## 部署 EdgeMesh 应用

在使用应用网格能力前，需要先部署 EdgeMesh 应用，本文介绍具体操作流程。

### 前置准备

#### 1. 去除 K8s master 节点的污点

如果 K8s master 节点上有运行业务应用，并且需要访问集群节点上的其他应用，需要

先去除 K8s master 节点的污点，执行如下命令。

```
kubecttl taint nodes --all node-role.kubernetes.io/master-
```

### !!! note

如果 K8s master 节点上没有部署需要被代理的应用，可以跳过这一步。

## 2. 给 Kubernetes API 服务添加过滤标签

正常情况下，为避免 EdgeMesh 去代理 Kubernetes API 服务，因此需要给它添加过滤

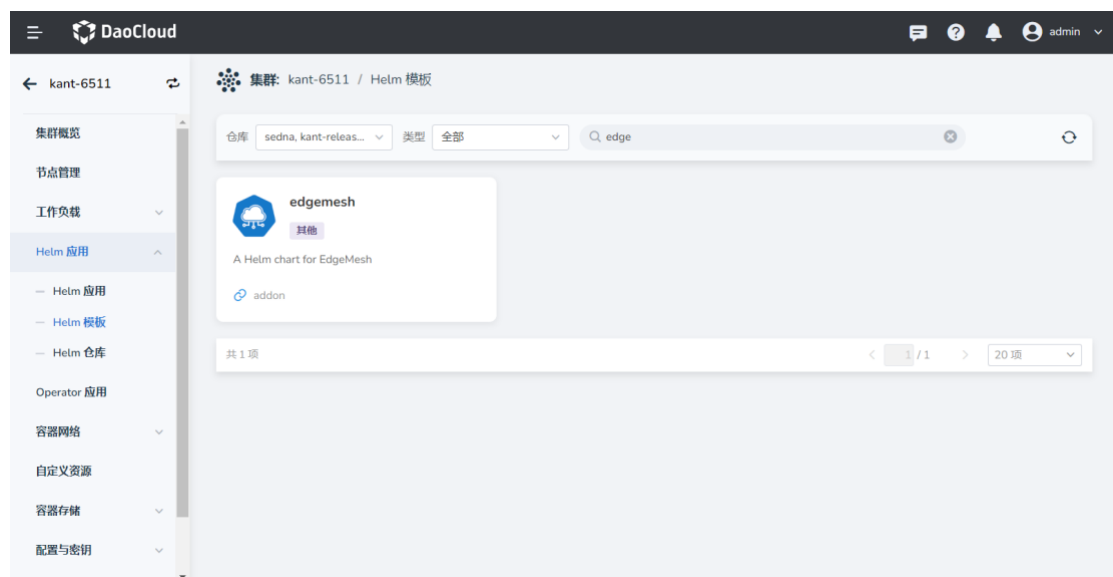
标签，更多信息请参考[服务过滤](#)。

```
kubectl label services kubernetes service.edgemesh.kubeedge.io/service-proxy-name=""
```

# Helm 安装

操作步骤如下：

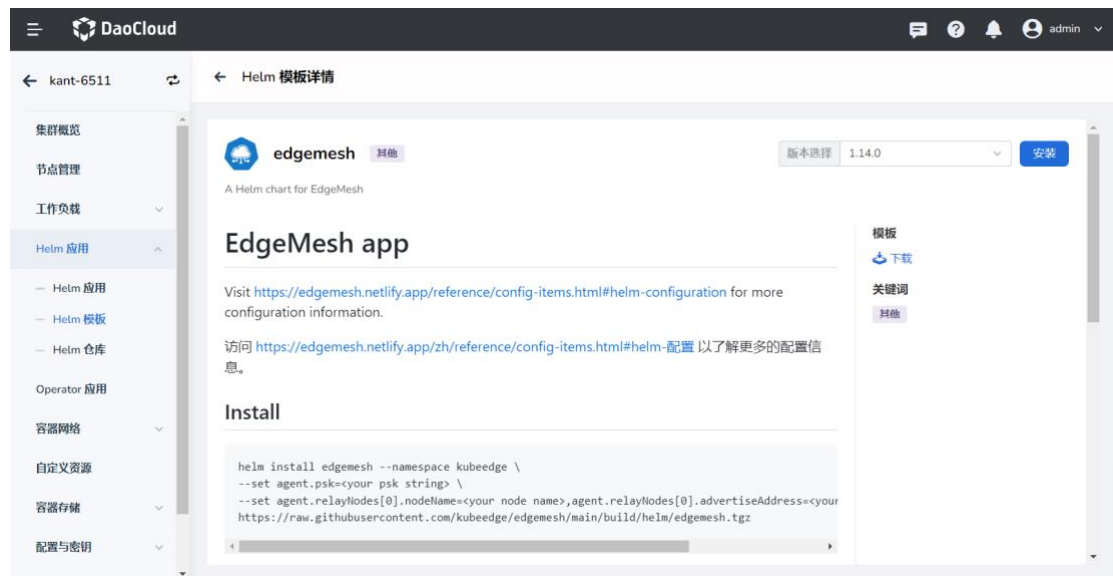
1. 选择左侧导航栏的 **容器管理** -> **集群列表**，进入集群列表页面，点击 **集群名称**，进入集群详情页。
2. 选择左侧菜单 **Helm 应用** -> **Helm 模板**，在 addon 仓库下找到 **edgemesh** 插件。



## Helm 模板

3. 点击 **edgemesh** 条目，进入模板详情页。
4. 在页面右上角选择 EdgeMesh 版本，点击 **安装** 按钮，进入 EdgeMesh 安装页面。

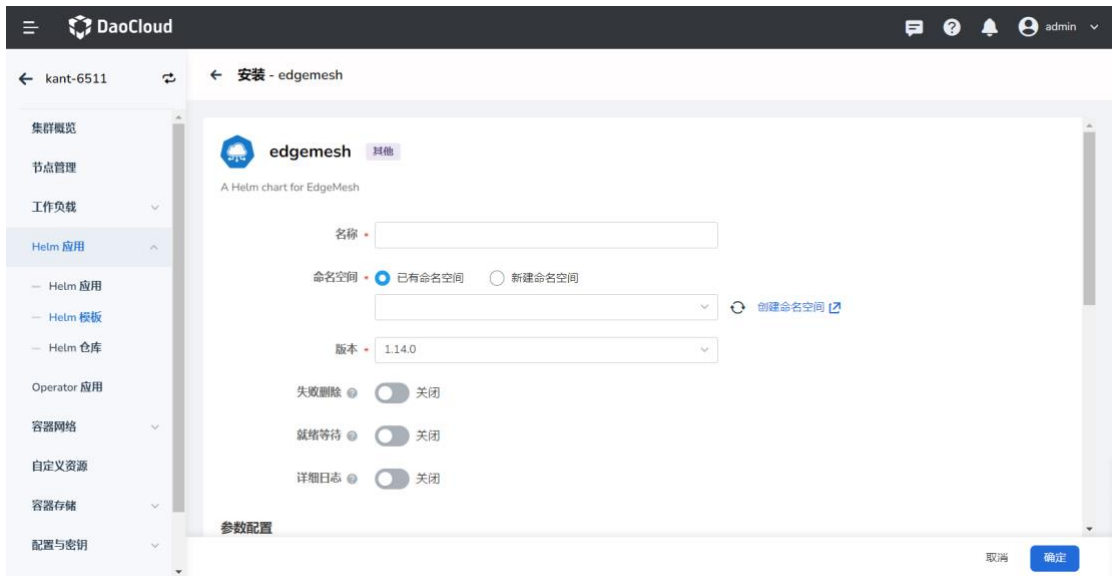




## edgemesh 安装

### 5. 填写 edgemesh 基础配置。

- 名称：小写字母、数字字符或 - 组成，并且必须以字母开头及字母或数字字符结尾。
- 命名空间：EdgeMesh 应用所在命名空间。如果命名空间没有创建，可以选择 **新建命名空间**。
- 版本：结合实际业务需求，选择想要安装的 EdgeMesh 版本。
- 失败删除：开启后，将默认同步开启安装等待。将在安装失败时删除安装。
- 就绪等待：启用后，将等待应用下所有关联资源处于就绪状态才标记应用安装成功。
- 详细日志：开启安装过程日志的详细输出。



## Helm 模板

### 6. YAML 参数配置。

!!! note

在默认的 YAML 配置下，需要补充设置认证密码（PSK）和中继节点（Relay Node），否则会导致部署失败。

### PSK 和 Relay Node 设置说明

# PSK: 是一种认证密码，确保每个 edgemes-agent 只有当拥有相同的“PSK 密码”时才能建立连接，更多信息请参考

# [PSK](https://edgemes.netlify.app/zh/guide/security.html)。建议使用 openssl 生成，也可以设置成自定义的随机字符串。

在节点上执行如下命令生成 PSK：

```
kubectl taint nodes --all node-role.kubernetes.io/master-
```

# Relay Node: 是指在网络通信中转发数据包的节点。它在通信的源节点和目标节点之间起到桥接的作用，

# 帮助数据包在网络中传输并绕过某些限制或障碍，EdgeMesh 中通常为云上节点，也可以添加多个中继节点。

### 参考示例

global:

imageRegistry: docker.m.daocloud.io

agent:

repository: kubeedge/edgemes-agent

tag: v1.14.0

affinity: {}

```

nodeSelector: {}
tolerations: []
resources:
  limits:
    cpu: 1
    memory: 256Mi
  requests:
    cpu: 0.5
    memory: 128Mi
psk: JugH9HP1XBouyO5pWGeZa8LtipDURrf17EJvUHcJGuQ=

relayNodes:
- nodeName: masternode ## your relay node name
  advertiseAddress:
  - x.x.x.x ## your relay node ip

modules:
  edgeProxy:
    enable: true
  edgeTunnel:
    enable: true

```

## 检验部署结果

部署完成后，可以执行以下操作检查 EdgeMesh 应用是否部署成功。

1. 选择左侧导航栏的 **容器管理** -> **集群列表**，进入集群列表页面，点击 **集群名称**，进入集群详情页。
2. 选择左侧菜单 **Helm 应用**，进入 Helm 应用列表页面。
3. 查看 Helm 应用的状态，当前状态为 **已部署** 表示 EdgeMesh 应用部署成功。

EdgeMesh 部署成功

EdgeMesh 部署成功

下一步：[创建服务](#)

# 服务

应用网格提供了服务管理功能，创建服务时给服务绑定应用实例，并配置访问端口，从而可以实现节点上应用间的相互访问。

## 创建服务

### !!! 说明

创建的服务确保可以访问的前提是：必须在发起访问的节点上安装 EdgeMesh 应用。

操作步骤如下：

- 1. 选择左侧导航栏的 **云边协同**，进入边缘单元列表页面，点击 **边缘单元名称**，进入边缘单元详情页。
- 2. 选择左侧菜单 **应用网格 -> 服务**，点击服务列表右上角 **创建服务** 按钮。

### 创建服务

### 创建服务

- 3. 填写相关参数。

参数	说明	举例
访问类型	【类型】无须填写【含义】指定 Pod 服务发现的方式，默认集群内访问（ClusterIP）。	ClusterIP
服务名称	【类型】必填【含义】输入新建服务的名称。【注意】请输入 4 到 63 个字符的字符串，可以包含小写英文字母、数字和中划线（-），并以小写英文字母开头，小写英文字母或数字结尾。	Svc-01
命名空间	【类型】必填【含义】选择新建服务所在的命名空间。关于命名空	default

参数	说明	举例
	间更多信息请参考 <a href="#">命名空间概述</a> 。【注意】请输入 4 到 63 个字符的字符串，可以包含小写英文字母、数字和中划线 (-)，并以小写英文字母开头，小写英文字母或数字结尾。	值
标签选择器	【类型】必填【含义】添加标签，Service 根据标签选择 Pod，填写后点击“添加”。	app:job01
端口配置	【类型】必填【含义】为服务添加协议端口，需要先选择端口协议类型，目前支持 TCP、UDP 两种传输协议。 <b>端口名称</b> ：输入自定义的端口的名称。 <b>服务端口 (port)</b> ：Pod 对外提供服务的访问端口。 <b>容器端口 (targetport)</b> ：工作负载实际监听的容器端口，用来对集群内暴露服务。	
会话保持	【类型】选填【含义】开启后，相同客户端的请求将转发至同一 Pod	开启
会话保持最大时长	【类型】选填【含义】开启会话保持后，保持的最大时长，默认为 30 秒	30 秒
标签	【类型】选填【含义】为服务添加标签	
注解	【类型】选填【含义】为服务添加注解	

### 创建服务

#### 创建服务

4. 点击 **确定**，即创建服务成功，返回到服务列表页面，可以在服务列表中查看服务对应的访问端口。

!!! tip

也可以通过 \_\_YAML 创建\_\_ 一个服务。

## 更新服务

服务支持更新服务别名、标签选择器、端口配置、会话保持等设置。

更新服务操作流程如下：

1. 进入边缘单元详情页，选择左侧导航栏的 **应用网格** -> **服务** 。
2. 点击服务名称，进入服务详情页，在页面右上角点击 **⋮** 按钮，在弹出菜单中选择 **更新**，可以对服务别名、标签选择器、端口配置、会话保持等设置进行修改操作。

更新服务

更新服务

## 查看事件

支持查看服务事件信息。

服务详情页，选择 **事件** tab，查看服务事件信息。

查看服务事件

查看服务事件

## 删除服务

1. 进入边缘单元详情页，选择左侧导航栏的 **应用网格** -> **服务** 。
2. 点击服务名称，进入服务详情页，在页面右上角点击 **⋮** 按钮，在弹出菜单中选择 **删除**，可以对服务别名、标签选择器、端口配置、会话保持等设置进行修改操作。

删除服务

删除服务

# Modbus 设备接入实践

本文介绍基于 Modbus 协议的终端设备接入边缘计算平台，并与云端交互的方法。本文以一个模拟设备为例，介绍整体实现流程。

接入流程：

graph TD

```
A(开始) --> B(终端设备连接边缘节点)
B --> C(创建设备模型)
C --> D(创建终端设备)
D --> E(部署设备 Mapper)
E --> F(结束)
```

```
classDef plain fill: #ddd,stroke:#fff,stroke-width:1px,color:#000;
classDef k8s fill: #326ce5,stroke:#fff,stroke-width:1px,color:#fff;
classDef cluster fill: #fff,stroke:#bbb,stroke-width:1px,color:#326ce5;
```

```
class A,F plain
class B,C,D,E cluster
```

```
click B "https://docs.daocloud.io/kant/best-practice/modbus-device-demo.html#_2"
click C "https://docs.daocloud.io/kant/best-practice/modbus-device-demo.html#_3"
click D "https://docs.daocloud.io/kant/best-practice/modbus-device-demo.html#_4"
click E "https://docs.daocloud.io/kant/best-practice/modbus-device-demo.html#_5"
```

## 准备工作

- 模拟 Modbus 协议终端设备，本文的终端设备以个工作负载来模拟

模拟设备镜像：**release-ci.daocloud.io/kant/modbusmock:v0.1.0**

- 设备 Mapper，基于 KubeEdge DMI 框架开发，实现了 Modbus 协议设备数据采集，此

mapper 仅适用于上文模拟设备

Mapper 镜像：**release-ci.daocloud.io/kant/devicedemo:v1.1**

- 边缘节点，节点接入要求参见[边缘节点接入要求](#)

## 终端设备连接边缘节点

由于本文中 Modbus 设备为模拟设备，故我们先将模拟设备镜像下发到边缘节点运行，模拟设备连接。部署流程参考[创建工作负载](#)。

在[高级配置](#)中包含以下参数说明：

- 节点调度，选择设备需要连接的节点
- 访问配置，网络类型选择主机网络

## 创建设备模型

设备模型创建流程参见[创建设备模型](#)。

以下设备模型参数请跟如下设置保持一致：

- 访问协议名称：modbus-tcp
- 设备模型孪生属性中的 **属性名**、**属性类型**、**访问权限** 与下图保持一致

← 创建设备模型

基础信息 设备配置

孪生属性

❗ 孪生属性会明文展示所输入的信息，请不要填入敏感数据，如涉及敏感信息，请先加密。

新增孪生属性

属性名	属性类型	访问权限	属性单位	属性值区间	
temperature	int	读/写	摄氏度	-	⋮
humidity	int	读/写	克/立方米	-	⋮

标签

+ 添加

取消 上一步 确定

设备模型孪生属性



## 创建终端设备

1. 填写 **基础信息**，设备模型选择上文创建的设备模型

2. 填写 **设备配置**，模型中关联的孪生属性配置如下

temperature 属性访问方式：

offset: 1 #寄存器的偏移量

register: HoldingRegister #保存寄存器

humidity 属性访问方式：

offset: 1 #寄存器的偏移量

register: InputRegister #输入寄存器

新增孪生属性

基本信息

属性名 \* temperature

期望值 36

采集间隔 2

上报间隔 2

属性访问方式 ?

YAML 格式参数 \*

```
1 offset: 1
2 register: HoldingRegister
3
```

YAML 示例

```
NTP_timezone: Asia/Shanghai
calico_node_extra_envs:
FELIX_DEVICEROUTESOURCEADDRESS: 172.17.0.1
FELIX_CHAININSERTMODE: Append
```

取消 确定

新增孪生属性

基本信息

属性名 \*

humidity

期望值

36

采集间隔 ?

2

上报间隔 ?

2

属性访问方式 ?

YAML 格式参数 \*

```
1 offset: 1
2 register: InputRegister
3
```

YAML 示例

NTP\_timezone: Asia/Shanghai
calico\_node\_extra\_envs:
FELIX\_DEVICEROUTESOURCEADDRESS: 172.17.0.1
FELIX\_CHAININSERTMODE: Append

取消

确定

3. 填写设备 **访问配置**，配置参数如下：

**address:** 0.0.0.0:11502

← 创建设备

基础信息

设备配置

访问配置

信息确认

参数配置 ?

YAML 格式参数 \*

```
1 address: 0.0.0.0:11502
```

YAML 示例

baudRate: 4800
dataBits: 8
parity: none
serialPort: /dev/ttyUSB1
stopBits: 1

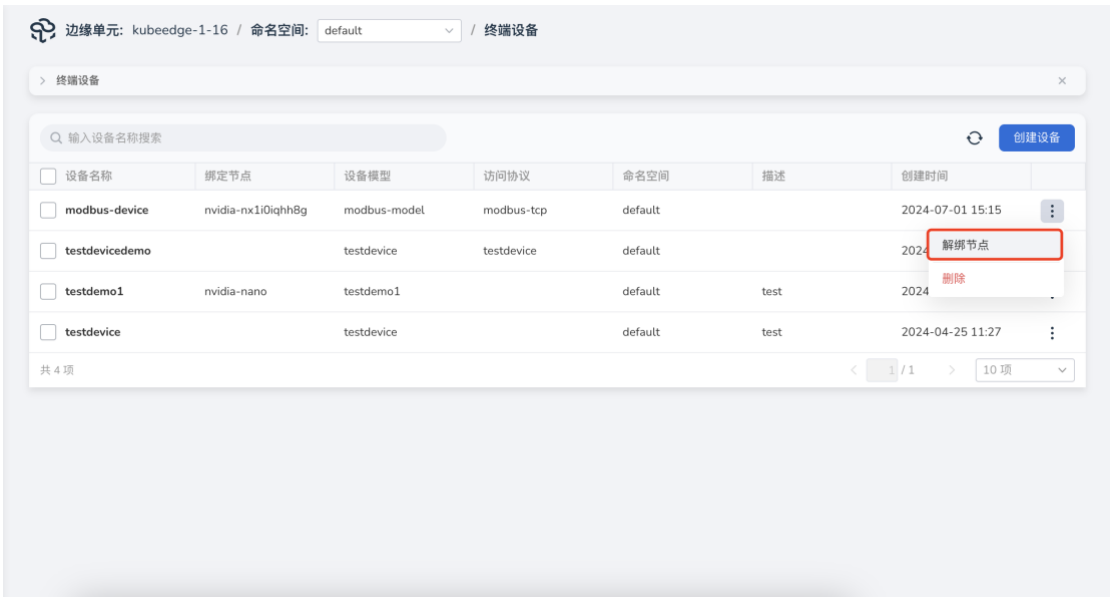
取消

上一步

下一步

## 设备访问配置

4. 设备创建成功后，自动跳转到设备列表，点击列表右侧 **绑定节点** 操作



设备孪生属性

## 部署设备 Mapper

设备 Mapper 主要用来采集设备数据，是一个无状态工作负载，Mapper 部署好后，即可获取到设备运行数据，在这个实践场景中，我们可以在设备详情页看到 Mapper 采集到的温湿度值。

在实际应用场景中，可以根据业务需要，修改 Mapper 的实现和行为，完成 DMI 标准的数据推送、数据拉取、数据存数据库、边端本地化处理、设备联动等功能。

!!! note

创建终端设备和部署设备 Mapper 顺序可以调换。

设备 Mapper 镜像地址：

release-ci.daocloud.io/kant/devicedemo: v1.1

设备 Mapper 部署流程参见[创建工作负载](#)

创建工作负载参数说明：

- **容器配置 -> 数据存储**，选择主机路径，**主机路径** /etc/kubeedge 映射 **容器路径** /etc/kubeedge，权限为 **读写权限**

- 高级配置 -> 访问配置，网络类型选择 **主机网络**
- 高级配置 -> 节点调度，选择跟设备绑定的同一节点

← 创建工作负载

1 2 3  
基本信息 容器配置 高级配置

> 健康检查 选项

> 环境变量 选项

> 数据存储 选项

节点路径映射

类型	主机路径	容器路径 (mountPath)	子路径 (subPath)	权限
主机路径 (HostPath)	/etc/kubeedge	/etc/kubeedge	使用子路径挂载本地磁盘。实例	读写

+ 添加

确定

> 安全设置 选项

取消 上一步 下一步

← 创建工作负载

1 2 3  
基本信息 容器配置 高级配置

节点调度 标签与注解 访问配置 升级策略

网络类型

☐ 不可访问  
工作负载不可访问

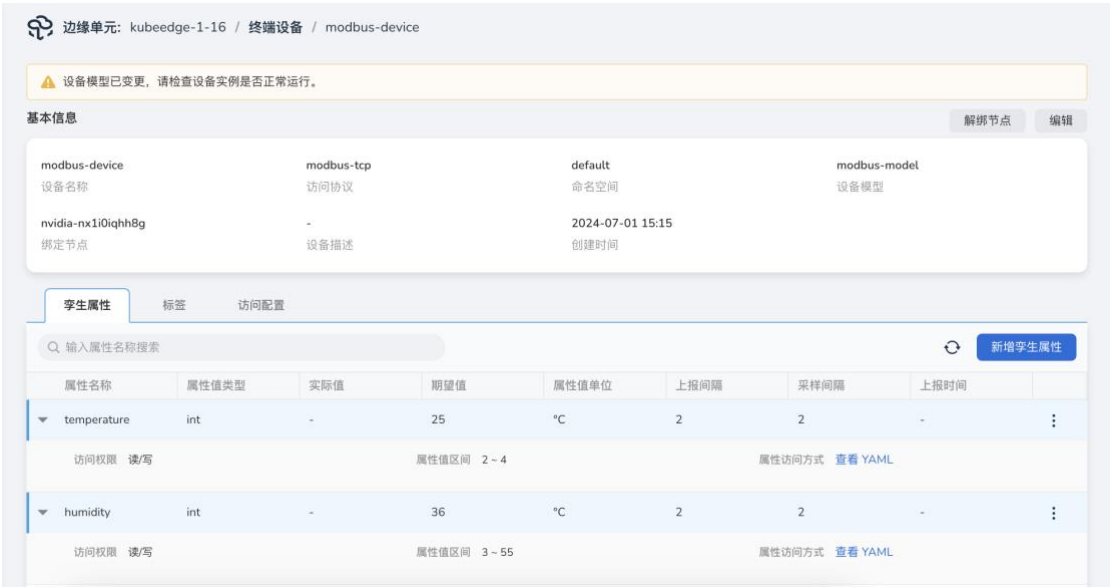
☐ 端口映射  
配置端口映射后，流向主机端口的流量会映射到对应的容器端口

☒ 主机网络  
使用边缘节点的网络，此时容器与主机间不做网络隔离，使用同一个 IP

取消 上一步 确定

## 验证设备运行效果

以上步骤完成后，前往设备详情页，可以在界面看到设备运行数据，此场景中我们可以看到温度和湿度值，本文中 mock 的设备，数据值不会发生改变。



设备运行数据

# 智能设备控制

本文介绍基于自定义协议的终端设备接入边缘计算平台，并与云端交互的方法。以一个手势控制灯开关的场景为例，介绍整体实现流程。

## 准备工作

- 边缘节点，节点接入要求参见[边缘节点接入要求](#)
- LED 灯，带有控制单元
- LED 灯驱动 mapper，mapper 开发参阅[如何开发设备驱动应用 mapper](#)

## 创建终端设备

1. 将 LED 灯驱动 mapper 部署到边缘节点，部署流程参考[创建工作负载](#)
2. 进入边缘单元详情页，选择左侧菜单 **边缘资源** -> **终端设备**，点击终端设备列表右上

角 **创建设备** 按钮。

## 创建设备

### 创建设备

#### 3. 填写基础信息。

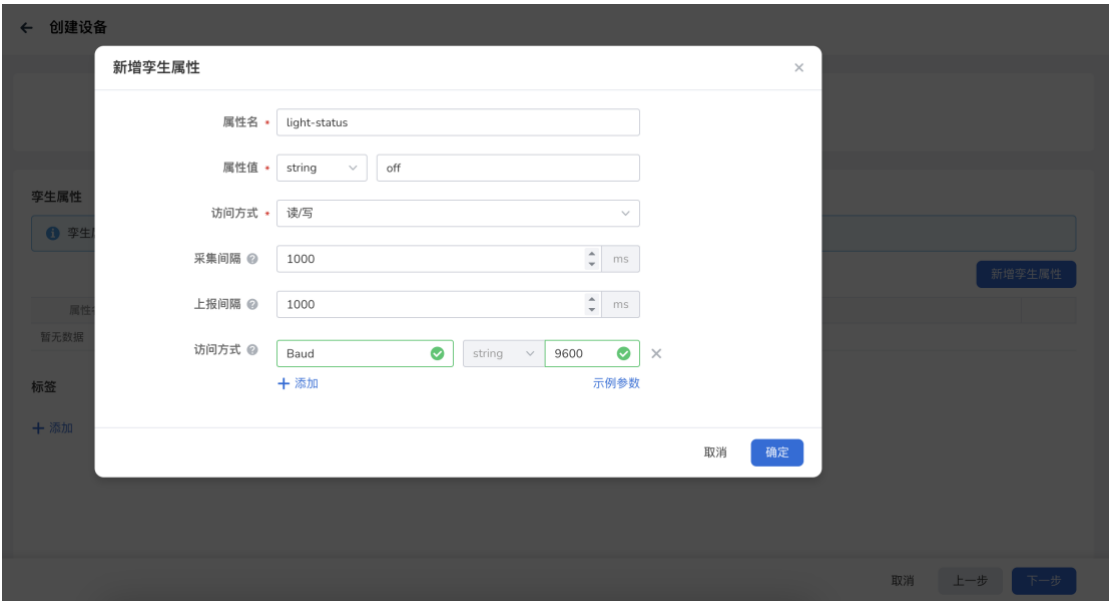
### 创建设备

#### !!! note

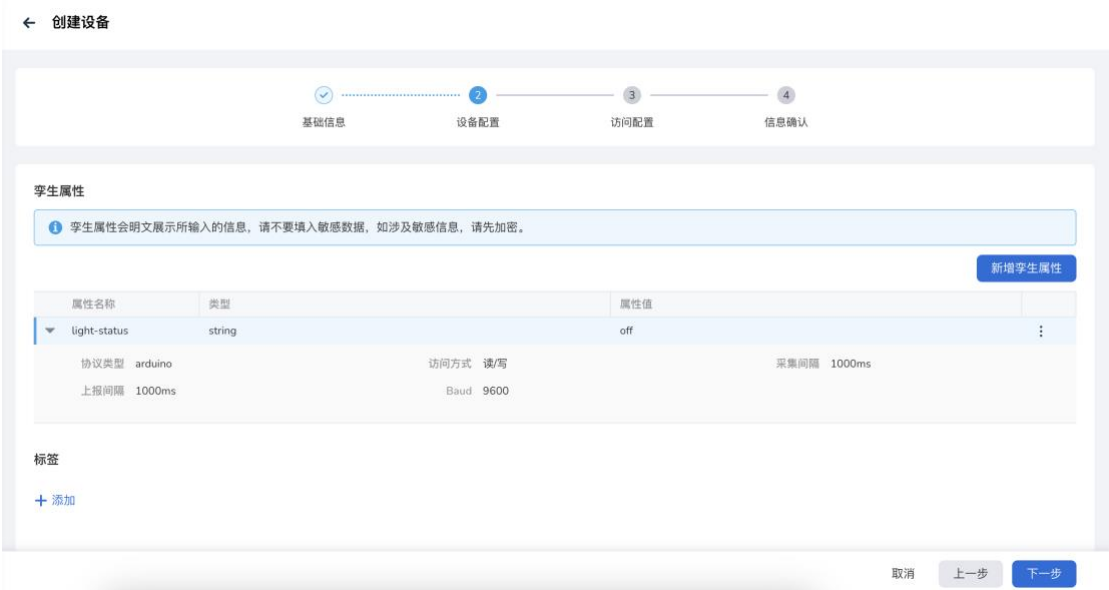
访问协议必须和 `mapper` 定义的协议名称一致。

#### 4. 填写设备配置，添加设备孪生属性和标签。

- 属性值为设备期望值。
- 访问方式必须和 `mapper` 定义的键值数据保持一致。



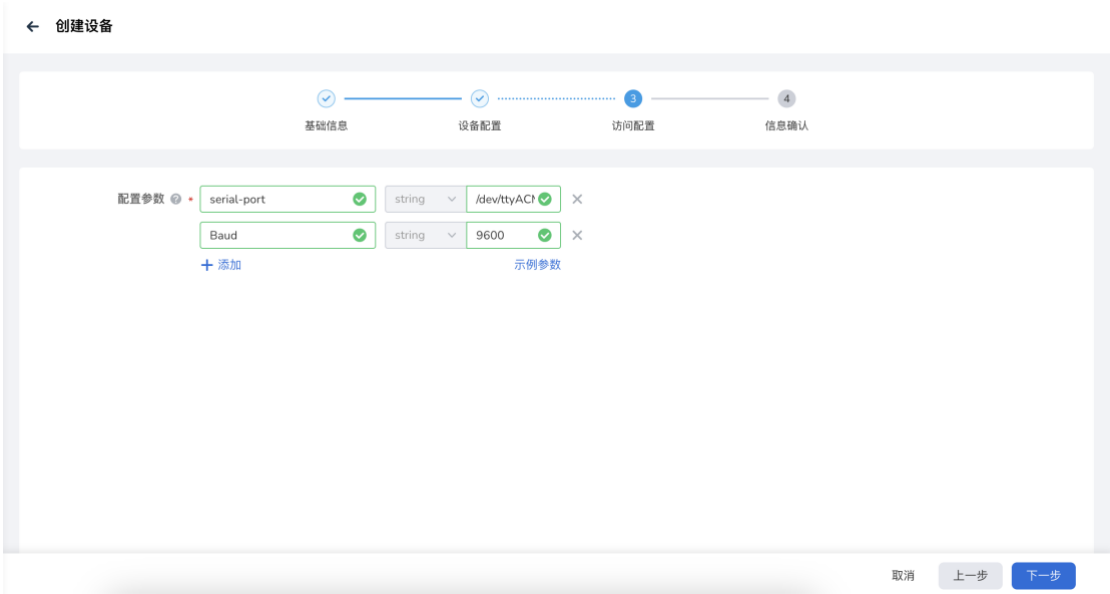
添加孪生属性



设备配置

5. 填写设备访问配置。

平台连接到设备的访问参数，在这个场景中，设备的访问路径为 /dev/ttyACM0，波特率为 9600。

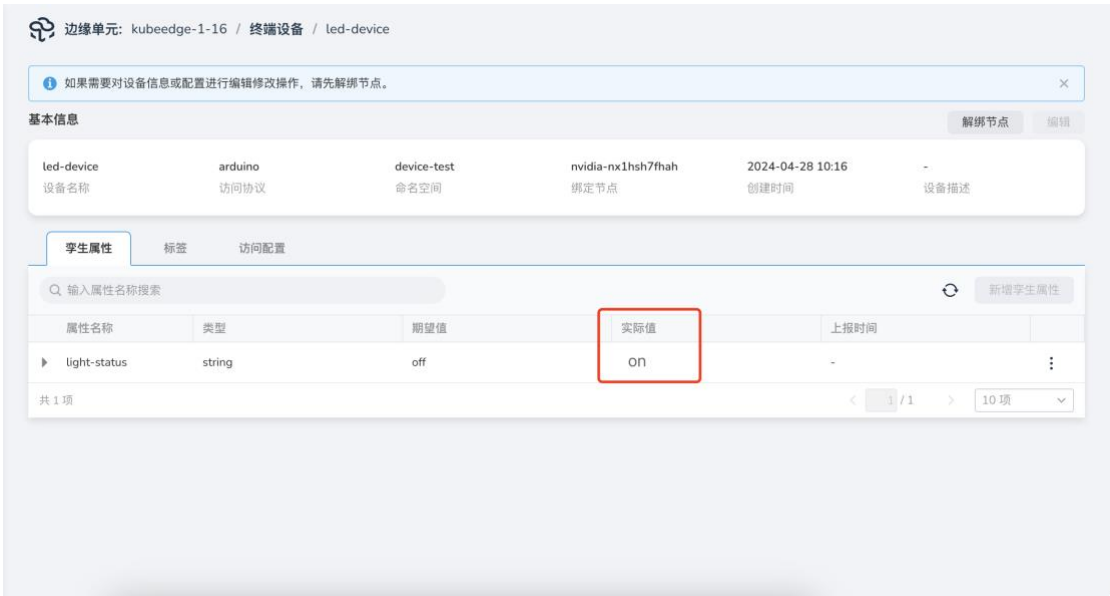


访问配置

- 6. 信息确认，确认所配置的信息无误，点击 **确定** ，完成设备创建。

验证运行效果

- 1. 设备列表，点击 设备名称，进入设备详情页，可以查看设备上报的状态值。



设备详情

- 2. 编辑设备孪生属性，修改设备期望值，LED 灯状态改变。



# 如何开发设备驱动应用 mapper

本文介绍设备驱动应用 mapper 的开发和部署流程。

1. 进入到 KubeEdge 仓库的

`kubeedge/staging/src/github.com/kubeedge/mapper-framework` 目录下执行 `make`

`generate` 命令：

`make generate`

Please input the mapper name (like 'Bluetooth', 'BLE'): foo # (1)!

1. 这里的协议是后续需要再创建 `deviceModel` 的时候填写的
- 执行完成后，会在 `mapper-framework` 的同级目录生成与协议同名的 `mapper` 代码目录。



`mapper` 目录

- 可以将该目录复制出来进行 `mapper` 开发，需要关注的地方是 `driver` 目录下的代码：

该文件对应 `mapper` 对设备的操作实现，主要实现 `InitDevice`(初始化设备)、`GetDeviceData`(获取设备数据)、`SetDeviceData`(给设备赋值)、`StopDevice`(停止设备)

```

package driver

import (
    "github.com/tarm/serial"
    "log"
    "os"
    "os/signal"
    "sync"
    "syscall"
)

func NewClient(protocol ProtocolConfig) (*CustomizedClient, error) {
    client := &CustomizedClient{
        ProtocolConfig: protocol,
        deviceMutex:    sync.Mutex{},
        // TODO initialize the variables you added
    }
    return client, nil
}

func (c *CustomizedClient) InitDevice() error {
    // TODO: add init operation
    // you can use c.ProtocolConfig
    return nil
}

func (c *CustomizedClient) GetDeviceData(visitor *VisitorConfig) (interface{},
error) {
    // TODO: add the code to get device's data
    // you can use c.ProtocolConfig and visitor
    // 打开串口设备
    return "ok", nil
}

func (c *CustomizedClient) SetDeviceData(data interface{}, visitor *VisitorCon
fig) error {
    // TODO: set device's data
    // you can use c.ProtocolConfig and visitor
    // 打开串口设备
    config := &serial.Config{
        Name: "/dev/ttyACM0", // 替换为您的串口名称, 例如 "/dev/ttyUSB0" (Li
nux) 或 "COM1" (Windows)
        Baud: 9600,
    }
}

```

```

port, err := serial.OpenPort(config)
if err != nil {
    log.Fatal(err)
}
defer port.Close()

// 监听操作系统的中断信号，以便在程序终止前关闭串口连接
signalCh := make(chan os.Signal, 1)
signal.Notify(signalCh, os.Interrupt, syscall.SIGTERM)
go func() {
    <-signalCh
    port.Close()
    os.Exit(0)
}()
_, err = port.Write([]byte(data.(string)))
if err != nil {
    log.Fatal(err)
}

return nil
}

func (c *CustomizedClient) StopDevice() error {
    // TODO: stop device
    // you can use c.ProtocolConfig
    return nil
}

```

- 修改 config.yaml 中的 protocol 字段，填写前文定义的协议名称

```

yaml  title="config.yaml"          grpc_server:          socket_path:
    /etc/kubeedge/arduino.sock    common:              name: arduino-mapper
version: v1.13.0                 api_version: v1.0.0   protocol: arduino # (1)!
address: 127.0.0.1               edgecore_sock: /etc/kubeedge/dmi.sock

```

1. 改为你的协议名称

## 2. 部署 mapper 应用

### === “二进制部署”

1. 在项目的主目录使用 `go build ./cmd/main.go` 编译出对应架构的二进制文件，比如编译 linux 环境下的可执行文件

```

```shell
GOOS=linux GOARCH=amd64 go build ./cmd/main.go -o {输出的文件名称}

```

```
# (1)!
```

```

1. -o 参数可以不填
2. 将二进制文件上传到设备绑定的节点，注意需要在可执行文件所在目录将 config.yaml 文件放在这里，否则会报文件找不到的错误

```
```shell
# 目录中应该包含以下两个文件，其中 main 是可执行文件，config.yaml 是配置
文件
root@nx:~/device-test# ls
config.yaml  main
# 接下来在该目录执行 ./main 即可
```

```

### === “容器化部署”

1. 使用提供的 Dockerfile 文件进行编译
2. 编译完成后，使用 resource 目录下的 ConfigMap 和 Deployment 资源进行部署

!!! note

修改 Deployment 的镜像为实际编译出的镜像名称，ConfigMap 也需要修改 `protocol` 字段。

```
```yaml
apiVersion: v1
kind: ConfigMap
metadata:
  name: cm-mapper
data:
  configData: |
    grpc_server:
      socket_path: /etc/kubeedge/arduino.sock
    common:
      name: arduino-mapper
      version: v1.13.0
      api_version: v1.0.0
      protocol: arduino # 改为你的协议名称
      address: 127.0.0.1
      edgecore_sock: /etc/kubeedge/dmi.sock
```

```

以上，这就完成了设备驱动应用 mapper 的开发。